

NOVA

IMS

Information
Management
School

MDSAA

Master's Degree Program in
Data Science and Advanced Analytics

**Machine Learning and Genetic Programming to Enhance
Diagnosis and Prognosis in Breast Cancer Care**

Catarina Natário Moreira

Master Thesis

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

MACHINE LEARNING AND GENETIC PROGRAMMING TO ENHANCE DIAGNOSIS AND PROGNOSIS IN BREAST CANCER CARE

by

Catarina Natário Moreira

Master Thesis presented as partial requirement for obtaining the Master's Degree in Data Science and Advanced Analytics, with a specialization in Data Science

Supervised by

Leonardo Vanneschi, PhD, NOVA Information Management School (NOVA IMS)

July, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, 2024

ABSTRACT

Breast cancer is a pathological condition characterized by uncontrolled cell multiplication within mammary tissue and is undoubtedly the malignant neoplasm that most affects women worldwide. Nevertheless, comprehending its etiology and its therapeutic strategies has progressively evolved throughout history and, in recent years, significant advancements in medicine and artificial intelligence have bolstered the deployment of machine learning and genetic programming algorithms. The use of these algorithms has not only caused a significant impact but has also demonstrated remarkable efficacy in developing predictive models for precise and expeditious patient disease diagnosis and prognosis.

The process that guided my research began with the identification and definition of the primary problem: the uncertainty in breast cancer diagnosis and prognosis, along with the limited availability of diverse diagnostic and prognostic algorithms for accurate results. This challenge served as the impetus for my thesis. Consequently, my research pursued the objective of applying machine learning and genetic programming techniques to improve the accuracy and efficiency of breast cancer diagnosis and prognosis. In terms of diagnosis, the primary problem to address is distinguishing between malignant and benign breast cytology. Regarding prognosis, my research focused on two main problems: firstly, distinguishing between recurrent and non-recurrent breast cancer, and secondly, predicting the recurrence time for recurrent cases as well as the disease-free time for non-recurrent cases.

The research encompasses analyzing data from the Breast Cancer Wisconsin repository. This includes understanding the data, preprocessing, feature selection, and implementing and training machine learning and genetic programming algorithms to accurately diagnose and prognose cancer. Subsequently, the performance of the developed models was rigorously evaluated using different metrics.

Upon result analysis, Machine Learning and Genetic Programming algorithms significantly enhance Breast Cancer diagnosis, with strong performances from Multi-Layer Perceptron, Voting Classifier, and Stacking Classifier. However, prognosis outcomes were generally suboptimal, except for the top performance of the Voting Classifier and promising results from TPOT in classification. The Gradient Boosting Regressor consistently excelled in regression tasks.

In conclusion, the development of the research methodology culminated in the successful implementation of machine learning and genetic programming models for breast cancer diagnosis and prognosis. This achievement deepened my expertise in artificial intelligence, machine learning, and genetic programming, highlighting their potential to improve diagnostic and prognostic processes. The developed models offer a promising approach to making cancer diagnosis and prognosis more accurate, efficient, and less invasive for patients, thereby contributing significantly to the field of medical science and patient care.

KEYWORDS

Breast Cancer; Diagnostic; Prognostic; Artificial Intelligence; Machine Learning; Genetic Programming; Classification; Regression

Sustainable Development Goals (SDG):



TABLE OF CONTENTS

STATEMENT OF INTEGRITY.....	I
ABSTRACT	II
LIST OF FIGURES.....	VII
LIST OF TABLES	IX
LIST OF ABBREVIATIONS AND ACRONYMS	XX
1. INTRODUCTION	1
1.1. CONTEXT, PROBLEM, AND MOTIVATION	1
1.2. OBJECTIVES.....	2
1.3. DOCUMENT STRUCTURE	3
2. LITERATURE REVIEW	4
2.1. EMPIRICAL WORK ON THE DIAGNOSIS	4
2.2. EMPIRICAL WORK ON THE PROGNOSIS	8
2.3. EMPIRICAL WORK ON BOTH: DIAGNOSIS AND PROGNOSIS	9
2.4. EMPIRICAL WORK ON SURVIVAL OF BREAST CANCER PATIENTS	12
2.5. CONCLUSION.....	14
3. METHODOLOGY	15
3.1. CROSS – INDUSTRY STANDARD PROCESS FOR DATA MINING.....	15
3.2. DIAGNOSTIC METHODOLOGY	16
3.2.1. <i>Business Understanding</i>	16
3.2.2. <i>Data Understanding</i>	17
3.2.2.1. Exploratory Analysis.....	17
3.2.2.2. Metric and Non-metric Features.....	17
3.2.2.3. Data Visualization.....	18
3.2.3. <i>Data Preprocessing</i>	18
3.2.3.1. New Variables	19
3.2.3.2. Target Split	19
3.2.3.3. Standardization	19
3.2.3.4. Outlier Removal	19
3.2.3.5. Imbalanced Data	20
3.2.3.6. Feature Selection	20
3.2.3.7. Conclusion.....	23
3.2.4. <i>Modeling</i>	23
3.2.4.1. Final Select Features	23
3.2.4.2. Train and Test Split.....	24
3.2.4.3. Models	24
3.2.5. <i>Evaluation</i>	28
3.3. PROGNOSTIC METHODOLOGY.....	29
3.3.1. <i>Business Understanding</i>	29
3.3.2. <i>Data Understanding</i>	30
3.3.2.1. Exploratory Analysis.....	30
3.3.2.2. Metric and Non-metric Features.....	31
3.3.2.3. Data Visualization.....	31
3.3.3. <i>Data Preprocessing</i>	32
3.3.3.1. New Variables	32
3.3.3.2. Target Split	32

3.3.3.3.	Standardization	33
3.3.3.4.	Outlier Removal	33
3.3.3.5.	Imbalanced Data	33
3.3.3.6.	Feature Selection	33
3.3.3.7.	Conclusion	36
3.3.4.	<i>Modeling</i>	36
3.3.4.1.	Final Select Features	36
3.3.4.2.	Train and Test Split.....	37
3.3.4.3.	Models	37
3.3.5.	<i>Evaluation</i>	39
4.	RESULTS AND DISCUSSION.....	41
4.1.	DIAGNOSTIC RESULTS.....	41
4.2.	PROGNOSTIC RESULTS.....	49
5.	CONCLUSIONS AND FUTURE WORK	68
5.1.	LIMITATIONS	70
5.2.	FUTURE WORK	71
5.3.	FINAL THOUGHTS	71
	BIBLIOGRAPHICAL REFERENCES	73
	APPENDIX A.....	76
A.1	DIAGNOSTIC METHODOLOGY – BUSINESS UNDERSTANDING	76
A.2	DIAGNOSTIC METHODOLOGY – DATA UNDERSTANDING	77
A.3	DIAGNOSTIC METHODOLOGY – DATA PREPROCESSING	83
A.4	DIAGNOSTIC METHODOLOGY – MODELLING.....	85
B.1	PROGNOSTIC METHODOLOGY – BUSINESS UNDERSTANDING	88
B.2	PROGNOSTIC METHODOLOGY – DATA UNDERSTANDING	89
B.3	PROGNOSTIC METHODOLOGY – DATA PREPROCESSING	95
B.4	PROGNOSTIC METHODOLOGY – MODELLING	96
		97
C.1	DIAGNOSTIC RESULTS - Naïve BAYES	99
C.2	DIAGNOSTIC RESULTS - LOGISTIC REGRESSION (LR).....	100
C.3	DIAGNOSTIC RESULTS – DECISION TREE (DT).....	101
C.4	DIAGNOSTIC RESULTS – RANDOM FOREST (RF).....	104
C.5	DIAGNOSTIC RESULTS – NEURAL NETWORK (NN).....	107
C.6	DIAGNOSTIC RESULTS – PROBABILISTIC NEURAL NETWORK (PROBABILISTIC NN)	110
C.7	DIAGNOSTIC RESULTS – LINEAR DISCRIMINANT ANALYSIS (LDA)	111
C.8	DIAGNOSTIC RESULTS – SUPPORT VECTOR MACHINE (SVM).....	112
C.9	DIAGNOSTIC RESULTS – MULTI-LAYER PERCEPTRON WITH GRID SEARCH (MLP w/ GS)	114
C.10	DIAGNOSTIC RESULTS – K-NEAREST NEIGHBOR (KNN).....	116
C.11	DIAGNOSTIC RESULTS – ENSEMBLE BAGGING CLASSIFIER (BC)	118
C.12	DIAGNOSTIC RESULTS – ENSEMBLE ADABOOST CLASSIFIER (AC).....	121
C.13	DIAGNOSTIC RESULTS – ENSEMBLE VOTING CLASSIFIER (VC)	124
C.14	DIAGNOSTIC RESULTS – ENSEMBLE GRADIENT BOOSTING CLASSIFIER (GBC).....	126
C.15	DIAGNOSTIC RESULTS – ENSEMBLE STACKING CLASSIFIER (SC).....	130
C.16	DIAGNOSTIC RESULTS – ENSEMBLE EXTREME GRADIENT BOOSTING CLASSIFIER (XGBC w/ GS)	131
C.17	DIAGNOSTIC RESULTS – TREE-BASED PIPELINE OPTIMIZATION TOOL (TPOT)	133
C.18	DIAGNOSTIC RESULTS – FINAL RESULTS	136
D.1	PROGNOSTIC RESULTS – CLASSIFICATION PROBLEM – Naïve BAYES	139

D.2 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – LOGISTIC REGRESSION (LR)	140
D.3 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – DECISION TREE (DT)	141
D.4 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – RANDOM FOREST (RF)	144
D.5 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – NEURAL NETWORK (NN)	147
D.6 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – SUPPORT VECTOR MACHINE (SVM)	148
D.7 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – MULTI-LAYER PERCEPTRON WITH GRID SEARCH (MLP w/ GS)	149
D.8 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – K-NEAREST NEIGHBOR (KNN)	151
D.9 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE BAGGING CLASSIFIER (BC)	152
D.10 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE ADABOOST CLASSIFIER (AC)	153
D.11 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE VOTING CLASSIFIER (VC)	154
D.12 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE GRADIENT BOOSTING CLASSIFIER (GBC)	156
D.13 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE STACKING CLASSIFIER (SC)	157
D.14 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE EXTREME GRADIENT BOOSTING CLASSIFIER (XGBC w/ GS)	158
D.15 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – TREE-BASED PIPELINE OPTIMIZATION TOOL(TPOT)	160
D.16 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – FINAL RESULTS	163
E.1 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – LINEAR REGRESSION (LIR)	166
E.2 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – GAUSSIAN PROCESS REGRESSION (GPR)	167
E.3 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – DECISION TREE (DT)	168
E.4 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – RANDOM FOREST (RF)	169
E.5 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ELASTIC NET (EN)	170
E.6 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – RIDGE REGRESSION (RR)	171
E.7 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – BAYESIAN RIDGE REGRESSION (BRR)	172
E.8 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – NEURAL NETWORK (NN)	173
E.9 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – MULTI-LAYER PERCEPTRON WITH GRID SEARCH (MLP)	174
E.10 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – SUPPORT VECTOR MACHINE (SVM)	176
E.11 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – K-NEAREST NEIGHBOR (KNN)	177
E.12 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE BAGGING REGRESSOR (BR)	178
E.13 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE ADABOOST REGRESSOR (AR)	179
E.14 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE GRADIENT BOOSTING REGRESSOR (GBR)	180
E.15 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE VOTING REGRESSOR (VR)	181
E.16 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE STACKING REGRESSOR (SR)	183
E.17 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE EXTREME GRADIENT BOOSTING REGRESSOR (XGBR)	184
E.18 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – TREE-BASED PIPELINE OPTIMIZATION TOOL (TPOT)	186
E.19 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – FINAL RESULTS	187
ANNEXES	191
ANNEX 1: CRISP-DM	191
ANNEX 2: MACHINE LEARNING PIPELINE AUTOMATED BY TPOT	191
ANNEX 3: GANT CHART	192

LIST OF FIGURES

Figure A.1 – Distinct observations in WDBC dataset	77
Figure A.2 – Descriptive statistics in WDBC dataset	78
Figure A.3 – Types of the columns in WDBC dataset	79
Figure A.4 – Number of people with different breast cancer diagnoses	80
Figure A.5 – Radius SE distribution – histogram and displot	80
Figure A.6 – Relationship between mean numerical variable	81
Figure A.7 – Relationship between mean numerical variable	82
Figure A.8 – Imbalanced Data	83
Figure A.9 – Balanced Data	83
Figure A.10 – Part of final dataframe with the most important features	84
Figure A.11 – Function IndividualModel_Scoring(X_train, X_val, y_train, y_tval, model, model_name, index)	85
Figure A.12 – Function model_scores(n_split, X, y, model)	86
Figure A.13 – Function dataframe_mean_std(df)	87
Figure B.1 – Distinct observations in WPBC dataset	89
Figure B.2 – Descriptive statistics in WPBC dataset	90
Figure B.3 – Types of the columns in WPBC dataset	91
Figure B.4 – Number of people with different breast cancer prognoses	92
Figure B.5 – Radius SE distribution – histogram and displot	92
Figure B.6 – Relationship between mean numerical variable	93
Figure B.7 – Relationship between mean numerical variable	94
Figure B.8 – Imbalanced Data	95
Figure B.9 – Balanced Data	95
Figure B.10 – Part of final dataframe with the most important features in Classification Problem	96
Figure B.11 – Part of final dataframe with the most important features in Regression Problem	96
Figure B.12 – Function IndividualModel_Scoring(X_train, X_val, y_train, y_tval, model, model_name, index)	97
Figure B.13 – Function model_scores(n_split, X, y, model) in Regression model	98
Figure C.1 – Accuracy of Ensemble Bagging Classifier with different <i>N Estimators</i> parameter	119
Figure C.2 – Accuracy of Ensemble Adaboost Classifier with different <i>N Estimators</i> parameter	122
Figure C.3 – Estimators used in the Voting Classifier model	124

Figure C.4 – Accuracy of Ensemble Gradient Boosting Classifier with different <i>Random State</i> parameter	128
Figure C.5 – Accuracy of Ensemble Gradient Boosting Classifier with different <i>N Estimators</i> parameter	128
Figure C.6 – Estimators used in the Stacking Classifier model.....	130
Figure C.8 – Accuracy of TPOT best model with the best parameters.....	135
Figure C.8 – F-1 Scores of all models	137
Figure C.9 – Specificity of all models.....	137
Figure C.10 – Sensitivity of all models	138
Figure C.11 – ROC Curve	138
Figure D.1 – Estimators used in the Voting Classifier model	154
Figure D.2 – Estimators used in the Stacking Classifier model	157
Figure D.3 – Accuracy of TPOT best model with the best parameters	161
Figure D.4 – F-1 Scores of all model.....	163
Figure D.5 – Specificity of all models	164
Figure D.6 – Sensitivity of all models	164
Figure D.7 – ROC Curve	165
Figure E.1 – Estimators used in the Voting Regressor model	181
Figure E.2 – Estimators used in the Stacking Regressor model	183
Figure E.3 – Accuracy of TPOT best model with the best parameters.....	186
Figure E.4 – MAE of all models	188
Figure E.5 – MSE of all models.....	188
Figure E.6 – RMSE of all models.....	189
Figure E.7 – R^2 of all models.....	189
Figure E.8 – MAPE of all models	190

LIST OF TABLES

Table A.1 - Real-Valued features.....	76
Table A.2 - Lists created with different number of features.....	85
Table B.1 - Real-Valued features.....	88
Table B.2 - Lists with final features in Classification Problem.....	96
Table B.3 - Lists with final features in Regression Problem	97
Table C.1 - Accuracy of Naïve Bayes with different numbers of features.....	99
Table C.2 - F1-Score, Accuracy, Specificity, Sensitivity of Naïve Bayes of Final Naïve Bayes model with 3 features.....	99
Table C.3 - Accuracy of Logistic Regression with different numbers of features.....	100
Table C.4 - Accuracy of Logistic Regression with different parameters.....	100
Table C.5 - F1-Score, Accuracy, Specificity, Sensitivity of Logistic Regression of Final Logistic Regression model with 9 features	100
Table C.6 - Accuracy of Decision Tree with different number of features.....	101
Table C.7 - Accuracy of Decision Tree with different <i>Criterion</i> parameter	101
Table C.8 - Accuracy of Decision Tree with different <i>Splitter</i> parameter	101
Table C.9 - Accuracy of Decision Tree with different <i>Max Depth</i> parameter	101
Table C.10 - Accuracy of Decision Tree with different <i>Min Samples Split</i> parameter..	101
Table C.11 - Accuracy of Decision Tree with different <i>Min Samples Leaf</i> parameter..	102
Table C.12 - Accuracy of Decision Tree with different <i>Min Weight Fraction Leaf</i> parameter	102
Table C.13 - Accuracy of Decision Tree with different <i>Min Impurity Decrease</i> parameter	102
Table C.14 - Accuracy of Decision Tree best model with the best parameters	103
Table C.15 - F1-Score, Accuracy, Specificity, Sensitivity of Decision Tree of Final Decision Tree model with 9 features.....	103
Table C.16 - Accuracy of Random Forest with different number of features	104
Table C.17 - Accuracy of Random Forest with different <i>N Estimators</i> parameter.....	104
Table C.18 - Accuracy of Random Forest with different <i>Criterion</i> parameter	104
Table C.19 - Accuracy of Random Forest with different <i>Max Depth</i> parameter.....	104
Table C.20 - Accuracy of Random Forest with different <i>Min Samples Leaf</i> parameter	105
Table C.21 - Accuracy of Random Forest with different <i>Bootstrap</i> parameter	105
Table C.22 - Accuracy of Random Forest with different <i>Max Samples</i> parameter	105
Table C.23 - Accuracy of Random Forest best model with the best parameters.....	105

Table C.24 - F1-Score, Accuracy, Specificity, Sensitivity of Random Forest of Final Random Forest model with 3 features.....	105
Table C.25 - Accuracy of Neural Network with different number of features.....	107
Table C.26 - Accuracy of Neural Network with different <i>Hidden Layer Sizes</i> parameter	107
Table C.27 - Accuracy of Neural Network with different <i>Activation</i> parameter	107
Table C.28 - Accuracy of Neural Network with different <i>Solver</i> parameter	107
Table C.29 - Accuracy of Neural Network with different <i>Alpha</i> parameter	108
Table C.30 - Accuracy of Neural Network with different <i>Learning Rate</i> parameter	108
Table C.31 - Accuracy of Neural Network with different <i>Learning Rate Init</i> parameter.	108
Table C.32 - Accuracy of Neural Network with different <i>Max Iter</i> parameter.....	108
Table C.33 - Accuracy of Neural Network with different <i>Shuffle</i> parameter	109
Table C.34 - Accuracy of Neural Network best model with the best parameters.....	109
Table C.35 - F1-Score, Accuracy, Specificity, Sensitivity of Neural Network of Final Neural Network model with 3 features.....	109
Table C.36 - Accuracy of Probabilistic Neural Network with different number of features	110
Table C.37 - Accuracy of Probabilistic Neural Network best model with the best parameters	110
Table C.38 - F1-Score, Accuracy, Specificity, Sensitivity of Probabilistic Neural Network of Final Probabilistic Neural Network model with 9 features	110
Table C.39 - Accuracy of Linear Discriminant Analysis with different number of features	111
Table C.40 - Accuracy of Linear Discriminant Analysis best model with the best parameters	111
Table C.41 - F1-Score, Accuracy, Specificity, Sensitivity of Linear Discriminant Analysis of Final Linear Discriminant Analysis model with all features	111
Table C.42 - Accuracy of Support Vector Machine with different number of features.	112
Table C.43 - Accuracy of Support Vector Machine with different <i>C</i> parameter	112
Table C.44 - Accuracy of Support Vector Machine with different <i>Kernel</i> parameter ..	112
Table C.45 - Accuracy of Support Vector Machine with different <i>Gamma</i> parameter	112
Table C.46 - Accuracy of Support Vector Machine with different <i>Max Iter</i> parameter.	112

Table C.47 - Accuracy of Support Vector Machine best model with the best parameters	113
Table C.48 - F1-Score, Accuracy, Specificity, Sensitivity of Support Vector Machine of Final Support Vector Machine model with 5 features	113
Table C.49 - Accuracy of Multi-Layer Perceptron with Grid Search with different number of features	114
Table C.50 - <i>Parameter Space</i>	114
Table C.51 - Best Parameters found from <i>Parameter Space</i> with <i>Grid Search</i>	114
Table C.52 - Accuracy of Multi-Layer Perceptron with Grid Search best model with the best parameters.....	114
Table C.53 - F1-Score, Accuracy, Specificity, Sensitivity of Multi-Layer Perceptron with Grid Search of Final Multi-Layer Perceptron with Grid Search model with all features.....	115
Table C.54 - Accuracy of K-Nearest Neighbor with different number of features	116
Table C.55 - Accuracy of K-Nearest Neighbor with different <i>N Neighbors</i> parameter.	116
Table C.56 - Accuracy of K-Nearest Neighbor with different <i>Weights</i> parameter	116
Table C.57 - Accuracy of K-Nearest Neighbor best model with the best parameters.	116
Table C.58 - F1-Score, Accuracy, Specificity, Sensitivity of K-Nearest Neighbor of Final K-Nearest Neighbor model with 9 features	117
Table C.59 - Accuracy of Ensemble Bagging Classifier with different number of features.	118
Table C.60 - Accuracy of Ensemble Bagging Classifier with different <i>Base Estimator</i> parameter	118
Table C.61 - Accuracy of Ensemble Bagging Classifier with different <i>Max Samples</i> parameter	119
Table C.62 - Accuracy of Ensemble Bagging Classifier with different parameters.....	119
Table C.63 - Accuracy of Ensemble Bagging Classifier with different <i>Random State</i> parameter	119
Table C.64 - Accuracy of Ensemble Bagging Classifier best model with the best parameters	120
Table C.65 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Bagging Classifier of Final Ensemble Bagging Classifier model with 9 features	120
Table C.66 - Accuracy of Ensemble Adaboost Classifier with different number of features	121
Table C.67 - Accuracy of Ensemble Adaboost Classifier with different <i>Base Estimator</i> parameter	121

Table C.68 - Accuracy of Ensemble Adaboost Classifier with different <i>Learning Rate</i> parameter	122
Table C.69 - Accuracy of Ensemble Adaboost Classifier with different <i>Algorithm</i> parameter	123
Table C.70 - Accuracy of Ensemble Adaboost Classifier best model with the best parameters	123
Table C.71 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Adaboost Classifier of Final Ensemble Adaboost Classifier model with all features.....	123
Table C.72 - Accuracy of Ensemble Voting Classifier with different number of features.	124
Table C.73 - Accuracy of Ensemble Voting Classifier with different <i>Weights</i> parameter.	124
Table C.74 - Accuracy of Ensemble Voting Classifier best model with the best parameters	125
Table C.75 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Voting Classifier of Final Ensemble Voting Classifier model with all features.....	125
Table C.76 - Accuracy of Ensemble Gradient Boosting Classifier with different number of features.....	126
Table C.77 - Accuracy of Ensemble Gradient Boosting Classifier with different <i>Min Impurity Decrease</i> parameter	126
Table C.78 - Accuracy of Ensemble Gradient Boosting Classifier with different <i>Loss</i> parameter	126
Table C.79 - Accuracy of Ensemble Gradient Boosting Classifier with different <i>Learning Rate</i> parameter	126
Table C.80 - Accuracy of Ensemble Gradient Boosting Classifier with different <i>Subsample</i> parameter	127
Table C.81 - Accuracy of Ensemble Gradient Boosting Classifier with different <i>Criterion</i> parameter	127
Table C.82 - Accuracy of Ensemble Gradient Boosting Classifier with different <i>Max Depth</i> parameter	127
Table C.83 - Accuracy of Ensemble Gradient Boosting Classifier with different <i>Min Samples Leaf</i> parameter	128
Table C.84 - Accuracy of Ensemble Gradient Boosting Classifier best model with the best parameters.....	129
Table C.85 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Gradient Boosting Classifier of Final Ensemble Gradient Boosting Classifier model with 9 features.....	129

Table C.86 - Accuracy of Ensemble Stacking Classifier best model with the best parameters	130
Table C.87 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Stacking Classifier of Final Ensemble Stacking Classifier model.....	130
Table C.88 - Accuracy of Ensemble Extreme Gradient Boosting Classifier with Grid Search with different number of features.....	131
Table C.89 - <i>Parameter Space</i>	131
Table C.90 - Best Parameters found from <i>Parameter Space</i> with <i>Grid Search</i>	131
Table C.91 - Accuracy of Ensemble Extreme Gradient Boosting Classifier with Grid Search best model with the best parameters	131
Table C.92 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Extreme Gradient Boosting Classifier with Grid Search of Final Ensemble Extreme Gradient Boosting Classifier with Grid Search model with 9 features	132
Table C.93 - Accuracy of TPOT with different number of features (default parameters).	133
Table C.94 - Accuracy of TPOT with 20 <i>Generations</i> parameter	133
Table C.95 - Accuracy of TPOT with 50 <i>Generations</i> parameter	133
Table C.96 - Accuracy of TPOT with 100 <i>Generations</i> parameter	133
Table C.97 - Accuracy of TPOT with different <i>Population Size</i> parameter	133
Table C.98 - Accuracy of TPOT with different <i>Mutation Rate</i> parameter	134
Table C.99 - Accuracy of TPOT with different <i>Crossover Rate</i> parameter	134
Table C.100 - Accuracy of TPOT with different <i>cv</i> parameter	134
Table C.101 - Accuracy of TPOT with different <i>Random State</i> parameter.....	135
Table C.102 - F1-Score, Accuracy, Specificity, Sensitivity of TPOT of Final TPOT Classifier model	135
Table C.103 - Accuracy of all models	136
Table D.1 - Accuracy of Naïve Bayes with different numbers of features	139
Table D.2 - F1-Score, Accuracy, Specificity, Sensitivity of Naïve Bayes of Final Naïve Bayes model with 5 features.....	139
Table D.3 - Accuracy of Logistic Regression with different numbers of features	140
Table D.4 - Accuracy of Logistic Regression with different parameters	140
Table D.5 - F1-Score, Accuracy, Specificity, Sensitivity of Logistic Regression of Final Logistic Regression model with 5 features	140
Table D.6 - Accuracy of Decision Tree with different number of features	141
Table D.7 - Accuracy of Decision Tree with different <i>Criterion</i> parameter	141

Table D.8 - Accuracy of Decision Tree with different <i>Splitter</i> parameter	141
Table D.9 - Accuracy of Decision Tree with different <i>Max Depth</i> parameter	141
Table D.10 - Accuracy of Decision Tree with different <i>Min Samples Split</i> parameter .	142
Table D.11 - Accuracy of Decision Tree with different <i>Min Samples Leaf</i> parameter .	142
Table D.12 - Accuracy of Decision Tree with different <i>Min Weight Fraction Leaf</i> parameter	142
Table D.13 - Accuracy of Decision Tree with different <i>Min Impurity Decrease</i> parameter.	142
Table D.14 - Accuracy of Decision Tree best model with the best parameters	143
Table D.15 - F1-Score, Accuracy, Specificity, Sensitivity of Decision Tree of Final Decision Tree model with 9 features.....	143
Table D.16 - Accuracy of Random Forest with different number of features.....	144
Table D.17 - Accuracy of Random Forest with different <i>N Estimators</i> parameter	144
Table D.18 - Accuracy of Random Forest with different <i>Criterion</i> parameter	144
Table D.19 - Accuracy of Random Forest with different <i>Max Depth</i> parameter	144
Table D.20 - Accuracy of Random Forest with different <i>Min Samples Leaf</i> parameter.	145
Table D.21 - Accuracy of Random Forest with different <i>Bootstrap</i> parameter	145
Table D.22 - Accuracy of Random Forest with different <i>Max Samples</i> parameter.....	145
Table D.23 - Accuracy of Random Forest best model with the best parameters	145
Table D.24 - F1-Score, Accuracy, Specificity, Sensitivity of Random Forest of Final Random Forest model with all features.....	145
Table D.25 - Accuracy of Neural Network with different number of features.....	147
Table D.26 - Accuracy of Neural Network best model with the best parameters	147
Table D.27 - F1-Score, Accuracy, Specificity, Sensitivity of Neural Network of Final Neural Network model with 5 features.....	147
Table D.28 - Accuracy of Support Vector Machine with different number of features.	148
Table D.29 - Accuracy of Support Vector Machine best model with the best parameters.	148
Table D.30 - F1-Score, Accuracy, Specificity, Sensitivity of Support Vector Machine of Final Support Vector Machine model with 9 features	148
Table D.31 - Accuracy of Multi-Layer Perceptron with Grid Search with different number of features.....	149
Table D.32 - <i>Parameter Space</i>	149
Table D.33 - Best Parameters found from <i>Parameter Space</i> with <i>Grid Search</i>	149

Table D.34 - Accuracy of Multi-Layer Perceptron with Grid Search best model with the best parameters.....	149
Table D.35 - F1-Score, Accuracy, Specificity, Sensitivity of Multi-Layer Perceptron with Grid Search of Final Multi-Layer Perceptron with Grid Search model with 5 features	150
Table D.36 - Accuracy of K-Nearest Neighbor with different number of features.....	151
Table D.37 - Accuracy of K-Nearest Neighbor best model with the best parameters.	151
Table D.38 - F1-Score, Accuracy, Specificity, Sensitivity of K-Nearest Neighbor of Final K-Nearest Neighbor model with all features.....	151
Table D.39 - Accuracy of Ensemble Bagging Classifier with different number of features.	152
Table D.40 - Accuracy of Ensemble Bagging Classifier best model with the best parameters	152
Table D.41 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Bagging Classifier of Final Ensemble Bagging Classifier model with all features.....	152
Table D.42 - Accuracy of Ensemble Adaboost Classifier with different number of features	153
Table D.43 - Accuracy of Ensemble Adaboost Classifier best model with the best parameters	153
Table D.44 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Adaboost Classifier of Final Ensemble Adaboost Classifier model.....	153
Table D.45 - Accuracy of Ensemble Voting Classifier with different number of features.	154
Table D.46 - Accuracy of Ensemble Voting Classifier best model with the best parameters	154
Table D.47 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Voting Classifier of Final Ensemble Voting Classifier model with 9 features	154
Table D.48 - Accuracy of Ensemble Gradient Boosting Classifier with different number of features.....	156
Table D.49 - Accuracy of Ensemble Gradient Boosting Classifier best model with the best parameters.....	156
Table D.50 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Gradient Boosting Classifier of Final Ensemble Gradient Boosting Classifier model with all features	156
Table D.51 - Accuracy of Ensemble Stacking Classifier best model with the best parameters	157

Table D.52 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Stacking Classifier of Final Ensemble Stacking Classifier model with 9 features.....	157
Table D.53 - Accuracy of Ensemble Extreme Gradient Boosting Classifier with Grid Search with different number of features.....	158
Table D.54 - <i>Parameter Space</i>	158
Table D.55 - Best Parameters found from <i>Parameter Space</i> with <i>Grid Search</i>	158
Table D.56 - Accuracy of Ensemble Extreme Gradient Boosting Classifier with Grid Search best model with the best parameters	158
Table D.57 - F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Extreme Gradient Boosting Classifier with Grid Search of Final Ensemble Extreme Gradient Boosting Classifier with Grid Search model with all features	159
Table D.58 - Accuracy of TPOT with different number of features (default parameters).	160
Table D.59 - Accuracy of TPOT with different <i>Population Size</i> parameter	160
Table D.60 - Accuracy of TPOT with different <i>Mutation Rate</i> parameter.....	160
Table D.61 - Accuracy of TPOT with different <i>Crossover Rate</i> parameter	160
Table D.62 - Accuracy of TPOT with different <i>cv</i> parameter.....	161
Table D.63 - Accuracy of TPOT with different <i>Random State</i> parameter	161
Table D.64 - F1-Score, Accuracy, Specificity, Sensitivity of TPOT of Final TPOT Classifier model with all features	162
Table D.65 - Accuracy of all models	163
Table E.1 - MSE and R^2 of Linear Regression with different numbers of features.....	166
Table E.2 - MAE, MSE, RMSE, R^2 and MAPE of Linear Regression of Final model with 5 features	166
Table E.3 - MSE and R^2 of Gaussian Process Regression with different numbers of features	167
Table E.4 - MAE, MSE, RMSE, R^2 and MAPE of Gaussian Process Regression of Final model with 5 features.....	167
Table E.5 - MSE and R^2 of Decision Tree with different numbers of features	168
Table E.6 - MSE and R^2 of Decision Tree best model with the best parameters.....	168
Table E.7 - MAE, MSE, RMSE, R^2 and MAPE of Decision Tree of Final model with 5 features	168
Table E.8 - MSE and R^2 of Random Forest with different numbers of features.....	169
Table E.9 - MSE and R^2 of Random Forest best model with the best parameters.....	169
Table E.10 - MAE, MSE, RMSE, R^2 and MAPE of Random Forest of Final model with 5 features	169

Table E.11 - MSE and R^2 of Elastic Net with different numbers of features	170
Table E.12 - MSE and R^2 of Elastic Net best model with the best parameters	170
Table E.13 - MAE, MSE, RMSE, R^2 and MAPE of Elastic Net of Final model with all features	170
Table E.14 - MSE and R^2 of Ridge Regression with different numbers of features.....	171
Table E.15 - MSE and R^2 of Ridge Regression best model with the best parameters..	171
Table E.16 - MAE, MSE, RMSE, R^2 and MAPE of Ridge Regression of Final model with 5 features	171
Table E.17 - MSE and R^2 of Bayesian Ridge Regression with different numbers of features	172
Table E.18 - MSE and R^2 of Bayesian Ridge Regression best model with the best parameters	172
Table E.19 - MAE, MSE, RMSE, R^2 and MAPE of Bayesian Ridge Regression of Final model with all features	172
Table E.20 - MSE and R^2 of Neural Network with different numbers of features.....	173
Table E.21 - MSE and R^2 of Neural Network best model with the best parameters....	173
Table E.22 - MAE, MSE, RMSE, R^2 and MAPE of Neural Network of Final model with all features	173
Table E.23 - MSE and R^2 of Multi-Layer Perceptron with Grid Search with different numbers of features.....	174
Table E.24 - <i>Parameter Space</i>	174
Table E.25 - Best Parameters found from <i>Parameter Space</i> with <i>Grid Search</i>	174
Table E.26 - MSE and R^2 of Multi-Layer Perceptron with Grid Search best model with the best parameters.....	175
Table E.27 - MAE, MSE, RMSE, R^2 and MAPE of Multi-Layer Perceptron with Grid Search of Final model with 3 features	175
Table E.28 - MSE and R^2 of Support Vector Machine with different numbers of features.	176
Table E.29 - MSE and R^2 of Support Vector Machine best model with the best parameters	176
Table E.30 - MAE, MSE, RMSE, R^2 and MAPE of Support Vector Machine of Final model with 3 features.....	176
Table E.31 - MSE and R^2 of K-Nearest Neighbor with different numbers of features.	177
Table E.32 - MSE and R^2 of K-Nearest Neighbor best model with the best parameters.	177

Table E.33 - MAE, MSE, RMSE, R^2 and MAPE of K-Nearest Neighbor of Final model with 3 features.....	177
Table E.34 - MSE and R^2 of Ensemble Bagging Regressor with different numbers of features	178
Table E.35 - MSE and R^2 of Ensemble Bagging Regressor best model with the best parameter	178
Table E.36 - MAE, MSE, RMSE, R^2 and MAPE of Ensemble Bagging Regressor of Final model with 3 features.....	178
Table E.37 - MSE and R^2 of Ensemble Adaboost Regressor with different numbers of features	179
Table E.38 - MSE and R^2 of Ensemble Adaboost Regressor best model with the best parameter	179
Table E.39 - MAE, MSE, RMSE, R^2 and MAPE of Ensemble Adaboost Regressor of Final model with 3 features.....	179
Table E.40 - MSE and R^2 of Ensemble Gradient Boosting Regressor with different numbers of features.....	180
Table E.41 - MSE and R^2 of Ensemble Gradient Boosting Regressor best model with the best parameter	180
Table E.42 - MAE, MSE, RMSE, R^2 and MAPE of Ensemble Gradient Boosting Regressor of Final model with 3 features.....	180
Table E.43 - MSE and R^2 of Ensemble Voting Regressor with different numbers of features	181
Table E.44 - MSE and R^2 of Ensemble Voting Regressor best model with the best parameter	181
Table E.45 - MAE, MSE, RMSE, R^2 and MAPE of Ensemble Voting Regressor of Final model with 3 features.....	182
Table E.46 - MSE and R^2 of Ensemble Stacking Regressor best model with the best parameter	183
Table E.47 - MAE, MSE, RMSE, R^2 and MAPE of Ensemble Stacking Regressor of Final model	183
Table E.48 - MSE and R^2 of Ensemble Extreme Gradient Boosting Regressor with Grid Search with different numbers of features	184
Table E.49 - <i>Parameter Space</i>	184
Table E.50 - Best Parameters found from <i>Parameter Space</i> with <i>Grid Search</i>	184
Table E.51 - MSE and R^2 of Ensemble Extreme Gradient Boosting Regressor with Grid Search best model with the best parameters	184

Table E.52 - MAE, MSE, RMSE, R^2 and MAPE of Ensemble Extreme Gradient Boosting Regressor with Grid Search of Final model with 3 features	185
Table E.53 - MSE and R^2 of TPOT with different numbers of features.....	186
Table E.54 - MAE, MSE, RMSE, R^2 and MAPE of TPOT of Final model with 3 features	186
Table E.55 - MSE and R^2 of all models.....	187

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
AUC	Area Under the ROC Curve
BC	Breast Cancer
CRISP-DM	Cross-Industry Standard Process for Data Mining
DT	Decision Tree
FI	Feature Importance
FNA	Fine-needle aspirates
GP	Genetic Programming
IQR	Interquartile Range
KNN	K-Nearest Neighbor
LiR	Linear Regression
LR	Logistic Regression
ML	Machine Learning
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
MLP	Multi-layer Perceptron
MSM	Multi-surface Method
MSM-T	Multi-surface Method – Tree
NB	Naïve Bayes
PNN	Probabilistic Neural Network
R²	R-Squared
RF	Random Forest
ROC	Receiver Operating Characteristic
RMSE	Root Mean Squared Error

SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
WDBC	Wisconsin Diagnostic Breast Cancer
WPBC	Wisconsin Prognostic Breast Cancer

1. INTRODUCTION

1.1. CONTEXT, PROBLEM, AND MOTIVATION

Breast Cancer (BC) is, undoubtedly, considered the most common cancer in females worldwide and has been recognized as a disease for thousands of years. However, the understanding of its causes and treatments has evolved over time.

Breast Cancer and its History

BC is a disease arising from the uncontrolled growth of cells in the breast. This disease can present different forms that depend on the type of breast cells that transform into cancer. The breast is made up of 3 main parts: lobules, ducts, and connective tissue. Lobules are glands responsible for milk production. Ducts are tubes responsible for transporting milk to the nipple. Lastly, connective tissue – consisting of fibrous and adipose tissue – is responsible for engaging and holding all constituencies together. Hence, BC can originate from different constituent sections of the breast, with its most often beginning either in the ducts or in the lobules. Until the 19th century, BC was uncommon as a result of the lack of improved sanitation and control of deadly infectious diseases that caused women to die too young to develop BC. It was from the 19th century onward that these conditions fundamental to life improved, and consequently, with the help of advances in general surgery technology, surgical procedures aimed at removing breast tumors began to be performed. In the 20th century, more advanced treatments such as radiotherapy and chemotherapy became available and, the first mammography screening programs were created aiming at the early detection of cancer. It should be noted that despite all these advances, there is still a long way to go as BC remains a significant health problem.

Machine Learning and Genetic Programming for Breast Cancer

Machine Learning (ML), a subset of artificial intelligence (AI), can be described as a field of computer science dedicated to understanding and creating methods that enable machines to have the ability to learn autonomously and improve performance through experience. Genetic Programming (GP) can be described as a type of Evolutionary Algorithm, a subset of ML. It is inspired by the principles of natural selection and genetics and mimics natural selection in order to find an optimal result.

Despite all the evolution in medicine and the fact that it has brought fundamental advances in the diagnosis and prognosis of BC, it is possible to go further and ally the realm of medicine with AI so that both cooperate and solve problems in a more optimized and faster way. So, connecting ML and/or GP with BC can help improve early detection, personalized treatment, diagnostic accuracy, and prediction outcomes, leading to better patient outcomes and a

reduction in the overall burden of BC. In other words, the implementation of ML and GP algorithms in order to enhance the diagnosis and prognosis in BC care.

The process that guided the advancement of my research began with the identification and definition of the primary problem: the uncertainty associated with BC diagnosis and prognosis, coupled with the limited availability of diverse diagnostic and prognostic algorithms for accurate results. This challenge served as the impetus for my thesis. Consequently, my research pursued the objective of applying ML and GP techniques to improve the accuracy and efficiency of BC diagnosis and prognosis. In terms of diagnosis, the primary problem to address is distinguishing between malignant and benign breast cytology. This distinction is crucial as it directly impacts treatment decisions and patient outcomes. Regarding prognosis, my research focused on two main problems: firstly, distinguishing between recurrent and non-recurrent BC, and secondly, predicting the recurrence time for recurrent cases as well as the disease-free time for non-recurrent cases. Accurate prognostic is essential for tailoring personalized treatment plans and improving long-term patient management. In this way, my research aims to address the central question of my thesis: **How can ML and GP be used effectively to enhance the diagnosis and prognosis of BC?**

By developing and refining algorithms through ML and GP, the goal is to create robust tools that can handle the complexity and variability inherent in BC cases. These tools are expected to provide more reliable diagnostic classifications and prognostic predictions.

1.2. OBJECTIVES

To develop and analyze algorithms aimed at improving the diagnosis and prognosis of BC, two datasets from the BC Wisconsin Dataset, which contains detailed information on the features of nuclear cells present in breast mass, were used. Additionally, a comprehensive plan was formulated, outlining several steps to achieve the proposed objective and overcome identified limitations while addressing the research question.

The first devised step involved conducting an in-depth literature review. The principal focus of this initial step was to conduct an extensive scientific literature exploration, aggregation, analysis, and review to gather insights on current methodologies, challenges, and advancements in the field of BC diagnosis and prognosis. Beyond that, identifying gaps and opportunities where ML and GP can be effectively applied is also an objective. It is expected that this detailed analysis of these studies will be crucial to contribute significantly to the enrichment of the practical part of the work, allowing a more solid approach based on scientific evidence. The second step focused on developing methodologies encompassing data understanding and preprocessing, model development, and the evaluation and validation of employed methods. The aim is to determine and identify the most reliable approaches for cooperation with the diagnosis and prognosis of BC. The primary purpose is to conduct the

investigation and derive meaningful insights to improve BC diagnosis and prognosis using ML and GP techniques. In this case, it is expected to analyze the datasets to understand the distribution and characteristics of the features. Perform necessary preprocessing steps such as data cleaning, normalization, and feature selection to prepare the data for model development. Additionally, implement ML and GP algorithms tailored to address the specific challenges of distinguishing between malignant and benign breast cytology, as well as predicting recurrence and disease-free intervals. Lastly, it is expected a rigorous test to the developed models using appropriate metrics to evaluate their performance to ensure their robustness and reliability. Finally, the process will culminate with a discussion and a detailed presentation of the final results. It is expected to compare the performance of the developed models as well as highlight the findings, highlighting the algorithms that improve the accuracy and reliability of BC diagnosis and prognosis.

In conclusion, despite these advances, BC remains a significant health problem, and ongoing research is focused on developing better treatments and strategies for diagnosis and prognosis. Therefore, by following these steps, the research aims to create robust and precise diagnostic and prognostic tools that can significantly enhance the current capabilities of BC care.

1.3. DOCUMENT STRUCTURE

This report is divided into the following chapters:

- **Chapter 2** is focused on a methodical exploration, aggregation, analysis, and review of scientific literature, including research papers and thesis related to BC diagnosis and prognosis.

- **Chapter 3** contains not only the precise procedures, techniques, and tools employed to achieve the research objectives but also outlines the approach taken to conduct the investigation and derive meaningful insights to improve BC diagnosis and prognosis using ML and GP techniques.

- **Chapter 4** is dedicated to analyzing and extracting valuable information from the implementation of ML and GP techniques, a deeper understanding of the factors and parameters influencing the diagnosis and prognosis of the disease was attained.

- **Chapter 5** consists of an analysis of all research conducted, more concretely, the actual work and research completed. It also outlines potential future work, limitations, and difficulties faced, and finally, some thoughts on this work.

2. LITERATURE REVIEW

To a better understanding of this project, an initial step encompassed a methodical exploration, aggregation, analysis, and review of scientific literature, including research papers and thesis related to breast cancer (BC) diagnosis and prognosis. The main objective was to identify and scrutinize promising methodologies in this domain. Hence, this section provides an in-depth examination of research endeavors delving into the application and deployment of machine learning (ML) and genetic programming (GP) techniques within the realm of BC. These studies have contributed not only to improving the diagnosis of the disease but also the prognosis, thereby paving the way for potential improvements in overall outcomes. Consequently, the understanding and study of these innovative approaches served as a reference for the implementation and development of effective algorithms and models in the practical part of the work. Therefore, the detailed analysis of these studies was crucial and contributed significantly to the enrichment of the practical part of the work, allowing a more solid approach based on scientific evidence. To conclude this introductory part, I would like to point out that, after exhaustive searching, gathering, and reviewing available papers, I systematically assessed the selected sources to discern prevalent themes, strengths, and weaknesses inherent in each. Subsequently, I delineated the structure of the work, and I carried out the literature review that is presented here. So, it is possible to classify the relevant literature that has been reviewed into 4 distinct groups: literature related to the diagnosis, prognosis, both, and the survival of BC patients.

2.1. EMPIRICAL WORK ON THE DIAGNOSIS

Within this section, all the significant papers related to the diagnosis are presented, and both older and more recent publications were reviewed throughout the project. The literature has therefore been organized and presented chronologically and a brief and concise introduction has been given for each one.

The review began with a paper that was written in 1993 (Street et al., 1993) and the principal objective of this investigation is to develop a highly accurate system for the diagnosis of breast tumors using interactive image processing methodologies along with an inductive classifier grounded in linear programming. Given the intricate nature of this disease, timely identification assumes paramount importance in facilitating efficacious therapeutic interventions. This research leverages a dataset comprising images sourced from the University of Wisconsin Hospitals, serving as the foundation for both training and testing the proposed model. The methodology initiates with the extraction of a small drop of fluid sample from a breast tumor using a fine needle, followed by subsequent digital analysis of the obtained image. Employing a computer vision diagnostic system, 10 distinct features (radius, perimeter, and others) are extracted from the cell nuclei boundaries generated by the snake-

generated algorithm. Afterwards, the mean value, largest value, and standard error of each feature are computed for each image. This process results in the incorporation of a total of 30 different features within the proposed model. In this problem, the classification procedure used was a variant of the Multi-surface Method (MSM), known as MSM-Tree (MSM-T). To enhance the generalization of the generated classifier to unseen cases, the objective was not only to minimize the number of separating planes but also the number of features used. For this reason, the resultant single-plane classifier discriminates the points based on 3 feature values: mean texture, along with extreme values of area, and smoothness. The derived plane separates 97.3% of the cases successfully. Following that, 10-fold cross-validation was performed, yielding a predicted correctness rate of 97.0%. Lastly, the sensitivity and specificity for this model were calculated. The obtained results demonstrated a high accuracy of the used model in classifying breast cytology types as benign or malignant. Nonetheless, this study revealed certain limitations, notably the exhaustive search within the space of potential feature subset. This process becomes impractical for larger subsets, posing a challenge to the scalability and efficiency of the methodology. So, in conclusion, in this study, (Street et al., 1993) describes a system that uses image processing and ML techniques to diagnose breast tumors by non-invasive fine needle aspiration. Furthermore, they highlight the potential of using these types of algorithms in BC classification.

Over the past few decades, empirical work has repeatedly demonstrated the remarkable evolution of artificial intelligence (AI) and its widespread applications, particularly evident in the medical sphere. In the work conducted by (Wolberg et al., 1994), the main objective of this research is to establish an exceedingly accurate diagnostic system for breast tumors by employing interactive image-processing techniques along with an inductive classifier based on linear programming. Essentially, the authors integrate digital image analysis coupled with ML methodologies to improve the diagnostic accuracy of breast fine needle aspirates. While sharing similarities with the first article mentioned, this paper distinguishes itself as a more comprehensive iteration, representing a continuous and evolving research endeavor. Here, beyond the consecutive series of 569 patients provided to the data to develop the system, 54 new consecutive patients were added to test the system. The accuracy of these new samples was 100.0%.

A year later, (Wolberg, Street, Heisey, et al., 1995), in the paper entitled *Computer-Derived Nuclear Features Distinguish Malignant from Benign Breast Cytology*, employ ML techniques for early detection and prediction of BC diagnosis, analyzing computer-generated nuclear features to distinguish between benign and malignant breast cytology. The study uses a set of images extracted from the University of Wisconsin Hospitals to develop a BC diagnostic model using fine needle aspiration samples from 569 patients. Before applying ML algorithms, they used the Pearson correlation coefficient to check the correlation between the different features. In consequence, strong correlations were identified between *Worst Radius* and *Worst Area* (0.983), and between the *Worst Texture* and *Mean Texture* (0.912). On the other hand, the correlation between *Worst Concave Points* and *Worst Smoothness* was weaker

(0.546). Following that, Logistic Regression (LR), and inductive ML algorithms (MSM-T) were applied and achieved cross-validated accuracies of 96.2% and 97.5%, respectively. LR only applied *standard error radius*, *worst radius*, *worst texture*, and *worst concave points*, leading to the misclassification of 9 benign and 12 malignant cells in 10-fold cross-validation repeated 100 times. Conversely, with MSM-T classification, the most accurate 3-feature, single-plane classification separation was achieved using *Worst area*, *Mean texture*, and *Worst smoothness*. The MSM-T model, employed with cross-validation, resulted in the misclassification of 7 benign and 7 malignant fine needle aspiration cells. Sensitivity and specificity metrics were used to evaluate classifier effectiveness. The study highlights the capacity of ML to identify complex patterns for improved BC diagnosis and treatment.

In 2010, with this paper *Machine Learning techniques to Diagnose Breast Cancer* (Osareh & Shadgar, 2010) introduces a study that seeks to develop a highly accurate system for BC diagnosis using ML techniques. Given the complexity of the disease, the emphasis is on early detection, recognizing its crucial role in effective treatment. This study used 2 publicly available reference datasets - one containing images of breast lesions, and the other with images of gene microarrays. The objective was to develop a predictive model for BC diagnosis and, to achieve this, different ML algorithms, including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Probabilistic Neural Network (PNN), were explored. Therefore, in order to distinguish between benign and malignant breast tumors, the study combined the aforementioned classifiers with signal-to-noise ratio feature ranking, sequential forward selection-based selection, and principal component analysis feature extraction. The obtained results indicate that the overall best accuracy for BC diagnosis reached 98.80% and 96.33%, respectively, using SVM classification models against two widely used reference datasets for BC.

Years later, in 2016, (Shubair, 2016) delves deeper and provides a comparative analysis of three prominent ML techniques - SVM, Random Forest (RF), and Bayesian Networks - for BC detection. The original Wisconsin BC Dataset is employed as the basis for the evaluation. After rigorous evaluation using accuracy, recall, precision, and Receiver Operating Characteristic (ROC) curve metric with 10-fold cross-validation, the results reveal that SVM exhibits the highest performance in terms of accuracy, specificity, and precision. However, RF demonstrates the highest probability of correctly classifying tumors. Specifically, SVM achieves an accuracy of 97.0%, while RF attains 99.9% in terms of area under ROC curve (AUC) values.

Breast Cancer Predictions using Machine Learning Algorithm published in 2017, employs the Wisconsin BC dataset to evaluate BC prediction using four ML algorithms. Principal component analysis is initially employed to reduce dimensionality, followed by the application of KNN, SVM with a linear kernel, LR, and Naïve Bayes (NB). The results indicate that KNN, SVM, and LR consistently outperform NB with comparable accuracy levels. KNN achieves the highest average accuracy, approximately, 97.56% after applying 10-fold cross-validation.

Furthermore, the SVM with a linear kernel achieves the highest Area Under the ROC Curve (AUC) value of, approximately 99.44%. In this study, (Yu, 2017) points out that an augmentation in the number of top eigenvectors contributes to enhanced prediction accuracy. Nevertheless, the findings suggest that beyond a certain threshold, the increase introduces more noise than signal, indicating a potential trade-off between the number of eigenvectors and prediction precision.

Two years subsequently, the paper titled *Machine Learning Classification Techniques for Breast Cancer Diagnosis* analyzes the Wisconsin BC dataset using dimensionality reduction techniques and three ML algorithms - SVM, Artificial Neural Networks, and NB - for classifying malignant and benign tumors. In addition to combining these ML techniques with dimensionality reduction methods, the paper compares their performances to identify the most effective approach. For this hybrid approach, linear discriminant analysis is applied to reduce the high dimensionality of features, and, subsequently, SVM is applied in the new reduced feature dataset. This approach achieves an accuracy of 98.82%, sensitivity of 98.41%, specificity of 99.07%, and AUC characteristic curve of 99.94%. Hence, the authors (Omondiagbe et al., 2019) end up concluding and demonstrating that incorporating feature selection and feature extraction enhances the accuracy of benign and malignant tumor diagnosis through ML techniques. Furthermore, future work is discussed, and more ML algorithms are an option to consider for comparison in BC diagnosis.

The central aim of Donga's 2022 thesis – *Comparing Machine Learning Models: For Diagnosis of Breast Cancer* - is to compare and evaluate the performance of supervised ML algorithms on the Wisconsin BC dataset. LR, SVM, RF Classification (RF), and Decision Tree (DT) were specifically applied for BC diagnosis. Initially, the dataset underwent collection, and subsequent preprocessing techniques were implemented. This involved creating a heatmap using a correlation matrix for feature selection. Following this, label encoding was implemented to convert category feature values into numerical representations, and, subsequently, the train-test-split approach was employed to divide the original dataset into 80:20 train and test data, which were manually separated for training ML algorithms. The efficiency of each algorithm was determined by comparing the accuracy scores obtained during the evaluation process. The accuracy scores obtained from both datasets – train and test dataset – were then compared to identify the most efficient algorithm. Consequently, the SVM approach stands out as the most effective algorithm for diagnosing BC, surpassing other ML algorithms with 100.0% accuracy on the training data and 98.0% accuracy on the testing data. In future work, the author (Donga, 2022), intends to expand their application of ML algorithms and deploy these models in hospitals. The primary objective is to enhance the efficiency of predicting results rapidly and at a lower cost.

The research, conducted by (Hammi, 2022) and using the Wisconsin BC dataset, undertakes a comparative analysis of diverse ML algorithms, including LR, RF, Gradient Boosting, AdaBoost, and Bagging Classifier. Preceding the application of these algorithms, the dataset undergoes

thorough cleaning, reshaping, and wrangling. Subsequently of these preparations, multiple feature selection methods are employed, and the data is partitioned into training and testing sets with a split ratio of 70:30. The efficacy and performance of the ML techniques are evaluated through metrics such as accuracy, recall, precision, and f1-score. The empirical results reveal that LR demonstrates the highest accuracy, reaching 98.8%, coupled with the lowest number of false negatives. This highlights the efficacy of LR as a robust classification algorithm for the given dataset.

The most recent research reviewed endeavors to provide a comprehensive overview of the application of ML in BC detection. The author systematically evaluates the performance of different ML algorithms using the Wisconsin BC dataset. First, the investigation initiates with meticulous data preprocessing involving data balancing and feature standardization. Successively, LR, RF, and Extreme Gradient Boosting Regressor algorithms are trained and tested on this diagnostic dataset. Performance metrics including accuracy, f1-score, sensitivity, and specificity are employed for evaluation. The outcomes underscore the superior performance of LR, attaining the highest accuracy at 98.32%. Additionally, LR demonstrates an f1-score of 98.0%, indicating a robust balance between precision and recall. Notably, the model exhibits a precision score of 100.0%, reflecting its ability to minimize false positives. The recall score of 96.0% further emphasizes the model's proficiency in identifying true positives. These findings underscore LR as a highly effective algorithm for BC detection within the context of the provided dataset. The author (Kottur, 2023) emphasizes the importance of conducting more extensive studies to evaluate how the models perform across a variety of datasets. Moreover, the results suggest the potential applicability of ML models in forecasting the probability of survival for individuals with BC.

2.2. EMPIRICAL WORK ON THE PROGNOSIS

Within this section, all noteworthy papers related to the prognosis are presented, encompassing a review of both older and more recent publications conducted throughout the project. In this context, only one paper was reviewed, and a succinct introduction to it is provided here.

In the present investigation undertaken by (Wolberg et al., 1995), the main purpose is to conduct empirical research related to BC prognosis. This involves a dual focus: firstly, to evaluate the prognostic importance of computer-derived nuclear features through ML; secondly, to employ ML techniques to prognosis based on the identified and selected features. The authors endeavor to develop a highly accurate system for the prognosis of breast tumors by incorporating interactive image-processing techniques alongside an inductive ML algorithm. Therefore, the authors compare and evaluate the prognostic importance of a variety of computer-derived nuclear features employing both statistical analysis and inductive ML algorithms. This involves a detailed statistical analysis of the correlation between these

computer-generated nuclear features and the occurrence or absence of distant metastases within 24 months of diagnosis, specifically within a subset of 146 patients from the overall study population. Delving into further detail, the process of obtaining these cell samples took place within the diagnostic procedures for 187 consecutive patients diagnosed with invasive BC. Within this specific group, a distinct subset was delineated, consisting of 118 patients who did not manifest distant recurrence within 24 months, alongside 28 patients who did experience recurrence. Furthermore, 41 were excluded from the analysis once they had not been followed for a minimum of 24 months. First and foremost, in the course of this study, a T-test analysis was executed on a subset of 147 patients to scrutinize the significance of features concerning distant disease recurrence within a 2-year timeframe. The results indicate that the number of cancer-containing axillary lymph nodes and tumor size were not significant compared to computer-derived nuclear features. In this situation, the computer-derived features, including mean Radius, mean Perimeter, mean Area, mean Fractal dimension, worst Radius, worst Perimeter, standard error of Area, worst Area, and worst Fractal dimension proved to be more robust predictors of 2-year distant recurrence compared to tumor size and lymph node status. Regarding the model applied for the selection of the most useful features for prognostic models, the inductive ML technique developed by the authors was Recurrence Surface Approximation. This ML model, validated through ten-fold cross-validation, consistently selected computer-derived nuclear features 97.8% of the time. In contrast, tumor size was chosen only 1.7% of the time, and lymph node status a mere 0.5% of the time. On the other hand, the MSM-T was used for prognosis, yielding an accuracy rate of 86.0%, as reported by the authors. Hence, the authors draw attention to the fact that computer-generated nuclear features, particularly size features are more important than tumor size, and lymph node statuses.

2.3. EMPIRICAL WORK ON BOTH: DIAGNOSIS AND PROGNOSIS

Within this section, an overview of papers focused on the diagnosis and prognosis is provided. These papers used the Wisconsin Diagnostic Breast Cancer (WDBC) and Wisconsin Prognostic Breast Cancer (WPBC) datasets. To maintain a chronological progression, the approach begins with a review of earlier literature and concludes with an exploration of more recent studies.

(Mangasarian et al., 1970) carried out a study with the primary aim of developing a precise system for detecting BC diagnosis and prognosis, employing ML techniques. For this purpose, two medical applications of linear programming are described in this paper, specifically, linear programming-based ML techniques are used to bolster BC diagnosis and prognosis accuracy and objectivity. The research uses a dataset comprising images extracted from the University of Wisconsin hospitals to both train and test the model. Consequently, the paper can be divided into two spare parts: the diagnostic segment and the prognostic segment. In the diagnostic part, the principal goal of the investigation was to develop a relatively objective system capable of diagnosing fine needle aspirations with an accuracy that rivals the best

visually achieved results. A training set consisting of 569 patients was used to construct a classifier enabling the diagnosis of future examples. For this purpose, the MSM-T classification model was employed and, it was concluded that better generalization could be achieved by reducing the number of input features considered. Different combinations that resulted in classifiers that effectively separated the training set, were evaluated using cross-validation to estimate their accuracy. Particularly, the best results were achieved by 1 plane and 3 features: *Extreme Area*, *Extreme Smoothness*, and *Mean Texture*, achieving a predicted accuracy of 97.5% as estimated through cross-validation. On the other hand, within the prognostic segment, attention is directed toward the long-term prognosis of the disease for patients whose cancer has been surgically removed. The Recurrence Surface Approximation method was employed to ascertain a linear combination of the input features that approximate the time to recurrence. In this case, RSA procedure was tested with leave-one-out testing to evaluate its accuracy in predicting future outcomes. Of the 569 patients, 187 malignant cases with follow-up data were used. Subsequently, the authors implemented 3 different prediction methods, and the mean generalization errors of the RSA were compared with the Pooled RSA and Least 1-norm Error on Recurs. The results highlight that the original RSA formulation demonstrates comparable performance in both recurrent and non-recurrent cases. Conversely, the pooled error method greatly favors the majority non-recurrent class, thereby reducing the mean error. However, since the main goal is to accurately predict all cases, the original formulation is deemed superior. Nonetheless, the recurrent results for the least 1-norm estimation are optimal, while this method performs least favorably on the non-recurrent cases and yields the highest overall mean error. Concluding, the authors ended up developing 2 linear-programming-based classification methods that significantly helped both the diagnosis and the prognosis of the disease. Additionally, their findings show that the most effective predictive models solely use nuclear features, without improvement from traditional medical prognostic factors such as tumor size and lymph node status. This suggests that the routine and potentially hazardous practice of removing lymph nodes from the armpit of BC patients for prognostic purposes can be circumvented.

The *Image Analysis and Machine Learning Applied to Breast Cancer Diagnosis and Prognosis* paper elucidates the effectiveness of a computer system for the diagnosis and prognosis of BC, facilitated by the analysis of computer-extracted features and ML methodologies based on linear programming. In other words, (Wolberg, Street, & Mangasarian, 1995) described a system that uses computer vision techniques to analyze the size, shape, and texture features of cell nuclei, subsequently employing an inductive approach rooted in linear programming for classification purposes. For this analysis, images were extracted from the Hospital of the University of Wisconsin. The key components of the research encompassed diagnostic investigation, prognostic evaluation, and an assessment of reproducibility. In the diagnostic segment, the principal objective was to develop an objective system for diagnosing fine needle aspirations with accuracy comparable to visual assessment. This involved training a classifier using the MSM-T method on a dataset containing 569 patients. 10-fold cross-validation revealed that improved generalization could be achieved through feature reduction.

Remarkably, a combination of one plane and 3 features, produced a predicted accuracy of 97.5%. Regarding the prognostic part, the authors focused on assessing long-term BC prognosis. This implied the use of the same MSM-T separation algorithm on a dataset comprising 166 patients. Consequently, the analysis centered on a 2-year breakpoint and the best 4 nuclear features with 2 separating planes. The resulting Kaplan-Meier curve showed the probability of distant disease-free survival. In this scenario, due to the smaller sample size, the one-leave-out was employed and, the overall accuracy was estimated at 86.0%, with a 95.0% confidence region. Finally, reproducibility was assessed by analyzing a subset of images in duplicate and triplicate, with disparities ranging from 1% to 10% across different parameters. To evaluate the performance of each classifier and its effectiveness, the overall accuracy, sensitivity, and specificity were calculated. In conclusion, the authors have illustrated the practical use of linear programming in diagnosing and prognosing BC. Empirical results show that in addition to perimeter being the most important individual feature for both diagnosis and prognosis, nuclear features are stronger predictors of recurrence compared to traditional prognostic features such as axillary lymph node status and tumor size.

Continuing from prior research conducted by the same authors, (Wolberg, Street, Heisey, et al., 1995) have authored this paper. Here, they describe an interactive computer system for evaluating, diagnosing, and determining prognosis based on cytologic features, using digital image analysis and ML. In this way, digital image analysis was used in order not only to improve breast mass diagnosis based on fine needle aspirates but also to improve BC prognostic estimations. Images were sourced from the Hospitals of the University of Wisconsin for training and testing the model, with 569 consecutive patients in the training dataset (212 with malignant masses and 357 with benign masses) and 118 consecutive new patients in the testing dataset (31 with malignant masses and 97 with benign masses). Additionally, 190 patients with invasive cancer and without distant metastases were used for prognosis. For diagnosis, the MSM-T classification procedure was employed, followed by 10-fold cross-validation to estimate the performance on unseen cases. In this analysis, worst area, mean texture, and worst smoothness yielded the most accurate three features, single plane classification separation, achieving a predicted diagnostic accuracy of 97.0%. The actual diagnostic accuracy on 118 new samples was 100.0%. Regarding prognosis, mean errors were calculated between the actual times of distance disease occurrence and predicted times using various prognostic features. Comparing the actual time for recurrence, the results indicate that the mean error of predicted times for recurrence with the nuclear features was 17.9 months and 20.1 months with tumor size and lymph node status ($p=0.11$). To conclude, the research demonstrates the diagnostic significance of computer-generated nuclear features and highlights the prognostic importance of nuclear size features over traditional features like tumor size and lymph node status. The paper presents and mentions other related studies to substantiate the proposed approach and the findings obtained.

In 2017, (Sharma et al., 2017) present a study wherein they utilize the WDBC dataset for the purpose of diagnosing BC, and the WPBC dataset to estimate the likelihood of recurrence in

affected patients. The methodological framework commences with data preprocessing, progresses through visualization, model selection, and implementation, and culminates with evaluation. Data preprocessing involves min-max normalization to mitigate the effects of noise, errors, outliers, and inconsistencies. Subsequently, the dataset is split into training and test subsets in an 80:20 ratio. Sophisticated classifiers such as LR, Nearest Neighbor (KNN), and SVM are afterward applied. To gauge the model performance, the authors employ metrics such as accuracy, sensitivity, specificity, and the AUC. The findings reveal that LR achieves the highest accuracy of 96.89% for diagnostic purposes, whereas SVM demonstrates superior performance for prognosis with an accuracy of 89.73%. In conclusion, the authors emphasize the importance of rigorous evaluation of these techniques before commercial application, particularly in fields like biomedical sciences, which demand utmost accuracy and reliability.

In the ensuing year, (Yue et al., 2018) endeavor to conduct a review of ML techniques and their applications in BC diagnosis and prognosis. The authors initiate their study by providing a comprehensive overview of different ML techniques, encompassing artificial neural network (ANN), SVM, DT, and KNN, elucidating the distinct characteristics and mechanisms of each approach. Subsequently, they delve into the exploration of these techniques' applications in the context of BC, utilizing the benchmark dataset WDBC. The results demonstrate that ML techniques can enhance classification and prediction accuracy. In their research, the authors present various methods, along with references, algorithms, sampling strategies, and classification accuracies, providing clear and intuitive information. Besides this, through their literature review, they emphasize that while many algorithms have achieved high accuracy in WBCD, there is still a need for the development of improved algorithms. Additionally, they note that while classification accuracy is crucial, it is not the only assessment criterion. Moreover, despite ANN traditionally dominating BC diagnosis and prognosis for several decades, the authors observe and point out that, recently, alternative ML methods have been increasingly applied in intelligent healthcare systems, offering a diverse range of options to physicians. Finally, the authors outline their future work, expressing their intention to conduct an in-depth analysis of rare datasets, such as the one obtained from Friedreich Ataxia (FRDA) patients. Furthermore, they plan to explore the application and testing of various novel Particle Swarm Optimization (PSO) algorithms in conjunction with ANN by tuning weights. They, effectively, aim to determine whether higher classification accuracy can be achieved with WBCD data.

2.4. EMPIRICAL WORK ON SURVIVAL OF BREAST CANCER PATIENTS

Within this section, papers focusing on the survival of BC patients are discussed. Therefore, herein, a synopsis of the literature, the datasets employed, the algorithms employed, and the corresponding model classification measures are presented. It is noteworthy that two of these papers propose solutions employing GP.

To date, there exists a conspicuous absence within the literature wherein GP has not been leveraged to effectively diagnose, prognosticate, or classify cancer patients into distinct risk classes. (Vanneschi et al., 2011) presents a pioneering investigation in which the use of several ML algorithms is invested. Diverging from conventional methodologies, the authors use the NKI BC dataset to predict the survival outcomes of BC patients through the application of a 70-gene signature. Furthermore, they determine the most efficacious algorithm for this purpose, thereby introducing a novel perspective by integrating GP into the analysis of this gene signature. In terms of methodology, the authors conduct the collection and preprocessing of the dataset encompassing relevant clinical and demographic characteristics, alongside survival data concerning BC patients. For this case, they use NKI BC dataset, which provides gene expression and survival data for 295 consecutive breast carcinoma patients. Subsequently, in order to make a comparative analysis between models, they employ different ML algorithms, including SVM, RF, Multi-layer Perceptrons (MLP), and GP. Consequently, for the prediction of survival among BC patients included in the NKI dataset, they apply the models mentioned above to the problem. Empirical findings advocate that, despite the predictive capabilities demonstrated by all ML algorithms employed, GP notably outperforms its counterparts in effectively classifying BC patients into distinct risk categories. Additionally, the investigation delved further into the role of feature selection and performance on unbalanced datasets. Regarding feature selection, the study initially identifies the ten features most frequently selected by GP from the initial set of seventy features. Both GP and SVM were then evaluated using these features, revealing the intrinsic value of GP's feature selection. Moreover, the study suggests the potential of a recursive GP approach for feature optimization. Finally, it was concluded that dataset balancing is not imperative for achieving satisfactory performance with GP. When predicting the survival of BC patients using gene signatures, the main goal is to minimize false negatives. GP consistently outperformed other methods, as confirmed by statistical analysis. In order to prioritize false negatives, the authors modified the GP fitness function yielded a slight improvement, albeit not statistically significant. However, GP demonstrated superior performance compared to SVM, MLP, and RF.

(Bozorgi, 2018), undertakes a dissertation aiming to explore the prediction of five-year survivability among BC patients through the use of ML methodologies. Distinguished from prior studies, this work diverges in terms of the dataset used and addresses the challenges posed by the unbalanced distribution of data between minority and majority classes, along with the specific data cleaning procedures employed. Taking advantage of the experience gained with the Surveillance, Epidemiology, and End Results (SEER) database, the study focuses on predicting survivability. NB, DT, Linear (LiR) and LR algorithms were applied to this dataset with the purpose of enhancing the recall levels for the minority class. A final conclusion drawn from the results indicates that LR, with appropriate predictors and class weight adjustments, gives the highest precision/recall level for the minority class.

The study, proposed by (Rabiei et al., 2022), focuses on predicting BC using different machine approaches, incorporating demographic, laboratory, and mammographic data. For this goal, the database utilized originates from the Motamed Cancer Institute (ACECR) in Iran. The research methodology encompasses 4 steps: data collection, data pre-processing, modeling and model evaluation. Initially, data collection involved gathering records from the aforementioned dataset, consisting of 5178 records and 24 features, including 11 demographic, 9 laboratory, and 4 mammographic. Each entry has a result that can be identified as the presence or absence of cancer. Afterward, the Synthetic Minority Oversampling Technique (SMOTE) was applied to balance the training data, and then, the authors employed various ML models, namely RF, MLP neural network, Gradient Boosting Trees, and Genetic Algorithms. Performance evaluation of the classifiers was conducted using metrics such as accuracy, sensitivity, specificity, and the AUC. Notably, the RF model presented the highest sensitivity of 95% and achieved an accuracy rate of 80%.

2.5. CONCLUSION

Over the past few decades, empirical work has repeatedly demonstrated the remarkable evolution of AI and its widespread applications, particularly evident in the medical sphere. Nevertheless, as highlighted by (Sharma et al., 2017), (Naji et al., 2021), (Naseem et al., 2022), and (Mohammed et al., 2020), the existing approaches used to diagnose and prognose BC largely rely on traditional, simple ML algorithms. On the other hand, despite having a lot of literature on diagnosing cancer, for prognosis, there is a scarcity of research focusing on its relevance (Ferroni et al., 2019). Furthermore, (Yue et al., 2018) points out a critical research gap in integrating GP techniques with ML in order to develop more versatile and effective models for BC diagnosis and prognosis. Additionally, to these limits, papers tend to prioritize accuracy, neglecting other performance parameters such as precision, recall, and AUC (Nassif et al., 2022). Finally, despite the considerable literature available on diagnosis, there is a limited diversity of ML algorithms applied, alongside a notable lack of reliable literature on the implementation of GP algorithms. Thus, this dissertation seeks to introduce an innovative solution that surpasses the limitations of prior research, ultimately enhancing BC diagnosis and prognosis.

3. METHODOLOGY

After conducting an in-depth literature review to contextualize the application and implementation of machine learning (ML) and genetic programming (GP) techniques within the domain of breast cancer (BC) research, a meticulously robust methodology emerged. This encompassed not only the precise procedures, techniques, and tools employed to achieve the research objectives but also outlined the approach taken to conduct the investigation and derive meaningful insights to improve BC diagnosis and prognosis using ML and GP techniques. To achieve the main goal of the thesis, the methodology applied in this project was based on the Cross-Industry Standard Process for Data Mining (CRISP-DM) and, therefore, it is structured into 2 primary domains: diagnostic methodology and prognostic methodology.

3.1. CROSS – INDUSTRY STANDARD PROCESS FOR DATA MINING

The CRISP-DM framework was rigorously employed to systematically guide the methodological process, as depicted in ANNEX 1. This framework epitomizes a structured approach, facilitating methodical and iterative progression toward attaining the research objectives while providing an overview of the data mining life cycle. Consequently, to conduct the methodology research for both diagnosis and prognosis, I adhered to the CRISP-DM framework for each respective part. In a general sense, the following actions were pursued:

1. **Business Understanding:** The underlying perspective driving the thesis theme was thoroughly investigated and explored. This initial phase encompasses defining research objectives and comprehending the specific requirements and constraints associated with enhancing BC diagnosis and prognosis. Through the establishment of clear goals, this phase forms the basis for subsequent methodological decisions.
2. **Data Understanding:** A meticulous identification, collection, and examination of the available data for mining was undertaken. This step was crucial to preempt unexpected issues during data preparation and to ascertain data quality. Employing data exploration techniques, it was possible to acquire a comprehensive understanding of the inherent characteristics and challenges of the data.
3. **Data Preparation:** This critical phase involves all activities aimed at transforming the initial raw data into the final dataset for further processing. Rigorous preprocessing steps, such as data cleaning, normalization, and feature extraction, were performed to enhance the quality of the data and prepare it for model development. Effective data preparation is essential for optimizing the performance of algorithms in subsequent stages.
4. **Modeling:** The analysis methodologies, congruent with the objectives defined in the initial stage are carefully selected and implemented. Employing these methodologies, the model is built using modeling techniques that facilitate the extraction of

information from the available data, thereby enabling a response to the objectives of the project.

5. **Evaluation:** The models underwent rigorous testing and evaluation to determine their alignment with the objectives of the current study. The developed models are evaluated using standard performance metrics.
6. **Deployment:** This is the phase where the developed models are deployed into operational use within clinical settings. However, this step is excluded from this work.

3.2. DIAGNOSTIC METHODOLOGY

3.2.1. Business Understanding

In this part of the work, a comprehensive description and analysis of the undertaken tasks will be presented, structured, and delineated according to the CRISP-DM methodology. In theoretical terms, BC diagnosis involves the process of determining whether an individual has BC. As previously mentioned, ML and GP techniques were employed to diagnose cancer from fine-needle aspirates (FNA). Consequently, the principal challenge and objective within this context centered on effectively distinguishing between malignant and benign breast cytology using an optimized model, thereby addressing a classification problem. After conducting a comprehensive review of all relevant literature, the implementation phase of the study was commenced. To address the challenge at hand, the **Wisconsin Diagnostic Breast Cancer (WDBC)** dataset was used. This dataset was donated by Nick Street in 1995 and presents:

1. **569** instances, featuring a binary class distribution: **357** instances classified as benign, and **212** instances classified as malignant.
2. **32** features (ID, diagnosis, 30 real-valued input features). These features are computed based on a digitized image of a FNA of a breast mass, depicting characteristics of the cell nuclei. Hence, the attribute information is presented below:
 - **ID number**,
 - **Diagnosis** (M = malignant, B = benign)
 - **(3 – 32) – Ten real-valued features** are computed for each cell nucleus. Hence, the information is delineated in Table A.1.

The **mean**, **standard error**, and "**worst**" or largest (which refers to the mean of the three largest values) of these features were computed for each image, yielding a total of 30 features. For instance: **Field 3** is the Mean Radius, **Field 13** is the Radius Standard Error, **Field 23** is the Worst Radius.

3.2.2. Data Understanding

In this section, a detailed examination of the available data will be conducted, including exploratory analysis and discussions of visualizations. These endeavors aim to foster a thorough and dependable comprehension of the dataset, enabling deeper exploration, insights, and the detection of potential irregularities.

3.2.2.1. Exploratory Analysis

In the initial stage, the problem was carefully examined to determine its purpose. Subsequently, before proceeding with the development of the models, an extensive exploration and analysis of the dataset variables were conducted. Therefore, the project commenced with the importation of the WDBC dataset, followed by an initial analysis of data variables. After this preliminary analysis, to enhance efficiency, alignment, filtering, and data integrity, the *ID* was set as the index of the dataset. A tuple representing the dataset's dimensionality was then returned, indicating 569 inputs and 31 features. Continuing with this initial analysis, the distinct values for each feature were enumerated, and the comprehensive descriptive statistics of all features in the dataset were computed and scrutinized. The results of these analytical procedures are visually represented in Figure A.1 and Figure A.2. From Figure A.1, it is evident that the *diagnosis* feature comprises only two unique values, whereas the remaining features exhibit a higher number of unique values. Figure A.2 provides a detailed exploration and analysis of various statistical metrics such as count, mean, standard deviation, minimum value, maximum value, lower and upper percentiles, and median for numerical data. For object data, it provides the count, number of unique values, top value, and frequency of occurrence for each object feature. In this context, *diagnosis* is the only object variable, and its top is *Benign* result with a frequency of 357. Subsequently, the data was grouped by *diagnosis*, revealing the total number of entries corresponding to each diagnosis category and 357 entries match a *Benign diagnosis*, and 212 entries match a *Malignant diagnosis*, which is approximately 62.7% and 37.3% of the data, respectively. Afterward, the data types in each column were verified to explore potential transformations. As illustrated in Figure A.3, while *diagnosis* is identified as a categorical variable, the remaining features are characterized as floats. Regarding duplicated and missing values, this dataset has neither. Subsequently, a more detailed examination of the variables was conducted.

3.2.2.2. Metric and Non-metric Features

In ML, the nature of features influences the types of analyses that can be performed on the data. Specifically, certain methods are used for analyzing metric features, while others are suited for non-metric features. Metric features refer to variables with numerical values, measured on a continuous scale. Conversely, non-metric features denote variables without

numerical values, used to describe qualities or characteristics. As previously confirmed, this dataset exclusively includes numerical features. In this way, a list was generated containing all numeric features, i.e., metric features. The *diagnosis* variable serves as the target value and is thus excluded from this list.

3.2.2.3. Data Visualization

There are various methods to visually represent the data, facilitating exploration, insights, and anomaly detection. These representations include histograms, barplots, displots, pairplots, and boxplots. Firstly, an initial histogram – Figure A.4 - was constructed aiming to delineate the distribution of individuals with benign and malignant BC. Secondly, for numerical variables, histograms, and distribution plots were generated to examine the distribution of each feature. Histograms provided insights into the shape and spread of the data, enabling comparison of the frequency of each value. On the other hand, distribution plots offered a comprehensive overview of dataset distribution encompassing shape, spread, central tendency, and outliers. Figure A.5 presents a theoretical example of these examples, focusing on *radius_se*. It is important to emphasize that this analysis was conducted for all features, although only one example is included in the report for brevity. In this figure, a right-skewed distribution is evident, indicating that most of the data clusters around the left side of the distribution, with a few values extending further to the right. Consequently, it can be concluded that this variable does not follow a normal distribution. Thirdly, to understand the relationship between numerical variables, pairplot visualizations were created. Specifically, three pairplots were generated: one for all mean features, another for all standard error features, and the last one for all worst features. Figure A.6 showcases the pairwise relationship among the mean of numerical variables. From this graph, it is evident that there are direct proportional correlations between *radius_mean*, *area_mean*, and *perimeter_mean*, as well as between *perimeter_mean* and *area_mean*. This pattern remains consistent in the pairplot of worst variables. In contrast, while the relationships are less distinct in the pairplot for standard error variables, there is still evidence of several correlations between the variables. Finally, boxplots for numeric variables were generated to observe the outliers in each variable. Figure A.7 displays these boxplots, revealing that most variables include outliers. Notably, according to the figure, *concave_points_worst* appears to be the only one without any outliers.

3.2.3. Data Preprocessing

In this section, data preparation was conducted to gather, clean, and transform the raw data into processed data suitable for analysis. Herein, the behavior of the variables, including their distribution and incoherencies, will be described and explored. Additionally, 3 new variables were created and investigated to enrich the future model. Data cleaning, transformation,

reduction, and splitting were performed, encompassing outlier removal, standardization, balancing unbalanced data, and feature selection.

3.2.3.1. New Variables

For this research, 3 new variables were created to enrich the model and assess their potential in providing additional and valuable information. Utilizing the variables *radius_mean*, *radius_se*, and *radius_worst*, the volume was computed, resulting in the creation of the following variables: *mean_volume*, *se_volume*, and *worst_volume*. These 3 float variables were generated to facilitate a 3-dimensional evaluation of the cells.

3.2.3.2. Target Split

To ensure that the data utilized for training and testing a ML algorithm accurately represents the overall distribution, the technique of splitting the target variable was employed. Thus, the target variable comprised the *diagnosis* column, while the data variable consisted of all features except the *diagnosis* feature. Through this technique, the objective was to mitigate biases, enhance model performance, validate the model, and get a deeper understanding of the data.

3.2.3.3. Standardization

Regarding feature scaling, a robust scaler was employed due to its effectiveness in handling/dealing with outliers. This scaler uses robust statistics methods, including removing the median and scaling the data based on the interquartile range (IQR), thereby minimizing sensitivity to extreme values. This approach is advantageous for enhancing model performance by mitigating the influence of outliers. As a result, all data was standardized to maintain consistent scaling.

3.2.3.4. Outlier Removal

For outlier detection and removal, 3 methods were applied to the metric features. Initially, the IQR method was applied, but it was observed that this approach would eliminate over 30% of the data, suggesting a potential loss of valuable information. Indeed, the results obtained by this methodology may not necessarily indicate the presence of outliers; rather, they signify data points lying beyond the limits defined by the IQR. Consequently, a more detailed analysis is warranted to comprehensively assess these values and ascertain their significance within the dataset. Subsequently, the second approach involved manual removal of the most

extreme values using histograms and boxplots, resulting in the elimination of only 1.23% of the data. In the end, the Tukey method was employed as the third option for outlier identification and removal. This algorithm uses the IQR to identify potential outliers and subsequently tests them for statistical significance using the Tukey HSD test. With this method, only 1.23% of the data was eliminated, corresponding to the removal of 7 entries. Notably, these 7 entries were identified as outliers in over 12 features. Before proceeding, an examination was conducted to understand the nature of the removed entries, revealing that all 7 entries were classified as malignant.

3.2.3.5. Imbalanced Data

Before assessing and examining data imbalance, the categorical target variable underwent encoding, with B representing 0 and M representing 1 . Following this encoding, visualizations were created to verify if data was significantly imbalanced. Figure A.8 provides an overview of the observed imbalance, and it is observed that data is not balanced, prompting the application of strategies for improvement. The primary aim was to adapt the training set to align with standard learning algorithms. To address the imbalance, distributions were balanced using oversampling techniques. The simplest approach involves duplicating random records from minority class. For this purpose, the Synthetic Minority Oversampling Technique (SMOTE) was applied. Following the implementation of this technique, Figure A.9 demonstrates the achievement of a balanced dataset.

3.2.3.6. Feature Selection

After completing the data preparation phase, I proceeded with feature selection, dividing this process into 3 parts to determine the importance of features and select the most relevant variables for the final method. In this way, I will present and discuss **Filter Methods**, followed by **Wrapper Methods**, and, finally, **Embedded Methods**.

Filter Methods

I initiated the feature selection process by employing filter methods, which are techniques independent of any ML algorithm, assessing feature relevance based on their statistical properties. These methods evaluate the intrinsic characteristics of features and their relationship to the target variable. In this way, the filter methods used were:

- **Pearson Correlation:** Evaluates the linear association between two variables, quantifying the strength and direction of the linear relationship between them. The threshold used was $|0.70|$, resulting in several correlated features being identified.

For instance, *radius_mean* and *perimeter_mean* exhibit a correlation of 1.0, indicating a perfect positive linear relationship between them.

- **Spearman Correlation:** Evaluates the rank correlation association between two variables, indicating the strength and direction of the monotonic relationship between them, regardless of linearity. The threshold used was $|0.70|$, resulting in correlated features being identified. For instance, *perimeter_se* and *area_se* exhibit a correlation of 0.94, indicating a very strong positive monotonic relationship between them.
- **Kendall Correlation:** The threshold used was $|0.70|$, resulting in correlated features being identified. For instance, *perimeter_worst* and *area_mean* exhibit a correlation of 0.86, indicating a very strong positive monotonic relationship between them.
- **Variance Variables:** The variance of each variable was calculated to serve as an effective criterion for selecting important features. The rationale behind this approach is that features with higher variance tend to demonstrate more variability across the dataset, making them potentially more informative for models. In this case, a threshold of values greater than 0.6 was used to identify important features, with *se_volume* standing out due to its notably higher variance.
- **ANOVA F-values:** This calculates the F-statistic values for each feature and ranking them based on their contribution to the classification task. The top 10 features were selected and *radius_mean* emerged as an important variable classified by ANOVA.
- **Principal Component Analysis:** Employed to reduce the dimensionality of the dataset, with the objective of retaining the top three principal components to capture the highest variance in the data. *Area_se* emerged as one of the most significant variables represented by the first principal component.

Wrapper Methods

Wrapper methods entail selecting features by assessing the performance of a specific ML algorithm on different subsets of features. This process involved a search mechanism, where different feature combinations are evaluated based on their impact on the performance of the model. In this way, the wrapper method used was:

- **Recursive Feature Elimination:** This method iteratively removes less important features based on their contribution to the model performance, thereby identifying the subset of features that maximizes model performance. In this case, the logistic regressor is the estimator. In the end, the method determines the optimum number of features, which was found to be 7.

Embedded Methods

Thirdly, the feature selection process involved employing embedded methods, which are techniques that integrate feature selection directly into the model training process. With

these methods, feature selection becomes an integral of the learning algorithm's optimization procedure. In this way, the embedded methods used were:

- **Lasso Cross-Validation (LassoCV):** LassoCV is a variant of Lasso regression that incorporates cross-validation to automatically select the regularization parameter that optimizes the performance of the model. In this case, the dataset is split into 10 folds for cross-validation. Notably, LassoCV picked 23 features which included *worst_volume* and *area_worst* are the most important features identified.
- **Elastic Net Cross-Validation (ElasticNetCV):** This is a variant of Elastic Net Regression that incorporates cross-validation to automatically select the optimal values for the hyperparameters alpha and l1_ratio. In this case, the dataset is split into 5 folds for cross-validation. Notably, ElasticNetCV picked 23 features where *worst_volume*, *area_worst*, and *radius_worst* are the most important features identified.
- **Gini and Entropy Importance:** Feature importance (FI) scores were obtained from gini impurity and entropy to rank the importance of features in the decision tree (DT) classifier. Gini impurity assesses how frequently a randomly selected element from the dataset would be mislabeled if it was randomly labeled based on the distribution of labels in the dataset. On the other end, entropy quantifies the uncertainty or disorder within a dataset. In both cases, *perimeter_worst* emerged, indeed, the most important feature.
- **CatBoost:** CatBoost Classifier with Root Mean Squared Error (RMSE) was applied as the evaluation metric for feature selection. Catboost is adept at handling categorical features and belongs to the class of gradient boosting which combines multiple weak learners to generate a robust model. In this instance, the threshold was set to retain the 15 most important features identified by this method. Remarkably, *perimeter_worst* and *concave_points_worst* emerged as the most significant features.
- **Three-Based Classifiers:** Employing the ExtraTreesClassifier, FI is calculated during the training of the ensemble of DT. These importance scores reflect the contribution of each feature to the performance of the model. In this instance, the threshold was set to retain the 15 most important features identified by this method. While there are no significant discrepancies in values, *Perimeter_worst*, *concave_points_worst*, and *area_mean* were the most significant features.
- **AdaBoost Feature Importance:** Assessing FI with Adaboost involves the weights assigned to each feature during the boosting process. These weights are assigned to training samples based on their difficulty in classifying correctly. Higher weights indicate features that consistently lead to misclassified. *Area_se* and *texture_worst* are the variables with the most significance, meaning they have less weight.
- **Permutation Feature Importance:** Assessing FI with permutation involves evaluating the impact of randomly permuting the values of each feature on the performance of the model. In this case, the K Nearest Neighbor model was used and *se_volume* stands out as the most important variable using this approach.

- **Random Forest Feature Importance:** Assessing FI with random forest (RF) FI allows to determine the relative importance of features in a RF model. This provides valuable insights into which features are most informative for the model. In this case, *perimeter_worst* stands out as the most important variable using this approach.

3.2.3.7. Conclusion

After applying all these feature selection methods, in the majority of cases where numerical values were obtained, features were deemed important if the value exceeded or fell below 0.0. If the value equaled 0.0, it was concluded that the variable was not considered important in that particular feature selection model. On the other hand, regarding variance, a threshold was set where values greater than 0.6 were important features. In the end, after applying all methods and gathering the selected features from each model, the results were combined into a dataframe where each feature was represented by either a 1 or a 0. A value of 1 indicated that the feature was considered important by the respective method, while a value of 0 indicated otherwise. By creating and populating this dataframe, it was possible to identify the features that appeared most frequently across the methods and utilize them in subsequent modeling. Figure A.10 illustrates a portion of the final dataframe highlighting the most important features.

3.2.4. Modeling

After meticulously implementing the feature selection methods, the process of selecting the most pertinent variables to incorporate them into models was undertaken. In this stage, the final feature selection was determined, followed by the division of data into training and testing sets, and, ultimately, the application of several models.

3.2.4.1. Final Select Features

Prior to model implementation, a comprehensive analysis was conducted on the features consistently selected by the models in the preceding stage. For this purpose, 4 distinct lists were created: the initial list comprised 3 variables identified as the most important features in over 16 feature selection models; the second list contained the 5 most significant features appearing in more than 15 feature selection models; the third list included 9 features recurrently selected in over 14 models; finally, the last list contained all variables under consideration. These lists were utilized to explore the influence of varying feature sets on model performance and accuracy. Table A.2 clarifies the created lists with different features.

3.2.4.2. Train and Test Split

After the feature selection process, the chosen features were assigned to respective datasets: X3 (comprising 3 features), X5 (with 5 features), X9 (containing 9 features), and XTotal (encompassing all variables). Subsequently, various combinations of features were tested for each model. Following this, the data underwent a split into training and test sets. Separate splits were performed for each dataset (X3, X5, X9, and XTotal), while the target variable (y) remained consistent across all splits. For each split, a test size of 30% was employed.

3.2.4.3. Models

In this subsection, the models used in the research will be outlined. I employed a variety of models to address this problem, beginning with simpler ones and gradually advancing to more complex ones. Firstly, 3 functions were created to help the modeling, in a more efficient way. The first function, displayed in A.11, is designed to execute a model and evaluate its performance based on accuracy. In essence, it serves as a scoring function. Initially, the algorithm is trained (fit), followed by the application of the *predict* method to generate predictions on unseen data using the training data. In the final segment of this function, the primary objective is to meticulously evaluate the performance of the model using the accuracy metric. Accuracy was chosen as the primary evaluation metric for this classification problem because, after employing the SMOTE algorithm, the dataset became balanced. Consequently, accuracy provides a straightforward and intuitive measure of the model's overall performance. In the second function, depicted in A.12, K-Fold cross-validation was employed to avoid and mitigate overfitting and evaluate the performance of the classification model. This technique allows for a more precise and accurate estimation of the model's ability to generalize to unseen data. In this function, K-Fold cross-validation was used as the validation method, and the performance of the model was evaluated using metrics such as F1-score, accuracy, sensitivity, and specificity. These metrics are saved in a dataframe to calculate them for each algorithm. The third function, illustrated in A.13, serves as an auxiliary function designed to calculate the mean and standard deviation of the models' performance metrics score in the score dataframe. Before applying the models, a dataframe was created intended to store the best results obtained from the different models. Afterward, a range of models were implemented, including:

ML Algorithms: *Gaussian Naïve Bayes, Logistic Regression, Decision Trees, Random Forest, Neural Network, Probabilistic Neural Network, Linear Discriminant Analysis, Support Vector Machine, Multi-Layer Perceptron with Grid Search, K-Nearest Neighbors, Ensemble Bagging Classifier, Ensemble Adaboost Classifier, Ensemble Voting Classifier, Ensemble Gradient Boosting Classifier, Ensemble Stacking Classifier, Ensemble Extreme Gradient Boosting Classifier*

GP algorithms: Three-Based Pipeline Optimization Tool (TPOT) was implemented in order to identify the most effective pipeline for addressing the problem.

Gaussian Naïve Bayes

This is a variant of the Naïve Bayes (NB) algorithm and it was the first model to be tested due to its popularity, efficiency, speed, and simplicity in probabilistic classification tasks. The Gaussian Naïve Bayes algorithm assumes that each class follows a normal distribution and makes predictions based on Bayes' Theorem. It models the distribution of each class combination as a normal distribution.

Logistic Regression

It is a statistical method that represents a linear model with a non-linear transformation applied to the output. This algorithm is favored for its simplicity, interpretability, and computational efficiency. However, its effectiveness in capturing complex relationships between variables is limited compared to other algorithms.

Decision Trees

This is a popular tree-like structure where each internal node represents a feature, each subtree represents a decision rule, and each leaf node represents the outcome, in this case, the prediction. DT are known for their ability to handle high-dimensional data and demonstrate good performance even with a small amount of training data. Beyond these particularities, they are useful for providing FI ratings, as used in the feature selection phase. Despite these advantages, DT can be sensitive to small variations in the training data, often resulting in the creation of complex trees.

Random Forest

It represents an ensemble method that combines multiple DT to make predictions. Despite several advantages, RF also possess certain drawbacks. While they excel in handling high-dimensional data, capturing complex relationships, and providing FI rankings (as performed previously), they may require more computational resources and exhibit slower training times.

Neural Networks

Neural Network represents a powerful and versatile ML model inspired by the structure and functioning of the human brain. Comprising interconnected nodes, or neurons, organized in layers, this algorithm boasts several advantages. Alongside its capability to learn complex patterns and discern non-linear relationships in data, Neural Networks can effectively handle large datasets with high-dimensional features. However, despite these strengths, the algorithm comes with its drawbacks, as it can be computationally expensive and requires a substantial amount of labeled data to prevent overfitting.

Probabilistic Neural Networks

Probabilistic Neural Network represents a type of artificial neural network and distinguish themselves from traditional neural networks in both architecture and learning approaches. Rather than learning weights through iterative training, PNNs rely on probability theory. As a result, this algorithm offers several advantages. In addition to its fast-training speed, and simple architecture, this algorithm efficiently handles high-dimensional data. However, despite these advantages, they may suffer challenges related to memory resource requirements.

Linear Discriminant Analysis

Linear Discriminant Analysis is a method used for both dimensionality reduction and classification. Its primary aim is to project input data into a lower-dimensional space to maximize the separation between classes. This technique seeks to identify discriminant functions that can effectively differentiate classes by considering the dispersion within and between classes.

Support Vector Machine Classifier

Support Vector Machine (SVM) is a robust and powerful ML model known for its effectiveness in handling high-dimensional data and its ability to generalize well to unseen data, making it less prone to overfitting compared to some other models. However, this methodology can be computationally intensive, when using large datasets.

Multi-Layer Perceptron with Grid Search

MLP is a type of neural network model that contains multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. It is a powerful model capable of learning complex patterns. In this study, grid search was used to systematically search the hyperparameter space and, identify the optimal combination based on a specified performance metric. Unlike other models where parameter combinations were manually specified, grid search automates this process by exhaustively testing a predefined subset of hyperparameter combinations using cross-validation. This approach aims to determine the most effective parameters for optimizing model performance.

K-Nearest Neighbors

KNN operates as an instance-based learning algorithm, predicting outcomes by identifying the k-closest training examples in the feature space and aggregating their values through a majority vote. Noteworthy for its simplicity, versatility, and intuitive nature, KNN is robust in handling noisy data. However, it is sensitive to the choice of K and imposes computational demands during prediction.

Ensemble Bagging Classifier

Bagging Classifier is a powerful ML model that combines multiple base classifiers using the bootstrap aggregating technique. By combining predictions from multiple base classifiers, this model reduces and mitigates variance, and enhances the performance, stability, and the overall generalization capability of the model.

Ensemble Adaboost Classifier

Adaboost Classifier is a widely used and effective ensemble ML model that combines multiple weak classifiers to create a robust one. It gives more importance to the misclassified samples by iteratively adjusting the weights of the training instances, thereby improving overall prediction accuracy. Despite its advantages, Adaboost Classifier is susceptible to the presence of noisy data and outliers, which can affect its performance.

Ensemble Voting Classifier

Voting Classifier is an ensemble ML model that combines multiple individual classifiers to make predictions through voting strategies. Indeed, this model harnesses the combined knowledge of multiple classifiers to improve accuracy. Essentially, it aggregates predictions from diverse models to achieve superior overall performance compared to using individual models.

Ensemble Gradient Boost Classifier

Similar to Adaboost Classifier, this algorithm is an ensemble model that sequentially combines multiple weak classifiers to create a strong classifier. While Adaboost Classifier focuses on adjusting weights in training instances to emphasize difficult instances, Gradient Boosting is based on the boosting idea, wherein each subsequent weak classifier is trained to correct the errors made by previous weak classifiers, gradually improving the overall prediction accuracy. Despite of being sensitive to outliers like Adaboost Classifier, Gradient Boosting incorporates regularization techniques to mitigate the impact of these values.

Ensemble Stacking Classifier

This ensemble ML model combines multiple individual classifiers into a single high-level classifier to make predictions through the strengths and weaknesses of different classifiers. By selecting base classifiers and training the model appropriately, it is possible to seize complex relationships for successful implementation. In Stacking Classifier, the base classifiers are trained on the dataset to learn to make predictions based on the input features, each producing its own predictions for the target variable. Unlike other ensemble models, Stacking Classifier employs an aggregator to learn how to combine, or stack, the outputs of the base models. This aggregator takes the base model predictions as inputs and learns to generate the final prediction.

Ensemble Extreme Gradient Boosting Classifier

This powerful and popular gradient-boosting algorithm constructs an ensemble of weak prediction models and iteratively enhances their performance by optimizing an objective function. This algorithm uses Gradient Boosting to minimize the loss function by adding new models to rectify errors made by the existing ensemble. With numerous parameters available for customization, it is possible to get a better performance by optimizing the score of this algorithm. In this model, parameters can be set using grid search. In this particular case, grid search was employed to meticulously explore the hyperparameter space, ultimately identifying the optimal combination based on the chosen performance metric.

TPOT

TPOT stands for Tree-based Pipeline Optimization tool. It is an open-source automated ML library that uses GP concepts to optimize ML pipelines by automatically searching for the best one, including data preprocessing, feature selection, model selection and, hyperparameter tuning. In Annex 2, it is possible to evaluate parts of the ML pipeline automated by TPOT. TPOT uses GP to generate the optimized search space, and, for that reason, it is inspired by Darwin's idea of natural selection, consequently, it uses the following properties:

1. **Selection**, in this stage, the fitness function is evaluated at each of the individuals and their values are normalized.
2. **Crossover**, in this stage, the fittest individuals are selected from above and the crossover is performed between them to generate a new population.
3. **Mutation**, in this stage, the individuals generated by the crossover undergo a mutation and some random modifications are made and, after, repeated during some steps or until the best population is obtained.

This algorithm presents several advantages, in addition to improving the performance through automating the pipeline construction and consequently saving time and effort, it presents the results transparently and supports a wide range of ML algorithms. Despite having these advantages, this algorithm has some drawbacks since it requires substantial computational resources and can be time-consuming. Furthermore, although TPOT can automatically select and combine various preprocessing techniques, it may not offer the same level of flexibility and creativity as a feature engineering manual.

To conclude this modeling chapter, the methodology involved an initial exploration employing simpler models across various feature sets, followed by the implementation of more complex models with differing feature combinations. Throughout this process, extensive parameter testing was conducted across all models to optimize their performance. With the exception of the Extreme Gradient Boosting model and the Multi-Layer Perceptron (MLP) with Grid Search, which employed grid search to systematically explore hyperparameter space, all parameters were manually selected and tuned. Therefore, evaluations were performed on various solutions within each model for different feature sets and parameters, facilitating comparisons using metrics across different models. Finally, after implementing all the ML models, TPOT, a GP model, was implemented. Adjustments were made to the parameters and the model was also tested with different numbers of features. Consequently, results were obtained that made it easier to compare different models and provided a solution to the initial problem.

3.2.5. Evaluation

In this chapter, an in-depth overview of the various metrics used in the evaluation of models will be provided. Therefore, the metrics employed to evaluate and validate the models

included the following: F1-Score, Accuracy, Sensitivity, Specificity, and Receiver Operating Characteristic (ROC) curve.

F1-Score

This F1-score metric measures the harmonic mean of precision and recall, providing a balance between these 2 metrics. It proves to be particularly useful in scenarios with imbalanced classes. Within this context, the F1-score indicates the ability of the model to identify accurately both benign and malignant cases.

Accuracy

This accuracy metric quantifies the overall correctness of the predictions of the model. It is calculated as the ratio of correctly classified instances to the total number of predictions made. Accuracy is fundamental for assessing model performance, indicating, in this context, how effectively the model discriminates between malignant and benign instances.

Sensitivity

This metric is the true positive rate, which measures the proportion of actual positives that the model correctly identifies. In this context, sensitivity reflects the ability of the model to correctly identify malignant cases among all the positive cases.

Specificity

This metric is the true negative rate, which measures the proportion of actual negatives that the model correctly identifies. In this context, specificity reflects the ability of the model to correctly identify benign cases among all the positive cases.

ROC curve

The ROC curve illustrates how the true positive rate (sensitivity) changes in relation to the false positive rate across different threshold values. In this case, this metric offers insights into how well the model can distinguish between benign and malign cases across various threshold levels.

3.3. PROGNOSTIC METHODOLOGY

3.3.1. Business Understanding

In this part of the work, a comprehensive description and analysis of the undertaken tasks will be presented, structured, and delineated according to the CRISP-DM methodology. Theoretically speaking, BC prognosis refers to the anticipated disease progression or likely outcome for individuals diagnosed with BC. This is influenced by several factors, including specific tumor characteristics. Similar to Diagnostic endeavors, ML, and GP techniques have been applied to prognosticate cancer from FNA. Consequently, two principal learning challenges emerge: Firstly, the principal objective is to predict the outcome by distinguishing between **recurrent** and **non-recurrent** BC cases, constituting a classification problem. Secondly, the main objective is to predict the **recurrence time** for recurrent cases and the

disease-free interval for non-recurrent cases, presenting a regression problem. After conducting a comprehensive review of all relevant literature, the implementation phase of the study was commenced. To address the challenge at hand, the **Wisconsin Prognostic Breast Cancer** (WPBC) dataset was used. This dataset also donated by Nick Street in 1995 and presents:

1. **198** instances, featuring a binary class distribution: **151** non-recurrent, **47** recurrent.
2. **34** features (ID, outcome, time 30 real-valued input features, tumor size, and lymph node status) - These features are computed based on a digitized image of a FNA of a breast mass, depicting characteristics of the cell nuclei. Hence, the attribute information is presented below:
 - **ID number**
 - **Outcome** (R = Recur, N = Non-recur)
 - **Time** (Recurrence time if for Outcome = R, and Disease-free time if for Outcome = N)
 - **(4 – 32) – Ten real-valued features** are computed for each cell nucleus. Hence, the information is delineated in Table B.1:

The **mean**, **standard error**, and "**worst**" or largest (which refers to the mean of the three largest values) of these features were computed for each image, yielding a total of 30 features. For instance: **Field 4** is the Mean Radius, **Field 14** is the Radius Standard Error, **Field 24** is the Worst Radius.

- Tumor Size (Diameter of the excised tumor in centimeters)
- Lymph Node Status (Number of positive axillary lymph nodes observed at the time of surgery)

3.3.2. Data Understanding

In this section, a detailed examination of the available data will be conducted, including exploratory analysis and discussions of visualizations. These endeavors aim to foster a thorough and dependable comprehension of the dataset, enabling deeper exploration, insights, and the detection of potential irregularities.

3.3.2.1. Exploratory Analysis

In the initial stage, the problem was carefully examined and, before proceeding with the development of the models, an extensive analysis of the dataset variables was conducted. Thus, the project began with the importation of the WPBC dataset, followed by an initial analysis of data variables. After this preliminary analysis, to enhance efficiency, filtering, and

data integrity, the *ID* was set as the index of the dataset. A tuple representing the dataset's dimensionality was then returned, indicating 198 inputs and 34 features. Continuing with this analysis, the distinct and unique values for each feature were enumerated, and the comprehensive descriptive statistics of all features in the dataset were computed and scrutinized. The results of these analytical procedures are visually represented in Figure B.1 and Figure B.2. From Figure B.1, it is evident that the prognostic *outcome* feature comprises only two unique values, whereas the remaining features exhibit a higher number of unique values. Figure B.2 provides a detailed exploration and analysis of various statistical metrics such as count, mean, standard deviation, minimum value, maximum value, lower and upper percentiles, and median for numerical data. For object data, it provides the count, number of unique values, top value, and frequency of occurrence for each object feature. In this context, *outcome* is the only object variable, and its top is a *non-recur* result with a frequency of 151. Subsequently, the data was grouped by *outcome* and then by *time*, revealing both the total number of entries corresponding to each outcome category and the distribution of entries across different time values. Consequently, 151 entries match a *non-recur* prognostic, and 47 entries match a *recur* prognostic, which is approximately 76.3% and 23.7% of the data, respectively. Concerning the *time* variable, characterized by its continuous nature, it presents 95 distinct values. Afterward, the data types in each column were verified to explore potential transformations. As illustrated in Figure B.3, the dataset comprises 2 object variables, 1 integer variable, and the remaining features are floats. Specifically, *Outcome* and *Lymph node status* are identified as categorical variables, *time* as an integer, and the remaining features as floats. Regarding duplicated and missing values, the dataset despite not having duplicate values, presents 4 missing values represented by '?'. To address these missing values, '?' instances were replaced with -1, followed by conversion of the respective column to an integer datatype. Subsequently, a more detailed examination of the variables was conducted.

3.3.2.2. Metric and Non-metric Features

As previously confirmed, this dataset exclusively includes numerical features. In this way, a list was generated containing all numeric features, i.e., metric features. The *Outcome* and *Time* variables serve as the target value in the two initial problems and are excluded from this list.

3.3.2.3. Data Visualization

Firstly, an initial histogram – Figure B.4 - was constructed aiming to delineate the distribution of individuals with non-recur and recur BC. Secondly, for numerical variables, histograms, and distribution plots were generated to examine the distribution of each feature. Figure B.5 presents an example focusing on *radius_se*. It is important to emphasize that this analysis was conducted for all features, although only one example is included in the report for brevity. In this figure, a right-skewed distribution is evident, indicating that most of the data clusters

around the left side of the distribution, with a few values extending further to the right. Consequently, it can be concluded that this variable does not follow a normal distribution. Thirdly, to understand the relationship between numerical variables, pairplot visualizations were created: one for all mean features, another for all standard error features, and the last one for all worst features. Figure B.6 showcases the pairwise relationship among the mean of numerical variables and it is evident that there are direct proportional correlations between *radius_mean*, *area_mean*, and *perimeter_mean*, as well as between *perimeter_mean* and *area_mean*. This pattern remains consistent in the pairplot of worst variables. In contrast, while the relationships are less distinct in the pairplot for standard error variables, there is still evidence of several correlations between the variables. Finally, boxplots for numeric variables were generated to observe the outliers in each variable and Figure B.7 displays these boxplots, revealing that all variables include outliers.

3.3.3. Data Preprocessing

In this section, data preparation was conducted to gather, clean, and transform the raw data into processed data suitable for analysis. Herein, the behavior of the variables, including their distribution and incoherencies, will be described and explored. Additionally, 3 new variables were created and investigated to enrich the future model. Data cleaning, transformation, reduction, and splitting were performed, encompassing outlier removal, standardization, balancing unbalanced data, and feature selection. These procedures mirror those conducted for the diagnostics but yield distinct outcomes.

3.3.3.1. New Variables

By using the variables *radius_mean*, *radius_se*, and *radius_worst*, 3 new variables were created to enrich the model: *mean_volume*, *se_volume*, and *worst_volume*. These 3 float variables were generated to facilitate a 3-dimensional evaluation of the cells.

3.3.3.2. Target Split

To ensure that the data utilized for training and testing a ML algorithm accurately represents the overall distribution, the technique of splitting the target variable was employed. Hence, two distinct target variables are identified, reflecting the presence of two independent problems. Consequently, the target variables consist of the *time* and the *outcome* columns, while the data variable encompasses all features excluding these 2.

3.3.3.3. Standardization

Robust scaler was employed due to its effectiveness in handling/dealing with outliers. This approach aids in improving model performance by reducing the impact of outliers and, as a result, all data underwent standardized to ensure consistent scaling.

3.3.3.4. Outlier Removal

For outlier detection and removal, 3 methods were applied to the metric features. The first method applied was IQR which measures the spread of the middle 50% of a dataset. It is calculated as the difference between the third quartile (Q3) and the first quartile (Q1). Initially, the IQR method was implemented, but it was observed that this approach would eliminate over 45% of the data, suggesting a potential loss of valuable information. Subsequently, the second approach involved manual removal of the most extreme values using histograms and boxplots, resulting in the elimination of only 1.52% of the data. In the end, the Tukey method was employed as the third option for outlier identification and removal. With this method, only 1.52% of the data was eliminated, corresponding to the removal of 3 entries. Notably, these 3 entries were identified as outliers in over 8 features. Before proceeding, an examination was conducted to understand the nature of the removed entries, revealing that all 2 entries were classified as *non-recur* and 1 as *recur*.

3.3.3.5. Imbalanced Data

Before examining data imbalance, the categorical target (*outcome*) variable underwent encoding, with *N* representing 0 and *R* representing 1. Following this, visualizations were created to verify if the data was significantly imbalance and Figure B.8 provides an overview of the observed imbalance and it is observed that data is not balanced, prompting the application of strategies for improvement. To address the imbalance, distributions were balanced using a simple oversampling technique involving duplicating random records from minority class. For this purpose, SMOTE was applied. Following its implementation, Figure B.9 reveals the achievement of a balanced dataset.

3.3.3.6. Feature Selection

I proceeded with feature selection, dividing this process into 3 parts to determine the importance of features and select the most relevant variables for the final method. In this way, I will present and discuss **Filter Methods**, followed by **Wrapper Methods**, and, finally, **Embedded Methods**. As previously indicated, two distinct problems are under development: a regression problem and a classification problem. Despite having different target variables,

the same feature selection methodology was employed for both, with the particularity that, in the development of the classification problem, the *outcome* was used as the target variable, whereas in the development of the feature selection for the regression problem, *time* was designated as the target variable. Hence, within the methodology leading feature selection, the precise approach for each problem will be discriminated and delineated accordingly.

Filter Methods

I initiated the feature selection process by employing filter methods. Given that detailed explanations of each applied model were provided in the diagnostic section, I will refrain from elaborating on them in this section. In this way, the filter methods used were:

- **Pearson|Spearman|Kendall Correlation:** In both classification and regression, the threshold used was $|0.70|$, resulting in *radius_mean* and *perimeter_mean* exhibiting a very strong positive relationship in 3 algorithms.
- **Variance Variables:** A threshold of values greater than 0.6 was used to identify important features, with *se_volume* standing out due to its notably higher variance in both classification and regression problems contexts.
- **ANOVA F-values:** Calculates the F-statistic values for each feature and ranking them based on their contribution to the classification task. It was possible to reduce from 35 to 10 best features in both classification and regression problems contexts.
- **Mutual Information:** Applied to measure the dependence between a feature and the respective target variable. It was only applied to the classification problem.
- **Principal Component Analysis (PCA):** Employed to reduce the dimensionality of the dataset, to retain the top three principal components to capture the highest variance in the data. *Area_se* emerged as one of the most significant variables represented by the first principal component, in both classification and regression problems contexts.

Wrapper Methods

Secondly, wrapper methods were applied, and the methods used were:

- **Recursive Feature Elimination:** Employed in both classification and regression problems and the optimum number of features identified was 9 and 3, respectively.
- **Boruta:** Applied in the regression problem and it operates as an extension of the RF algorithm and identifies relevant features by comparing their importance scores to those of randomly generated features. Higher scores denote higher importance. In this context, the optimum number of features identified was 2.

Embedded Methods

Thirdly, embedded methods were employed, and the methods used were:

- **Lasso Cross-Validation (LassoCV):** In the classification problem, the 24 best features were identified, with *perimeter_se* and *fractal_dimension_mean* being the most important features. Conversely, in the regression problem, LassoCV selected the 5 best features, with *mean_volume* being identified as the most important feature.
- **Elastic Net Cross-Validation (ElasticNetCV):** In the classification problem, ElasticNetCV identified the 19 best features, with *perimeter_se* and *fractal_dimension_mean* being the most important features. Conversely, in the regression problem, ElasticNetCV selected the 14 best features, with *fractal_dimension_worst* and *texture_mean* being identified as the most important features.
- **Gini and Entropy Importance:** Applied in the classification problem where *texture_mean* emerged as the most important feature in both criteria.
- **CatBoost:** Implemented in both the classification and regression problems, with a threshold set to retain the 15 most important features in the classification problem and the 9 most important features in the regression problem, as identified by the method. In the first problem, *lymph_node_status* arose as the most significant feature, whereas in the second, *texture_mean* stood out as the most significant feature.
- **Three-Based Classifiers:** Implemented in both the classification and regression problems, with a threshold set to retain the 15 most important features in the classification problem and the 9 most important features in the regression problem, as identified by this method. While there are no significant discrepancies in values, in the classification problem, *tumor_size* arose as the most significant feature, whereas in the regression problem, *texture_mean* stood out as the most significant feature.
- **AdaBoost Feature Importance:** Applied in both classification and regression problems. In the first problem, *concave_points_se* emerged as the most significant feature, while in the regression problem, *texture_mean* stood out as the most influential.
- **Permutation Feature Importance:** Applied in both classification and regression problems. In the first, *lymph_node_status* arose as the most significant feature, while in the regression problem, *concave_points_worst* arose as the most significant feature.
- **Random Forest Feature Importance:** Applied in both classification and regression problems. In the first, *lymph_node_status* arose as the most significant feature. In the regression problem, a different approach was undertaken and Boruta was used as a feature selector, resulting in only *radius_mean* and *texture_mena* being selected as important features.
- **Linear Regression Feature Importance:** Applied in the regression problem where *radius_mean* emerged as the most important feature.
- **MSE | MAE | FRIEDMAN:** By using these 3 metrics within a DT model, it is feasible to extract the most important features. In the regression problem, while *texture_mean* arose as the most significant feature using *FRIEDMAN* and Mean Squared Error (MSE),

smootheness_worst was identified as the most significant feature using Mean Absolute Error (MAE).

3.3.3.7. Conclusion

In both main problems, after applying all these feature selection methods, in most cases where numerical values were obtained, features were deemed important if the value exceeded or fell below 0.0. If the value equaled 0.0, the variable was not considered important in that particular feature selection model. On the other hand, regarding variance, a threshold was set where values greater than 0.6 were important features. In the end, after applying all methods and gathering the selected features from each model, the results were combined into a dataframe where each feature was represented by either a 1 or a 0. A value of 1 indicated that the feature was considered important by the respective method, while a value of 0 indicated otherwise. By creating this dataframe, it was possible to identify the features that appeared most frequently across the methods and utilize them in subsequent modeling. Figure B.10 and Figure B.11 illustrate a portion of the final dataframe highlighting the most important features both in classification and regression problems, respectively.

3.3.4. Modeling

After feature selection, the most pertinent variables to incorporate into models were selected. In this stage, the final feature selection was determined, followed by the division of data into training and testing sets, and, ultimately, the application of several models. Like the feature selection process, this section will be segmented into classification and regression problems due to variations in the number of entries and target variables.

3.3.4.1. Final Select Features

Prior to model implementation, a complete analysis was conducted on the features consistently selected by the models in the preceding stage. For the **classification problem**, 6 lists were created: the initial list had 3 variables identified as the most important features in over 17 feature selection models; the second list also contained 3 variables, however, despite encompassing the 3 most common variables, *tumor_size* was excluded because it is not a cell characteristic. An interesting aspect to explore is whether breast cellular variables demonstrate superior performance compared to non-cellular variables; the third list contained 4 features and these features are the ones used by (Wolberg et al., 1995); the fourth list contains the 5 most significant features appearing in more than 16 feature selection models; the fifth list included 9 features recurrently selected in over than 15 modes; Finally, the last list contained all variables under consideration. These lists were utilized to explore the influence of varying feature sets on model performance and accuracy and Table B.2 show

these lists detailed. For the **regression problem**, 4 lists were created: the initial list had 3 variables identified as the most important features in over 14 feature selection models; the second list contains the 5 most significant features appearing in more than 13 feature selection models; the third list included 9 features recurrently selected in over 12 modes; Finally, the last list contained all variables under consideration. These lists were utilized to explore the influence of varying feature sets on model performance and accuracy and Table A.3 clarifies the created lists with different features in regression problem.

3.3.4.2. Train and Test Split

After the feature selection process, the chosen features were assigned to respective datasets. In the **classification problem**, the assignment was: X3 (comprising 3 features), X3s (comprising 3 features excluding *Tumor_size*), X4 (with 4 features), X5 (with 5 features), X9 (containing 9 features), and XTotal (encompassing all variables). Following this, the data underwent a split into training and test sets. Separate splits were performed for each dataset (X3, X3s, X4, X5, X9, and XTotal), while the target variable (y) remained consistent across all splits. A test size of 30% was employed for each split. In the **regression problem**, the assignment was: X3 (comprising 3 features), X5 (with 5 features), X9 (containing 9 features), and XTotal (encompassing all variables). Following this, the data underwent a split into training and test sets. Separate splits were performed for each dataset (X3, X5, X9, and XTotal), while the target variable (y) remained consistent across all splits. A test size of 20% was employed for each split.

3.3.4.3. Models

In this subsection, the models used in the 2 researchs will be outlined. I employed a variety of models to address these 2 problems, beginning with simpler ones and gradually advancing to more complex ones. Firstly, in the classification problem, the 3 functions used in the diagnostic phase were also employed to aid the modeling process. Secondly, in the regression problem, also 3 functions were created to help the modeling, in a more efficient way - Figure B.12, Figure B.13 and Figure A.13. The first function, displayed in Figure B.12, is designed to apply a model and evaluate its performance through MSE and with the algorithm's score. Essentially, it works like a scoring function. Initially, the algorithm is trained (fit), followed by the application of the *prediction* method to generate predictions on unseen data using the training data. In the final segment of this function, the primary objective is to meticulously evaluate the performance of the model using the MSE metric and score metrics. In the second function, depicted in Figure B.13, K-Fold cross-validation was used as the validation method, and the performance of the model was evaluated using metrics such as MAE, MSE, RMSE, R2, and Mean Absolute Percentage Error (MAPE). These metrics are saved in a dataframe to calculate them for each algorithm. The third function remains unchanged from its previous

usage – A.13. Afterward, a range of models were implemented in both classification and regression model:

Classification model

The models used were the same as those in the diagnostic problem and are not detailed here.

ML Models: *Gaussian Naïve Bayes, Logistic Regression, Decision Trees, Random Forest, Neural Networks, Support Vector Machine, Multi-Layer Perceptron with Grid Search, K-Nearest Neighbors, Ensemble Bagging Classifier, Ensemble Adaboost Classifier, Ensemble Voting Classifier, Ensemble Gradient Boosting Classifier, Ensemble Stacking Classifier, Ensemble Extreme Gradient Boosting Classifier*

GP Models: TPOT applied to identify the most effective pipeline for addressing the problem.

Regression model

ML Models: *Linear Regression, Gaussian Process Regression, Decision Trees, Random Forest, Elastic Net, Ridge Regressor, Bayesen Ridge Regressor, Neural Network, Multi-Layer Perceptron with Grid Search, Support Vector Machine, K-Nearest Neighbors, Ensemble Bagging Regressor, Ensemble Adaboost Regressor, Ensemble Gradient Boosting Regressor, Ensemble Voting Regressor, Ensemble Stacking Regressor, Ensemble Extreme Gradient Boosting Regressor*

GP Models: TPOT applied to identify the most effective pipeline for addressing the problem.

Linear Regression

Linear Regression (LiR) was initially tested due to its popularity, speed, and simplicity. It predicts continuous values by finding the optimal-fitting line representing the relationship between the input and the output variables using a linear equation.

Gaussian Process Regression

Versatile and powerful regression technique renowned for its flexibility, capability in estimating uncertainty, and robustness. It represents a non-parametric regression technique that uses Gaussian processes to model the relationship between input and output variables.

Elastic Net

Combines the advantages of both Lasso and Ridge regression by incorporating both L1 and L2 regularization. This versatile LiR approach handles complex real-world scenarios and improves model performance.

Ridge Regression

LiR model augmented by the inclusion of a regularization term within its objective function, serving to mitigate issues related to multicollinearity and overfitting.

Bayesian Ridge Regression

Extends ridge regression model by using a Bayesian framework to estimate model parameters. It combines ridge regularization with prior distributions assigned to the coefficients, allowing for uncertainty quantification and automatic determination of the regularization parameter.

The methodology involved an initial exploration employing simpler models across various feature sets, followed by the implementation of more complex models with differing feature combinations. Extensive parameter testing was conducted to optimize performance. Except for the Extreme Gradient Boosting model and the MLP, which used grid search for hyperparameter exploration, all parameters were manually selected and tuned. Evaluations were performed on various solutions within each model for different feature sets and parameters, facilitating comparisons using metrics across different models. Finally, TPOT, a GP model, was implemented with adjustments made to the parameters and feature sets. This comprehensive approach provided comparative results and addressed the initial problem. These methodologies were applied to both classification and regression problems.

3.3.5. Evaluation

This chapter provides an in-depth overview of the diverse metrics used to evaluate models, divide into 2 parts: classification and regression. For the **classification problem**, distinguishing a recurrent (R) from a non-recurrent (NR) BC, the metrics used are Accuracy, F1-score, Specificity, Sensitivity, and ROC curve. Given their previous use, further explanation will not be provided. For the **regression problem**, predicting the Recurrence Time for R and disease-free time for NR, the metrics used to evaluate the performance of the algorithms are MAE, MSE, RMSE, R-Squared (R^2) and MAPE.

Mean Absolute Error

MAE measures the average absolute discrepancy between the actual and predicted values. This metric demonstrates lower sensitive to outliers compared to others used in this work.

Mean Squared Error

MSE calculates the average squared difference between the actual and predicted values. Squaring the errors significantly penalizes larger errors, rendering MSE more sensitive to outliers.

Root Mean Squared Error

RMSE represents the square root of the MSE, and it provides an estimate of the average magnitude of error made by the models.

R-Squared

R^2 represents the proportion of variance in the dependent variable that can be explained by the independent variable. It ranges from 0 to 1, where 1 indicates perfect fit, meaning all the variability in the dependent variable is explained by the independent variable. On the other hand, a value of 0 indicates that there is no linear relationship between variables.

Mean Absolute Percentage Error

MAPE calculates the average percentage difference between the actual value and the predicted value in relation to the actual values.

4. RESULTS AND DISCUSSION

After diligently implementing a robust methodology, the results stemming from the application of Machine Learning (ML) and Genetic Programming (GP) algorithms will be elucidated and clarified, culminating in drawn conclusions. By analyzing and extracting valuable information from the implementation of these techniques, a deeper understanding of the factors and parameters influencing the diagnosis and prognosis of the disease was attained. Furthermore, meaningful insights were obtained to enhance both the diagnosis and prognosis of breast cancer (BC). This section delves into the outcomes derived from an in-depth and exhaustive analysis of the diverse models gathered and processed during the research. The primary objective is to address the central question posed at the outset of the study and to fulfill this aim, the section is divided into two primary domains: diagnostic results and prognostic results. Within each section, the experimental settings and corresponding results for each model will be outlined. Finally, a detailed overview of the final results will be presented at the end of each section.

4.1. DIAGNOSTIC RESULTS

This section presents detailed results obtained from each model employed in the diagnostic research. Beginning with simpler models and gradually progressively advancing to more complex ones, the approach culminates in a comprehensive description of the final results and a comparative analysis of outcomes obtained from all models will be presented.

Regarding the results obtained from each model

For each model, the process consistently begins with training the respective model using the default parameters, followed by evaluating its performance across different number of variables. Given the balanced nature of the dataset post-application of Synthetic Minority Oversampling Technique (SMOTE), accuracy is used as the metric to assess the model performance across different numbers of features. Moreover, after identifying the optimal number of features for each model based on accuracy, adjustments to specific model parameters are made to potentially augment performance and mitigate overfitting or underfitting. This involves the systematic exploration and testing of different parameter values, prioritizing those yielding the highest accuracy scores. Upon analyzing the results, parameter values achieving the highest scores are carefully recorded. This iterative process is repeated for all parameters, aiming to construct the optimal model by integrating these optimal parameter values. Finally, the optimal model is built by integrating the characteristics and parameters identified from the best-performing model. Its robustness is evaluated using 10-fold cross-validation, with performance metrics such F1-score, accuracy, specificity, and

sensitivity calculated, and results recorded. Detailed explanations for each individual model applied will follow.

Gaussian Naïve Bayes

After exploring the different accuracies, it was observed that the best performance was achieved by employing the top 3 most important features, as evidenced in Table C.1. Based on this finding, the Naïve Bayes (NB) model was constructed, incorporating these 3 significant features, and employing a 10-fold cross-validation approach. Subsequently, metrics including F1-score, accuracy, specificity, and sensitivity were calculated in detail, as depicted in Table C.2. Finally, the accuracy metric of the model was recorded within a dataframe that aggregates the best results obtained from the different models.

Logistic Regression

After exploring the different accuracies, it was observed that employing the top 9 most important features achieved the best performance, as evidenced in Table C.3. Subsequently, manual parameter tuning was conducted to enhance the model performance. In this case, the solver was adjusted while keeping the same number of iterations set to *max_iter* = 300 for optimal convergence of the optimization algorithm. Through this approach, the optimal performance was attained by setting *max_iter* = 300 and using the 'newton-cg' solver. Based on this finding, the LR model was constructed using the identified optimal parameters and the 9 most significant features, as previously defined and illustrated in Table C.4. Then, a 10-fold cross-validation approach was applied. After constructing the model, metrics including F1-score, accuracy, specificity, and sensitivity were computed in detail, as illustrated in Table C.5. Finally, the accuracy metric of the model was recorded within the dataframe that aggregates the best results obtained from all models.

Decision Trees

After evaluating the different accuracies, employing the top 9 most important features resulted in the best performance, as shown in Table C.6. Further manual parameter tuning aimed to improve model performance and reduce underfitting involved adjusting criteria such as *criterion*, *splitter*, *max_depth*, *min_samples_split*, *min_samples_leaf*, *min_weight_fraction_leaf*, and *min_impurity_decrease*, detailed from Table C.7 to C.13. Each parameter was systematically tested with different values to identify the combination yielding the highest accuracy score. After analyzing the results, the parameters that yielded the highest scores were retained. This iterative process determined optimal values: *max_depth* = 5, *min_samples_leaf* = 5. Based on this finding, a Decision Tree (DT) model was constructed using the 9 significant features and optimal parameters, as illustrated in Table C.14. Then, a 10-fold cross-validation approach was employed. After model construction, metrics including F1-score, accuracy, specificity, and sensitivity were computed in detail, as illustrated in Table C.15. Finally, the accuracy of the model was recorded within the dataframe that aggregates the best results from the models.

Random Forest

After evaluating the different accuracies, top 3 most important features yielded the best performance, as evidenced in Table C.16. Further manual parameter tuning aimed to enhance the model performance and reduce underfitting. Parameters such as *n_estimators*, *criterion*, *max_depth*, *min_samples_leaf*, *bootstrap*, and *max_samples* were modified to converge to an optimal solution, as observed from Table C.17 to C.22. Different values were tested for each parameter to identify the combination with the highest accuracy. The best parameter values were retained, resulting in the optimal model configuration: *n_estimators* = 10, *min_samples_leaf* = 2, *max_depth* = 5, *max_samples* = 0.9. Building on this determination, the Random Forest (RF) model was constructed using the 3 significant features and optimal parameters previously defined, as shown in Table C.23. Then, a 10-fold cross-validation approach was employed. Following model construction, metrics such as F1-score, accuracy, specificity, and sensitivity were computed in detail, as illustrated in Table C.24. Finally, the accuracy of the model was recorded in the dataframe with the best results.

Neural Networks

After evaluating different accuracies, the best performance was achieved using the top 3 most important features, as evidenced in Table C.25. Following this, further manual parameter tuning was conducted to enhance model performance and reduce overfitting. Parameters such as *hidden_layer_sizes*, *activation*, *solver*, *alpha*, *learning_rate*, *learning_rate_init*, *max_iter*, and *shuffle* were modified to converge to an optimal solution as illustrated from Table C.26 to C.33. Different values for each parameter were tested to determine the combination yielding the highest accuracy. The best parameter values were retained, resulting in the best model configuration: *hidden_layer_sizes* = (25,25), *alpha* = 0.01, *learning_rate* = 'invscaling'. Building on this determination, the Neural Network model was constructed using the 3 significant features and optimal parameters previously defined, as shown in Table C.34. Then, a 10-fold cross-validation approach was employed. Following model construction, metrics such as the F1-score, accuracy, specificity, and sensitivity were meticulously computed as illustrated in Table C.35. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Probabilistic Neural Networks

After evaluating the different accuracies, the best performance was achieved using the top 9 most important features, as evidenced in Table C.36. Following this, further manual parameter tuning was conducted to enhance model performance and reduce underfitting. Specifically, the parameter *std* was adjusted to find the optimal solution. Different values were tested to identify the combination yielding the highest accuracy score. The highest-scoring parameter values were retained through this iterative process, resulting in the best model configuration. In this example, the optimal parameters were determined to be: *std*=0.5. Building on determination, the Probabilistic Neural Network (PNN) model was constructed, using the 9 significant features and optimal parameters previously defined, as shown in Table C.37. A 10-fold cross-validation approach was then employed. Following model construction, metrics such as the F1-score, accuracy, specificity, and sensitivity were meticulously computed as

illustrated in Table C.38. Finally, the accuracy of the model was recorded in the dataframe with the best results.

Linear Discriminant Analysis

After examining the various accuracies, it was found that the best performance was achieved using all features, as evidenced in Table C.39. After noting this, additional manual parameter testing was conducted to improve model performance and reduce underfitting. The *solver* was adjusted to converge on an optimal solution by testing various parameter values to identify the combination yielding the highest accuracy score. After analysis, parameters with the highest scores were retained, leading to the selection of the best model configuration. In this example, the optimal parameters were determined using the default values. Building on this decision, the Linear Discriminant Analysis model was built using all features and optimal parameters as previously defined, shown in Table C.40. A 10-fold cross-validation approach was then employed. Post model construction, metrics such as F1-score, accuracy, specificity, and sensitivity were computed as illustrated in Table C.41. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Support Vector Machine Classifier

After evaluating the different accuracies, it was observed that the top 5 most important features achieved the best performance, as evidenced in Table C.42. Following this, further manual parameter experimentation aimed at enhancing model performance and reducing overfitting, adjusting the *C* (regularization parameter), *kernel*, *gamma*, *shrinking*, *probability*, *max_iter* to converge to an optimal solution, as illustrated from Table C.43 to C.46. Each parameter underwent testing with various values to determine the combination yielding the highest accuracy score. The parameter values that achieved the highest scores were retained through this process, resulting in the selection of the best model configuration. In this example, the optimal parameters were determined as: *C* = 10. Building on this determination, the SVMmodel was constructed using the 5 significant features and optimal parameters previously defined, as depicted in Table C.47. Then, a 10-fold cross-validation approach was employed. After constructing the model, metrics such as F1-score, accuracy, specificity, and sensitivity were computed as illustrated in Table C.48. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Multi-Layer Perceptron with Grid Search

After exploring the different accuracies, it was observed that employing all important features achieved the best performance, as evidenced in Table C.49. Following this, further parameter experimentation was conducted using *GridSearchCV*, where a parameter space defined by me was applied. This parameter space included different values of *hidden_layer_sizes*, *activation*, *solver*, *alpha*, and *learning_rate* to find an optimal solution detailed in Table C.50 and C.51. After analysis, the parameter values yielding to the highest scores were retained. In this example, the optimal parameters were determined as follows: *activation* = 'tanh', *alpha* = 0.0001, *hidden_layer_sizes* = (25,25), *solver* = 'adam', *learning_rate* = 'invscaling'. Building on this determination, the Multi-Layer Perceptron (MLP) with Grid Search model was constructed

using all features and optimal parameters previously defined, as depicted in Table C.52. Then, a 10-fold cross-validation approach was employed. Following model construction, metrics such as F1-score, accuracy, specificity, and sensitivity were meticulously computed as illustrated in Table C.53. Finally, the accuracy of the model was recorded within the dataframe with the best results.

K-Nearest Neighbors

After exploring the diverse accuracies, it was found that employing the top 9 most important features achieved the best performance, as shown in Table C.54. Subsequently, further manual parameter experimentation was conducted to enhance model performance and reduce overfitting. Parameters such as *n_neighbor* and *weights* were modified to find an optimal solution, as illustrated in Table C.55 and Table C.56. Each parameter underwent testing with different values to determine the combination yielding the highest accuracy score. Following the analysis, the parameter values corresponding to the highest scores were retained. This process was repeated for all parameters, culminating in the selection of the best model configuration. In this example, the default parameters were determined to be the optimal parameters. Building on this determination, the K-Nearest Neighbors (KNN) model was constructed using the top 9 significant features and the optimal parameters previously defined, as depicted in Table C.57. Subsequently, a 10-fold cross-validation approach was employed. After constructing the model, metrics such as F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed as illustrated in Table C.58. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Ensemble Bagging Classifier

After evaluating the different accuracies, it was found that the top 9 most important features achieved the best performance, as evidenced in Table C.59. Following this, additional manual parameter experimentation was conducted to enhance the model performance. Adjustments were made to parameters such as *base_estimator*, *n_estimators*, *max_samples*, *random_state*, *bootstrap*, *bootstrap_features*, *warm_start*, and *oob_score*, aiming to converge to an optimal solution, detailed from Table C.59 to Table C.63 and Figure C.1. Each parameter underwent testing with different values to determine the combination yielding the highest accuracy score. Following analysis, the optimal parameters were identified as: *base_estimator* = SVC_model_final, *n_estimators* = 12, *random_state* = 1. As the created SVMmodel achieved the best results it was selected as the base estimator for the final model. Building on this determination, the Ensemble Bagging Classifier model was constructed, using the top 9 significant features and optimal model parameters previously defined, as depicted in Table C.64. Subsequently, a 10-fold cross-validation approach was employed. After constructing the model, metrics such as the F1-score, accuracy, specificity, and sensitivity were meticulously computed as illustrated in Table C.65. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Ensemble Adaboost Classifier

After exploring the different accuracies, it was observed that employing all features achieved the best performance, as shown in Table C.67. Following this observation, further manual parameter experimentation was conducted to enhance the model performance and reduce underfitting. Parameters such as *base_estimator*, *n_estimators*, *learning_rate*, and *algorithm* were modified to converge to an optimal solution, as observed from Table C.67 to C.69 and Figure C.2. Each parameter underwent testing with different values to determine the combination yielding the highest accuracy score. Following analysis, the optimal parameters were identified as: *base_estimator* = *LR_model_newton_3Features*, *n_estimators* = 395, *learning_rate* = 1.0. Building on this determination, the Ensemble Adaboost Classifier model was constructed using all features and optimal model parameters previously defined, as depicted in Table C.70. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, metrics such as the F1-score, accuracy, specificity, and sensitivity were computed as illustrated in Table C.71. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Ensemble Voting Classifier

The construction of the final Voting Classifier model began with creating 13 estimators from previously developed models, as shown in Figure C.3. These models were trained, and their performances were evaluated across different sets of numbers of variables. After analysis, the chosen estimator was: *estimators_4* = [('gb', *gbc_model_final*), ('ac', *AdC_model_final*), ('bc', *BC_model_final*)]. The best performance was achieved using all features, as indicated in Table C.72. Further manual parameter tuning was conducted to enhance model performance, adjusting *weights* to converge on an optimal solution, as seen in Table C.73. Different parameter values were tested to find the highest accuracy score, retaining the best-performing values for all parameters. The optimal parameters determined were: *estimators* = *estimators*. Building upon this determination, the Ensemble Voting Classifier model was constructed, using all features and the optimal parameters previously identified, as shown in Table C.74. A 10-fold cross-validation approach was then employed. The F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed as illustrated in Table C.75. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Ensemble Gradient Boosting Classifier

After analyzing different accuracies, it was found that the best performance was achieved using the top 9 most important features, as shown in Figure C.76. Following this observation, further manual parameter experimentation was conducted to enhance model performance and reduce underfitting. Parameters such as *loss*, *learning_rate*, *random_state*, *n_estimators*, *subsample*, *criterion*, *max_depth*, *min_samples_leaf*, and *min_impurity_decrease* were modified to achieve an optimal solution, as seen from Table C.77 to C.83 and from Figure C.4 to C.5. Each parameter was tested to find the combination yielding the highest accuracy. The best-performing parameter values were retained, resulting in the optimal configuration: *learning_rate* = 0.55, *n_estimators* = 104, *max_depth* = 3. Building upon this determination, the Ensemble Gradient Boost Classifier model was constructed using the top 9 significant

features and the optimal parameters previously identified, as visually represented in Table C.84. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, metrics such as the F1-score, accuracy, specificity, and sensitivity were calculated as illustrated in Table C.85. Finally, the accuracy metric of the model was recorded within the dataframe with the best results.

Ensemble Stacking Classifier

The initial stage in constructing the final Voting Classifier model involved the creation of 15 estimators, each comprising distinct models developed previously, as shown in Figure C.6. These models were trained, and their performances were evaluated across different sets of variables. After analyzing the results, the optimal number of features for each estimator was recorded, and the final estimator was updated for each selected model. All previously tested estimators were thoroughly explored. After a meticulous examination, the Neural Network model was chosen as the final estimator for the model incorporating all features. The chosen estimator configuration is as follows: *estimators_1 = [('nv', modelNB), ('lr', LR_model_newton_3Features), ('rf', RF_model_final), ('dt', DT_model_final), ('nn', NN_model_final), ('svc', SVC_model_final)]*. Building on this determination, the Ensemble Stacking Classifier model was constructed incorporating all features along with the optimal model parameters identified earlier, as shown in Table C.86. A 10-fold cross-validation approach was then employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were computed, as illustrated in Table C.87. Finally, the accuracy metric of the model was recorded within the dataframe containing the best results.

Ensemble Extreme Gradient Boosting Classifier

After exploring the different accuracies, it was observed that the best performance was achieved by employing the top 9 most important features, as evidenced in Table C.88. Following this observation, further parameter tuning was conducted using GridSearchCV with a parameter space defined by me. The parameter space included *max_depth*, *learning_rate*, *n_estimators*, *subsample*, and *colsample_bytree* to converge to an optimal solution. The optimal parameters were identified through this process and shown in Table C.89 and Table C.90. Following the analysis, the best-performing parameter values were retained, resulting in the optimal configuration: *colsample_bytree = 1*, *learning_rate = 0.3*, *max_depth = 6*, *n_estimators = 100*, *subsample = 1*. Building on this determination, the Ensemble Extreme Gradient Boosting Classifier model was constructed incorporating the top 9 significant features and the optimal parameter previously identified, as shown in Table C.91. Subsequently to this, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were computed as illustrated in Table C.91. Finally, the accuracy metric of the model was recorded within the dataframe including the best results.

TPOT

Initially, to understand the impact of each parameter adjustments on the optimization process, a simple model with TPOT's default parameter, except for the number of generations

set to 5, was created, trained, and evaluated. The model's accuracy was assessed across different variable sets, and the best performance was achieved using the top 9 most important features, as evidenced in Table C.93. Before modifying parameters, three additional models were generated with 20, 50, and 100 generations, as illustrated in Table C.94, C.95 and C.96, respectively. Due to computational constraints, subsequent experiments used 5 generations. Manual parameter tuning was then conducted to enhance performance and reduce underfitting by adjusting *population_size*, *mutation_rate*, *crossover_rate*, *cv*, and *random_state*, as observed from Table C.97 to C.101. Each parameter was tested to determine the combination yielding the highest accuracy. Following the analysis, the best-performing values were retained, culminating in the optimal parameters: *generations* = 5, *population_size* = 75, *verbosity* = 2, *mutation_rate* = 0.3, *crossover_rate* = 0.3. Building upon this determination, the final model was constructed using the top 9 significant features along with the optimal parameters previously identified, as shown in Figure C.7. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were computed as illustrated in Table C.102. Finally, the accuracy metric of the model was recorded within the dataframe containing the best results. Furthermore, the best model for this problem, `MLPClassifier(input_matrix, alpha=0.001, learning_rate_init=0.001)`, was determined, and its accuracy was recorded in the dataframe.

Regarding the final results obtained

After implementing all developed models, a comprehensive comparison was conducted to assess their performance. This evaluation involved utilizing a range of ML evaluation metrics including accuracy, F1-score, specificity, sensitivity, and Receiver Operating Characteristic (ROC) curve analysis, to thoroughly compare and analyze the results obtained from the different models.

As previously outlined throughout the document, a dataframe was created to represent the accuracy of all models, and Table C.103 visually depicts this information. This representation clearly shows that the models demonstrated commendable results, indicating promising performance. In this context, accuracy serves as a valuable metric for gauging the models' effectiveness in discriminating between malignant and benign instances. Despite all models demonstrating excellent results, the models that stood out the most for their higher values and similar training and validation results are the following: MLP, Voting Classifier, and Stacking Classifier.

The F1-Score, illustrated in Figure C.8, enables the evaluation of the models by combining precision and recall scores to measure accuracy. The results clearly indicate a very accurate identification of both benign and malignant cases, with minimal disparity between the training and validation sets. Among the models evaluated, MLP, Stacking Classifier, and PNN

consistently demonstrated outstanding performance, characterized by higher values and closely aligned training and validation results.

The specificity, depicted in Figure C.9, allows the calculation of the true negative rate, representing the probability of a negative test result given that the individual is truly negative. The results reveal significant discrepancies between the training and validation datasets, suggesting challenges for the models in accurately discerning between malignant and benign breast cytology, particularly in correctly predicting negative results. This led to instances where benign cells were misclassified as malignant.

The sensitivity, depicted in Figure C.10, computes the true positive rate, indicating the probability of a positive test result given that the individual is truly positive. The results indicate no significant discrepancies between the train and validation datasets in this regard. It is essential to emphasize the critical role of sensitivity in this context. The positive values obtained affirm the existence of well-developed models capable of accurately detecting, predicting, and classifying positive diagnostics.

Finally, the ROC curve, shown in Figure C.11, illustrates the performance of the classification models across all thresholds by plotting the true positive rate against the false positive rate. Overall, the results indicate remarkably performance of the models, achieving high accuracy in diagnosing patients with and without the disease based on the test results. It is noteworthy that all models consistently achieve an AUC above 0.90, underscoring their robustness and effectiveness.

To conclude this results chapter, it is notable that the results of the diagnostic study demonstrate promising outcomes when using both ML and GP algorithms for enhancing BC diagnosis and prognosis.

4.2. PROGNOSTIC RESULTS

This section provides a comprehensive overview of the results obtained from each model used in addressing both prognostic research problems. As previously outlined, a range of models was employed to tackle classification and regression challenges, starting from simpler approaches and advancing to more sophisticated ones. At the conclusion of each segment addressing the research problems, a detailed description of the final results will be provided, along with a comparative analysis of outcomes across all models.

Classification problem

Regarding the results obtained from each model

For each model, the process consistently begins with training the respective model using the default parameters, followed by an evaluation of its performance across different number of variables. Given the balanced nature of the dataset post-application of SMOTE, accuracy is used as the metric for assessing the model performance across different numbers of features. Furthermore, after determining the most accurate number of features for each model, adjustments to specific model parameters are executed and changed with the intention of potentially augmenting performance and mitigating overfitting or underfitting. This approach involves systematically exploring and testing different parameter values, prioritizing those yielding the highest accuracy scores. Upon analyzing the results, parameter values achieving the highest scores are meticulously recorded. This iterative process is repeated for all parameters, aiming to construct an optimal model by integrating the most effective parameter values. Finally, the final model is formulated by incorporating the characteristics and parameters of the best-performing model identified. Its robustness is evaluated using 10-fold cross-validation, with performance metrics including F1-score, accuracy, specificity, and sensitivity calculated, and the results recorded. Detailed explanations will now be provided for each individual model applied.

Gaussian Naïve Bayes

After exploring the different accuracies, it was observed that the best performance was achieved by employing the top 5 most important features, as evidenced in Table D.1. Based on this finding, the NB model was constructed, incorporating these 5 significant features, and employing a 10-fold cross-validation approach. Subsequently, metrics including F1-score, accuracy, specificity, and sensitivity were calculated in detail, as illustrated in Table D.2. Finally, the accuracy metric of the model was recorded within a dataframe which aggregates the best results obtained from the different models.

Logistic Regression

After exploring the different accuracies, it was observed that employing the top 5 most important features achieved the best performance, as evidenced in Table D.3. Following this, manual parameter tuning was conducted to enhance the model performance. In this instance, the solver was adjusted while keeping the same number of iterations set to `max_iter = 300` for optimal convergence of the optimization algorithm. Through this approach, the optimal performance was attained by setting `max_iter = 300` and using the 'newton-cg' solver. Based on this finding, the LR model was constructed incorporating the identified optimal parameters and the 5 most significant features, as previously defined and illustrated in Table D.4. Subsequently, a 10-fold cross-validation approach was applied. Following the model construction, metrics such as F1-score, accuracy, specificity, and sensitivity were computed in detail, as illustrated in Table D.5. Finally, the accuracy metric of the model was recorded within the dataframe that aggregates the best results obtained from all models.

Decision Trees

After evaluating the different accuracies, it was observed that the best performance was achieved using the top 9 most important features, as evidenced in Figure D.6. Following this, further manual parameter tuning aimed to improve model performance and reduce underfitting involved adjusting criteria such as *criterion*, *splitter*, *max_depth*, *min_samples_split*, *min_samples_leaf*, *min_weight_fraction_leaf*, and *min_impurity_decrease*, detailed from Figure D.7 to D.13. Each parameter was systematically tested with different values to identify the combination yielding the highest accuracy score. Following the analysis, the parameter values corresponding to the highest scores were retained. This iterative process determined optimal values: *max_depth* = 5, *min_samples_leaf* = 100, *min_samples_split* = 100. Building on this finding, the DT model was constructed incorporating the top 9 significant features and optimal parameters, as illustrated in Table D.14. Subsequently to this, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed, as illustrated in Table D.15. Finally, the accuracy of the model was recorded within the dataframe that aggregates the best results from the models.

Random Forest

After exploring the different accuracies, it was observed that the best performance was achieved by employing all features, as evidenced Table D.16. Following this observation, further manual parameter tuning was conducted to enhance the model performance and reduce underfitting. In this process, parameters such as *n_estimators*, *criterion*, *max_depth*, *min_samples_leaf*, *bootstrap*, and *max_samples* were modified to find the optimal solution, as shown Table D.17 to D.22. The main idea was to test different values for each parameter to determine the combination yielding the highest accuracy score. The analysis led to the retention of the parameter values corresponding to the highest scores. This process was repeated for all parameters, resulting in the selection of the best model configuration. In this example, the optimal parameters were determined as follows: *n_estimators* = 10, *min_samples_leaf* = 4, *max_depth* = 1. Building on this determination, the RF model was constructed using all features and optimal parameters previously defined, as shown in Figure D.23. Then, a 10-fold cross-validation approach was employed. Following model construction, metrics such as F1-score, accuracy, specificity, and sensitivity were computed in detail, as illustrated in Figure D.24. Finally, the accuracy of the model was recorded in the dataframe with the best results.

Neural Networks

After exploring the different accuracies, the best performance was achieved by employing the top 5 most important features, as evidenced in Table D.25. Following this observation, further manual parameter experimentation was conducted to enhance the model performance and reduce overfitting. Parameters such as *hidden_layer_sizes*, *activation*, *solver*, *alpha*, *learning_rate*, *learning_rate_init*, *max_iter*, and *shuffle* were modified to converge to an optimal solution. Each parameter was tested to determine the values yielding the highest accuracy score. The analysis resulted in the retention of the optimal parameters:

hidden_layer_sizes = (50,100), *alpha* = 0.01, *max_iter* = 400. Building upon this determination, the Neural Network model was constructed incorporating the top 5 significant features and optimal parameters previously defined, as shown in Table D.26. Subsequently to this, a 10-fold cross-validation approach was employed. Following the model construction, metrics such as F1-score, accuracy, specificity, and sensitivity were computed as illustrated in Table D.27. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Support Vector Machine Classifier

After exploring the accuracies, the best performance was achieved by employing the top 9 most important features, as evidenced in Table D.28. Subsequently, further manual parameter tuning was conducted to enhance model performance and reduce overfitting. Parameters such as the *C* (regularization parameter), *kernel*, *gamma*, *shrinking*, *probability*, *max_iter* were adjusted to find the optimal combination. Each parameter was tested to determine the combination of values yielding the highest accuracy score. The analysis led to the retention of the optimal parameters including *gamma* = 'auto'. Building upon this determination, the SVM model was constructed using the top 9 significant features and optimal model parameters previously defined, as depicted in Table D.29. Then, a 10-fold cross-validation approach was employed. After constructing the model, metrics such as F1-score, accuracy, specificity, and sensitivity metrics were computed, as illustrated in Table D.30. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Multi-Layer Perceptron with Grid Search

After exploring the different accuracies, the best performance was achieved using the top 5 most important features, as evidenced in Table D.31. Following this observation, parameter tuning was conducted using *GridSearchCV* with a predefined parameter space defined by me. The parameter space included different values for *hidden_layer_sizes*, *activation*, *solver*, *alpha*, and *learning_rate* to converge to an optimal solution. The optimal parameters were identified through this process, as observed in Table D.32 and Table D.33. After analysis, the parameter values yielding to the highest scores were retained. In this example, the optimal parameters were determined as follows: *activation* = 'tanh', *alpha* = 0.0001, *hidden_layer_sizes* = (25,25), *solver* = 'adam', *learning_rate* = 'invscaling'. Building upon this determination, the MLP with Grid Search model was constructed using the 5 most significant features and optimal parameters previously defined, as depicted in Table D.34. Afterward, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed as illustrated in Table D.35. Finally, the accuracy of the model was recorded within the dataframe with the best results.

K-Nearest Neighbors

After examining the different accuracies, it was found that employing the complete set of features resulted in the highest performance, as indicated in Table D.36. Subsequently, the process proceeded with additional manual parameter tuning to improve the model performance and mitigate overfitting. Specifically, parameters such as *n_neighbors* and

weights were adjusted to converge to an optimal solution. Each parameter was systematically tested with different values to identify the combination that yielded the highest accuracy score. The values corresponding to the highest scores for each parameter were adopted. This iterative process was repeated for all parameters, leading to the determination of the best model configuration. Ultimately, it was found that the default parameters provided the optimal configuration in this case. Building on this determination, the KNN model was constructed using the set of all features along with the optimal parameters previously defined, as depicted in Table D.37. After, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed, as illustrated in Table D.38. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Ensemble Bagging Classifier

After evaluating the different accuracies, it was observed that the highest performance was achieved by employing the complete set of all features, as evidenced in Figure D.39. After, the process involved further manual experimentation with parameters to enhance the model performance. Key adjustments were made to parameters such as *base_estimator*, *n_estimators*, *max_samples*, *random_state*, and other features, namely *bootstrap*, *bootstrap_features*, *warm_start*, and *oob_score*, with the aim of reaching an optimal solution. For each parameter, a range of values was tested to identify the combination that yielded the highest accuracy score. Following thorough analysis, the parameter values associated with the highest scores were selected. The iterative procedure was repeated for all parameters. Ultimately the optimal configuration was determined: *base_estimator* = *KNeighborsClassifier()*, *n_estimators* = 32, *random_state* = 56, *warm_start* = True. Building on this determination, the Ensemble Bagging Classifier model was constructed, using the set of all features along with the identified optimal model parameters previously, defined as depicted in Table D.40. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, metrics such as the F1-score, accuracy, specificity, and sensitivity were computed, as illustrated in Table D.41. Finally, the accuracy metric of the model was recorded within *df_results_o* dataframe. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Adaboost Classifier

After exploring the different accuracies, it was found that the best performance was achieved by employing the top 3 most important features, as evidenced in Table D.42. Subsequently, the process continued with additional manual parameter tuning aimed at improving the model performance and addressing underfitting. Adjustments were made to parameters such as *base_estimator*, *n_estimators*, *learning_rate*, and *algorithm* to converge on an optimal solution. The approach involved testing different values for each parameter to determine combination that produced the highest accuracy score. Following the analysis, the parameter values associated with the highest scores were selected and this iterative process was applied to all parameters involved. Ultimately, the optimal parameters configuration was defined as:

base_estimator = LR_model_newton_3Features, n_estimators = 395, learning_rate = 1.0. Building on this determination, the Ensemble Adaboost Classifier model was constructed, incorporating the top 3 most significant features along with the optimal model parameters previously defined, as visually represented in Table D.43. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed as illustrated in Table D.44. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Ensemble Voting Classifier

The initial phase of constructing the final Voting Classifier model involved creating 13 estimators, each comprising distinct models developed previously, as illustrated in Figure D.1. Following this, the models within the estimators were trained, and subsequently, their performances across different sets of variables were evaluated. Upon analyzing the results, a deliberate decision was made to proceed with the estimator configuration: *estimators_9 = [['svc', SVC_model_final], ('bc', BC_model_final)]*. Subsequently, it was observed that the optimal performance was achieved by employing the top 9 most important features, as evidenced in Table D.45. Following this observation, the process proceeded with further manual parameter tuning aimed at enhancing the model performance. Specifically, adjustments were made to *weights* to converge on an optimal solution. For each parameter, different values were tested to identify the combination yielding the highest accuracy score. Following the analysis, the parameter values associated with the highest scores were selected and the iterative process was repeated for all parameters. Ultimately, the optimal parameter configuration was determined, incorporating the selected estimators: *estimators = estimators*. Building upon this determination, the Ensemble Voting Classifier model was constructed, incorporating the top 9 most significant features and the optimal parameters previously identified, as shown in Table D.46. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed, as illustrated in Table D.47. Finally, the accuracy of the model was recorded within the dataframe with the best results.

Ensemble Gradient Boosting Classifier

After exploring the different accuracies, it was observed that the best performance was achieved by employing the set of all features, as evidenced in Table D.48. Following this observation, further manual parameter tuning was conducted to enhance the model performance and reduce underfitting. Parameters such as *loss, learning_rate, random_state, n_estimators, subsample, criterion, max_depth, min_samples_leaf, and min_impurity_decrease* were modified to converge on an optimal solution. For each parameter, different values were tested to identify the combination yielding the highest accuracy score. After analysis, the parameter values corresponding to the highest scores were retained. This iterative process was repeated for all parameters, culminating in the selection of the best model configuration. The optimal parameters were determined as follows: *learning_rate = 0.1, n_estimators = 80, max_depth = 1, random_state = 1*. Building upon this

determination, the Ensemble Gradient Boosting Classifier model was constructed using the set of all features and the optimal parameters previously identified, as visually represented in Table D.49. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, metrics such as the F1-score, accuracy, specificity, and sensitivity were computed, as illustrated in Table D.50. Finally, the accuracy metric of the model was recorded within the dataframe with the best results.

Ensemble Stacking Classifier

The initial stage in constructing the final Voting Classifier model involved the creation of 15 estimators, each comprising distinct models developed previously, as observed in Figure D.2. The models within the estimators were trained, and their performances were evaluated across different sets of variables. After analyzing the results, the optimal number of features for each estimator was recorded, and the final estimator was updated for each selected model. In this process, all previously tested estimators were thoroughly explored. After a meticulous examination, the Neural Network model was chosen as the final estimator for the model incorporating the top 9 most important features. The chosen estimator configuration is as follows: *estimators_2 = [('gb', gbc_model_final), ('bc', BC_model), ('svc', SVC_model_final)]*. Building on this determination, the Ensemble Stacking Classifier model was constructed, using the aforementioned 9 significant features along with the optimal model parameters identified earlier and represented in Table D.51. A 10-fold cross-validation approach was then employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed, as illustrated in D.52. Finally, the accuracy metric of the model was recorded within the dataframe containing the best results.

Ensemble Extreme Gradient Boosting Classifier

After exploring the different accuracies, it was observed that the best performance was achieved by employing the set of all features, as evidenced in Table D.53. Following this observation, further parameter experimentation was conducted using *GridSearchCV*, with a parameter space defined by me. The parameter space included *max_depth*, *learning_rate*, *n_estimators*, *subsample*, and *colsample_bytree* to converge to an optimal solution. Through this process, the optimal parameters were identified, and this can be observed in Table D.54 and Table D.55. Following the analysis, the best-performing parameter values were retained, resulting in the optimal configuration: *colsample_bytree = 1*, *learning_rate = 0.3*, *max_depth = 6*, *n_estimators = 100*, *subsample = 1*. Building upon this determination, the Ensemble Extreme Gradient Boosting Classifier model was constructed, incorporating the aforementioned all features along with the identified optimal model parameters, as shown in Table D.56. Subsequently to this, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were computed, as illustrated in Table D.57. Finally, the accuracy metric of the model was recorded within the dataframe including the best results.

TPOT

Initially, to understand the impact of each parameter adjustments on the optimization process, a simple model with TPOT's default parameter, except for the number of generations set to 5, was created, trained, and evaluated. The model's accuracy was assessed across different variable sets, and the best performance was achieved by employing the top 9 most important features, as evidenced in Table D.58. Following this observation, further manual parameter experimentation was then conducted to enhance performance and reduce underfitting by adjusting *population_size*, *mutation_rate*, *crossover_rate*, *cv*, and *random_state*, as observed from Table C.59 to Table C.63. Each parameter was tested to determine the combination yielding the highest accuracy. Following the analysis, the best-performing values were retained. This process was repeated for all parameters, culminating in the selection of the best pipeline with optimal parameters. In this example, the optimal parameters were determined as follows: *generations* = 5, *verbosity* = 2, *mutation_rate* = 0.5, *crossover_rate* = 0.4, *cv* = 50, *random_state* = 15. Building upon this determination, the final model was constructed using all features along with the optimal model parameters previously identified and shown in Figure D.3. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the F1-score, accuracy, specificity, and sensitivity metrics were meticulously computed, as illustrated in Table D.64. Finally, the accuracy metric of the model was recorded within the dataframe containing the best results. Furthermore, the best model for this problem, `ExtraTreesClassifier(FastICA(input_matrix, tol=0.35), bootstrap=False, criterion=entropy, max_features=0.95, min_samples_leaf=5, min_samples_split=10, n_estimators=100)` was determined, and its accuracy was recorded in the dataframe.

Regarding the final results obtained

In the classification problem, while predicting the outcome as Recurrent or Non-Recurrent:

After implementing all developed models, a comprehensive comparison was conducted to assess their performance. This evaluation involved utilizing a range of ML evaluation metrics including accuracy, F1-score, specificity, sensitivity, and ROC curve analysis, to thoroughly compare and analyze the results obtained from the different models.

As previously detailed throughout the document, a dataframe was created to represent the accuracy of all models, and Table D.65 visually depicts this information. This representation clearly shows that the models exhibited suboptimal results. In this context, accuracy serves as a valuable metric for gauging the models' effectiveness in discriminating between recurrent and non-recurrent instances. These non-ideal values may be attributed to the relatively small size of the dataset. Among the models, the Voting Classifier stood out the most for its higher values and for demonstrating very similar training and validation results.

The F1-Score, illustrated in Figure D.4, allowing the evaluation of the models by combining precision and recall scores to measure accuracy. As evident from the results, there were suboptimal positive outcomes in accurately identifying both recurrent and non-recurrent

cases, with some discrepancies between the training and validation sets. Among the models, the Voting Classifier and SVM stood out the most for their higher values and for demonstrating very similar training and validation results.

The specificity, depicted in Figure D.5, allows the calculation of the true negative rate, representing the probability of a negative test result given that the individual is truly negative. As evident from the results, it's remarkable that there were some discrepancies between the training and validation datasets. This suggests that the models faced challenges in accurately distinguishing between recurrent and non-recurrent breast cytology, particularly in correctly predicting negative values. Consequently, there were instances where non-recurrent cells were misclassified as recurrent.

The sensitivity, depicted in Figure D.6, allows the calculation of the true positive rate, representing the probability of a positive test result given that the individual is truly positive. As evident from the results, there were no significant discrepancies between the train and validation datasets in this regard. It's crucial to emphasize that sensitivity is a vital metric for this type of problem. The positive values obtained indicate that there are developed models capable of accurately detecting, predicting, and classifying positive prognostics. Among the models, the Bagging Classifier and KNN stood out the most for their higher values and for demonstrating very similar training and validation results.

Lastly, the ROC curve, as illustrated in Figure D.7, provides insight into the performance of the classification models across all classification thresholds by plotting the true positive rate against the false positive rate. Overall, the results are not so promising as anticipated, which can be attributed to the size of dataset. It is noteworthy to highlight the results from the GP algorithm – TPOP- which demonstrated very positive outcomes comparable to those achieved by the ML algorithms. Particularly notable is the Area Under the ROC Curve (AUC) of TPOT exceeding 0.940, surpassing others with values less than 0.90.

In conclusion, the findings from the prognostic study reveal less optimistic outcomes in distinguishing recurrent from non-recurrent cases when employing ML and GP algorithms to enhance BC diagnosis and prognosis. Despite initial expectations, the results indicate challenges in achieving significant improvements in prognostic accuracy and assessment through the methodologies applied.

Regression problem

Regarding the results obtained from each model

For each model, the process consistently begins with training the respective model using the default parameters, followed by an evaluation of its performance across different number of variables. Given the balanced nature of the dataset post-application of SMOTE, Mean Squared Error (MSE) and R-Squared (R^2) serve as the metrics for assessing the model performance

across different numbers of features. Furthermore, after determining the most accurate number of features for each model, adjustments to specific model parameters are executed and changed with the intention of potentially augmenting performance and mitigating overfitting or underfitting. This approach involves systematically exploring and testing different parameter values, prioritizing those yielding the lowest MSE. Upon analyzing the results, the parameter values associated with the highest scores (lowest error) are meticulously recorded. This iterative process is repeated for all parameters, with the overarching objective of constructing an optimal model by integrating the most effective parameter values. Finally, the final model is formulated by incorporating the characteristics and parameters of the best-performing model identified. Its robustness is evaluated using 10-fold cross-validation, calculating performance metrics including Mean Absolute Error (MAE), MSE, Root Mean Squared Error (RMSE), R^2 and Mean Absolute Percentage Error (MAPE), with the results recorded. A more detailed explanation will now be provided for each individual model applied.

Linear Regression

After exploring the different MSE and R^2 values, it was observed that the best performance was achieved by employing the top 5 most important features, as evidenced in Table E.1. Following this determination, the process proceeded with the construction of a Linear Regression (LiR) model, incorporating the aforementioned 5 significant features, and employing a 10-fold cross-validation approach. Following the model construction, metrics including MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.2. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Gaussian Process Regression

After exploring the different MSE and R^2 values, it was observed that the best performance was achieved by using the top 3 most important features, as evidenced in Table E.3. Following this, a Gaussian Process Regression model was constructed, incorporating these 3 significant features, and employing a 10-fold cross-validation approach. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.4. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Decision Trees

After exploring the different MSE and R^2 values, it was observed that the best performance was achieved by employing the top 5 most important features, as evidenced in Table E.5. Following this observation, the process proceeded with further manual parameter experimentation was conducted to enhance the model's performance and reduce underfitting. Parameters such as *criterion*, *splitter*, *max_depth*, *min_samples_split*, *min_samples_leaf*, *min_weight_fraction_leaf*, *min_impurity_decrease* were adjusted to converge on an optimal solution. For each parameter, different values were tested to

determine the combination yielding the best performance. Following the analysis, the parameter values corresponding to the highest scores were retained. This iterative process was repeated for all parameters, culminating in the selection of the best model configuration. In this example, the optimal parameters were determined as follows: *max_depth* = 5, *min_samples_split* = 10. Building on this determination, the DT model was constructed, incorporating the top 5 most significant features and the identified optimal model parameters, as shown in Table E.6. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were calculated as illustrated in Table E.7. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Random Forest

After exploring the different MSE and R^2 , it was observed that the best performance was achieved by employing the top 5 most important features, as evidenced in Table E.8. Following this observation, further manual parameter tuning was conducted to enhance the model performance and reduce underfitting. Parameters such as *n_estimators*, *criterion*, *max_depth*, *min_samples_leaf*, *min_samples_split*, and *max_samples* were modified to converge on an optimal solution. The main idea for each parameter was to test different values to determine the combination yielding the best performance by minimizing the MSE. After analysis, the parameter values corresponding to the highest scores were retained. This iterative process was repeated for all parameters, culminating in the selection of the best model configuration. In this example, the optimal parameters were determined as follows: *n_estimators* = 10, *min_samples_leaf* = 5, *max_depth* = 4, *min_samples_split* = 10. Building upon this determination, the RF model was constructed, incorporating the aforementioned 5 significant features along with the identified optimal model parameters, as represented in Table E.9. Subsequently, a 10-fold cross-validation approach was employed. Then, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.10. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Elastic Net

After exploring the different MSE and R^2 values, the best performance was achieved by using all features, as evidenced in Table E.11. Following this, further manual parameter experimentation was conducted to enhance the model performance and reduce underfitting. In this instance, the *alpha*, *l1_ratio*, and *max_iter* were modified to converge on an optimal solution. Each parameter was tested with different values to find the combination yielding the best performance by getting the minimum MSE. After analysis, the best-performing parameter values were retained. This process was repeated for all parameters, resulting in the selection of the optimal model configuration. The optimal parameters were determined as follows: *alpha* = 0.1, *l1_ratio* = 0.25. Building upon this determination, the Elastic Net model was constructed, incorporating all features along with the identified optimal model

parameters, as visually represented in Table E.12. Then, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.13. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Ridge Regression

After exploring the different MSE and R^2 , it was observed that the best performance was achieved by employing the top 5 most important features, as evidenced in Table E.14. Following this observation, further manual parameter tuning was conducted to enhance the model performance and reduce underfitting. Parameters such as *alpha*, *solver*, and *max_iter* were modified to converge on an optimal solution. Each parameter underwent testing with different values to find the combination yielding the best performance by getting the minimum MSE. After thorough analysis, the parameter values associated with the highest scores were selected. This iterative process was repeated for all parameters, resulting in the determination of the best model configuration. In this example, the optimal parameters were determined as follows: *alpha* = 10.0. Building on this determination, the Ridge Regression model was constructed using the top 5 most significant features and the identified optimal parameters, as defined previously. This construction is visually represented in Table E.15. Then, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.16. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Bayesian Ridge Regression

After exploring the different MSE and R^2 values, it was observed that the best performance was achieved by using all the features, as evidenced in Table E.17. Following this observation, further manual parameter experimentation was conducted to enhance the model performance and reduce underfitting. In this instance, *alpha_1*, *alpha_2*, and *n_iter* were modified to converge on an optimal solution. Each parameter underwent testing with different values to find the combination yielding the lowest MSE. After thorough analysis, it was determined that the default parameters provided the optimal configuration for this example. Building on this determination, the Bayesian Ridge Regression model was constructed using all features and the identified optimal parameters, as previously defined. This construction is visually represented in Table E.18. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.19. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Neural Networks

After exploring the different MSE and R^2 values, it was observed that the best performance was achieved by employing the top 3 most important features, as evidenced in Table E.20.

Following this observation, the process proceeded with further manual parameter tuning aimed at enhancing the performance of the model and reducing overfitting. Parameters such as *hidden_layer_sizes*, *activation*, *solver*, *alpha*, *learning_rate*, *learning_rate_init*, and *max_iter* were modified to converge in an optimal solution. Each parameter was tested with different values to identify the combination yielding the best performance by getting the minimum MSE. Following the analysis, the parameter values corresponding to the highest scores were retained. This process was repeated for all parameters, culminating in the selection of the best model configuration. In this example, the optimal parameters were determined as follows: *hidden_layer_sizes* = (50,100,50), *alpha* = 0.01, *max_iter* = 100. Building on this determination, the Neural Network model was constructed using all features and the identified optimal parameters, as previously defined. This construction is visually represented in Table E.21. Then, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were calculated, as illustrated in Figure E.22. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Multi-Layer Perceptron with Grid Search

After exploring the different MSE and R^2 , it was observed that the best performance was achieved by employing the top 3 most important features, as evidenced in Table E.23. Following this observation, further parameter experimentation was conducted through the application of *GridSearchCV*, employing a predefined parameter space defined by me. This parameter space included variations of *hidden_layer_sizes*, *activation*, *solver*, *alpha*, and *learning_rate* to converge to an optimal solution, as illustrated in Table E.24 and Table E.25. Through this process, the optimal parameters were identified by selecting values that resulted in the highest scores. In this example, the optimal parameters were determined as follows: *activation* = 'tanh', *alpha* = 0.0001, *hidden_layer_sizes* = (25,25), *solver* = 'adam', *learning_rate* = 'invscaling'. Building on this determination, the MLP with Grid Search model was constructed, incorporating the aforementioned 3 significant features along with the identified optimal model parameters, as previously defined. This construction visually represented in Table E.26. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.27. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Support Vector Machine Regressor

After exploring the different MSE and R^2 values, it was observed that the best performance was achieved by employing the top 3 most important features, as evidenced in Table E.28. Following this observation, further manual parameter tuning was conducted to enhance the model performance and reduce overfitting. Parameters such as *C* (regularization parameter), *kernel*, *gamma*, *epsilon*, and *max_iter* were adjusted to converge on an optimal solution. Each parameter underwent test ing with different values to determine the combination yielding

the best performance. Following the analysis, the parameter values corresponding to the highest scores were retained and this process was repeated for all parameters, culminating in the selection of the best model configuration. In this example, the optimal parameters were determined as follows: $C = 10$, $\epsilon = 0.25$, $max_iter = 145$. Building upon this determination, the SVM model was constructed, incorporating the top 3 significant features along with the identified optimal model parameters, as defined previously. This construction is visually represented in Table E.29. Then, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.30. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

K-Nearest Neighbors

After exploring the different MSE and R^2 values, it was observed that the best performance was achieved by employing the top 3 most important features, as evidenced in Table E.31. Following this observation, further manual parameter tuning was conducted to enhance model performance and reduce overfitting. In this instance, the $n_neighbors$, $weights$, and $algorithm$ were modified to converge to an optimal solution. Each parameter was tested with various values to identify the combination yielding the best performance. After comprehensive analysis, it was found that the default parameters provided the optimal configuration for this example. Building upon this determination, the KNN model was constructed, incorporating the aforementioned 3 significant features along with the identified optimal model parameters, as previously defined. This construction is visually represented in Table E.32. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.33. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Ensemble Bagging Regressor

After exploring the different MSE and R^2 , it was observed that the best performance was achieved by employing the top 3 most important features, as evidenced in Table E.34. Following this observation, further manual parameter experimentation was conducted to enhance the model performance and reduce overfitting. Adjustments were made to parameters such as $base_estimator$, $n_estimators$, $max_samples$, $random_state$, $bootstrap$, $bootstrap_features$, $warm_start$, and oob_score . These parameters were collectively tested to converge on an optimal solution. Each parameter underwent testing with different values to determine the combination yielding the best performance. Following the analysis, the parameter values corresponding to the highest scores were retained and this process was repeated for all parameters. In this example, the optimal parameters were determined: $base_estimator = SVR_model_final$, $n_estimators = 50$, $warm_start = True$. Building on this determination, the Ensemble Bagging Regressor model was constructed using the

aforementioned 3 significant features along with the identified optimal model parameters, as previously defined. This construction is visually represented in Table E.35. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Figure E.36. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Ensemble Adaboost Regressor

After exploring the different MSE and R^2 , it was observed that the best performance was achieved by employing the top 3 most important features, as evidenced in Table E.37. Following this observation, the process proceeded with further manual parameter experimentation aimed at enhancing the model performance and reducing overfitting. Parameters such as *base_estimator*, *n_estimators*, and *learning_rate* were adjusted to converge to an optimal solution. Each parameter underwent testing with different values to determine the combination yielding the best performance. After thorough the analysis, the parameter values corresponding to the highest scores were retained and this iterative process was repeated for all parameters involved. In this example, the optimal parameters were determined: *n_estimators* = 59, *base_estimator* = *SVR_model_final*, *learning_rate* = 0.5. Building upon this determination, the Ensemble Adaboost Regressor model was constructed, incorporating the aforementioned 3 significant features along with the identified optimal model parameters, as previously defined. This is visually represented in Table E.38. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously calculated, as illustrated in Table E.39. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Ensemble Gradient Boosting Regressor

After exploring the different MSE and R^2 , it was observed that the best performance was achieved by employing the top 5 most important features, as evidenced in Table E.40. Following this observation, further manual parameter tuning was conducted to enhance the model performance and reduce overfitting. Parameters such as *loss*, *learning_rate*, *random_state*, *n_estimators*, *subsample*, *criterion*, *max_depth*, *min_samples_leaf*, *min_samples_split*, and *min_impurity_decrease* were modified to converge on an optimal solution. Each parameter underwent testing with different values to identify the combination yielding the best performance. After thorough analysis, the parameter values corresponding to the highest scores were retained and this iterative process was repeated for all parameters involved. In this example, the optimal parameters were determined: *learning_rate* = 1.0, *random_state* = 38, *n_estimators* = 67. Building on this determination, the Ensemble Gradient Boosting Regressor model was constructed, incorporating the aforementioned 5 significant features along with the identified optimal model parameters, as previously defined. This construction is visually represented in Table E.41. Then, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics

were meticulously calculated, as illustrated in Table E.42. Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Ensemble Voting Regressor

The initial step in constructing the final Voting Regressor model involved creating 13 estimators, each comprising distinct models developed previously, as observed in Figure E.1. Following this, the models within the estimators were trained, and subsequently, their performances across different sets of variables were evaluated. After analyzing of the results, I made a deliberate decision to utilize the following estimator: *estimators_10* = *[('ar', AR_model_final), ('svr', SVR_model_final)]*. It was then observed that the best performance was achieved by employing the top 3 most important features, as evidenced in Table E.43. Following this observation, further manual parameter experimentation was conducted to enhance the model performance and reduce overfitting. In this instance, adjustments were made to the *weights* to converge on an optimal solution. Each parameter underwent testing with different values in order to determine the combination yielding the best performance. After thorough analysis, the parameter values corresponding to the highest scores were retained and this iterative process was repeated for all parameters involved. In this example, the optimal parameters were determined: *estimators* = *estimators*. Building upon this determination, the Ensemble Voting Regressor model was constructed, incorporating the aforementioned 3 significant features along with the identified optimal model parameters, as previously defined. This construction is visually represented in Table E.44. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously computed, as illustrated in Table E.45. Finally, the MSE and the score R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Ensemble Stacking Regressor

The initial stage in constructing the final Stacking Regressor model involved creating 15 estimators, each comprising distinct models developed previously, as observed in Figure E.2. Following this, the models were trained, and their performances evaluated across different sets of variables. After analyzing the results, the optimal number of features for each estimator was recorded and, afterward, the final estimator was updated for each previously selected model. In this process, all previously tested estimators were thoroughly explored. After a meticulous examination of each result, I determined to use the SVMmodel previously developed as the final estimator for the final model incorporating the 3 most significant features. The chosen estimator configuration is as follows: *estimators_10* = *[('ar', AR_model_final), ('svr', SVR_model_final)]*. Building upon this determination, the Ensemble Stacking Regressor model was constructed, incorporating the aforementioned 3 significant features along with the identified optimal model parameters, as previously defined. This is visually represented in Table E.46. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics

were meticulously computed, as illustrated in Table E.47. Finally, the MSE and the score R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Extreme Gradient Boosting Regressor

After evaluating the different MSE and R^2 , the best performance was achieved by using the top 3 most significant features, as evidenced in Table E.48. Following this, further parameter experimentation was conducted through *GridSearchCV*, using a defined parameter space that included different values for *max_depth*, *learning_rate*, *n_estimators*, *subsample*, and *colsample_bytree* to find the optimal solution. Through this process, the optimal parameters were identified, and this can be observed in Table E.49 and Table E.50. After thorough analysis, the parameter values corresponding to the highest scores were retained. The optimal parameters determined in this instance were: *colsample_bytree* = 0.2, *learning_rate* = 0.01, *max_depth* = 3, *n_estimators* = 200, *subsample* = 0.5. Building upon this determination, the Ensemble Extreme Gradient Boosting Regressor model was constructed, incorporating the aforementioned 3 significant features along with the identified optimal model parameters, as previously defined. This is visually represented in Table E.51. A 10-fold cross-validation approach was then employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously computed, as illustrated in Table E.52. Finally, the MSE and the score R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

TPOT

Initially, to observe the impact of each parameter adjustment and comprehend its effect on the optimization process, a simple model with default parameter values provided by TPOT was created, trained, and evaluated, except for the number of generations, which was set to 5. The evaluation involved assessing the model's MSE and R^2 across different sets with varying numbers of variables. Upon analyzing the results, the best performance was achieved by employing the set of all features, as evidenced in Table E.53. Following this, the process proceeded with further manual parameter tuning aimed at enhancing the model performance and reducing underfitting. Parameters such as *population_size*, *mutation_rate*, *crossover_rate*, *cv*, and *random_state* were modified to converge to an optimal solution. Each parameter was tested with different values to determine the combination yielding the lowest MSE. After analysis, the parameter values corresponding to the highest scores were retained. This process was repeated for all parameters, culminating in the selection of the best pipeline with optimal parameters. In this example, the optimal parameters were determined as follows: *generations* = 5, *verbosity* = 2, *population_size* = 95, *mutation_rate* = 0.1, *crossover_rate* = 0.3, *cv* = 3, *random_state* = 123. Building upon this determination, the final model was constructed, incorporating the aforementioned all features along with the identified optimal model parameters, as previously defined. This is visually represented in Figure E.3. Subsequently, a 10-fold cross-validation approach was employed. Following the model construction, the MAE, MSE, RMSE, R^2 , and MAPE metrics were meticulously

computed, as illustrated in Table E.54. Furthermore, the best model for this problem was determined to be RidgeCV (MinMaxScaler(VarianceThreshold(SGDRegressor(input_matrix, alpha=0.0, eta0=1.0, fit_intercept=False, l1_ratio=0.0, learning_rate=invscaling, loss=epsilon_insensitive, penalty=elasticnet, power_t=0.1), threshold=0.0005)). Finally, the MSE and R^2 metrics of the model were recorded within a dataframe which aggregates the best results obtained from the different models.

Regarding the final results obtained

In the regression problem, while predicting the time to recur:

After applying all developed models, a comprehensive comparison was conducted to assess their performance. This evaluation involved utilizing a range of ML evaluation metrics such as MAE, MSE, RMSE, R^2 , and MAPE analysis. These metrics were employed to thoroughly compare and analyze the results obtained from the different models. As detailed throughout the document, a dataframe which aggregates the best results obtained from the different models was created to represent the MSE and R^2 of all models. In this way, Table E.53 visually portray this comparison table. The analysis revealed that the models exhibited suboptimal results and discrepancies between training and validation datasets. These not-ideal values may be attributed to the relatively small size of the dataset. Among the models, the Gradient Boosting Regressor stood out for its higher values and demonstrated very similar results between training and validation sets. Even though it presents higher values, it is crucial to emphasize that this does not necessarily imply greatness or high performance; the results are somewhat lacking.

More precisely, the obtained values for the MAE and MSE are considered to be quite high, indicating potential limitations in the accuracy of the models. The MAE, depicted in Figure E.4, serves as a common metric for evaluating predictive model performance by measuring the average magnitude of errors between predicted and actual values. The higher error values observed are not ideal for predicting the time of recurrence, highlighting the challenges in achieving accurate predictions.

In Figure E.5, the MSE of each model is depicted. This metric evaluates the accuracy of a predictive model by measuring the average of the squared differences between predicted and actual values. Similar to the MAE values, the obtained MSE values are considered quite high, suggesting that the models may not perform accurately. In addition, it is noteworthy to mention that the Gradient Boost model exhibited the poorest performance, yielding very unsatisfactory results.

In Figure E.6, RMSE is shown, serving as another metric used to evaluate the accuracy of a predictive model. RMSE is derived from the MSE and provides a measure of the average magnitude of the prediction errors, offering further insight into the model's performance. As

expected, RMSE exhibits results similar results to MSE, reflecting the overall accuracy and performance of the models.

Figure E.7 displays the R^2 values of each model. This metric evaluates the goodness of fit of a regression model by quantifying the proportion of the variance in the dependent variable that is explained by the independent variables in the model. It is important to highlight that negative values can be obtained which may indicate that the model performs worse than a horizontal line fitting the data, which might suggest overfitting or poor model performance. In the context of this analysis, where the dataset is relatively small, negative R^2 values should be carefully interpreted as they may indicate challenges such as overfitting due to the limited amount of data available for training the models.

Lastly, Figure E.8 displays the MAPE of each model, which is a metric for evaluating the accuracy of a predictive model. This measure quantifies the average percentage difference between the predictive and the actual values. Notably, MAPE is particularly relevant in predicting the time of recurrence in BC cases, as it provides insights into the accuracy of prognostic predictions. In this instance, the overall results indicate that the models did not perform optimally, with TPOT demonstrating the best performance by yielding a lower MAPE compared to other algorithms. Lower MAPE values indicate that TPOT's predictions are closer to the actual values on average.

In conclusion, when attempting to predict the recurrence time for recurrent cases and the disease-free interval for non-recurrent cases, which represents a regression problem, the results reveal discrepancies between the training and validation datasets across various models. Despite employing different evaluation metrics, the results consistently fell short of the anticipated level of performance.

5. CONCLUSIONS AND FUTURE WORK

Annex 3 present a Gantt chart to illustrate the work done. The thesis development process began with formulating the thesis theme and outlining the initial trajectory. Subsequently, a succinct one-page summary was crafted, and this was followed by an extensive literature review. Following these preparatory steps, the dataset was explored, and a comprehensive understanding of the data was developed. Concurrently, various models and algorithms were studied. Subsequent actions included data preparation, application of machine learning (ML) techniques, and the implementation of genetic programming (GP) algorithms, culminating in the creation of a concise methodology document. Once the methodology was established, experimental settings were defined, and results were interpreted, leading to the formulation of conclusions. Subsequently, a concise document covering experimental settings, results, and discussions was compiled. Finally, attention was turned to developing and composing the abstract, introduction, and conclusion sections of the thesis. Regarding final findings:

In Diagnosis

Regarding the final findings discovered in this process:

In this study, the exploration of the data was followed by thorough analysis, preprocessing, preparation, and feature selection. Subsequently, various ML and GP algorithms were applied to develop classification models aimed at predicting and distinguishing between benign from malignant breast cancer (BC) cases. After conducting these analyses, the results were carefully evaluated to identify the most reliable and accurate models for addressing the problem at hand.

The following key findings emerged:

1. ML and GP algorithms have demonstrated significant and promising accuracy in enhancing BC diagnosis, indicating their potential for improving clinical outcomes.
2. The developed models consistently outperformed traditional and simpler methods, highlighting their efficacy in accurately diagnosing BC cases.
3. Among the models evaluated, the MLP, Voting Classifier, and Stacking Classifier exhibited the highest performance metrics, with very similar training and validation results. These models show promise for clinical implementation due to their robust performance and reliability.

These findings underscore the potential of advanced ML techniques in improving BC diagnosis and suggest avenues for further research and clinical application in this critical domain.

In Prognosis

Regarding the final findings discovered in the classification problem:

In this study, the classification problem involved predicting outcomes as recurrent or non-recurrent. The process began with thorough data exploration, followed by preparatory steps and feature selection. Various models with different parameters were then applied to find the optimal classification model for distinguishing between recurrent and non-recurrent outcomes. Following this practical phase, the results were analyzed to identify significant differences between all algorithms and to identify the more reliable and accurate models for addressing the problem at hand.

The following key findings emerged:

1. The ML and GP algorithms yielded suboptimal results, which may be attributed to the small size of the dataset. The limited data availability posed challenges in training robust models with sufficient predictive power.
2. Significant improvements in prognostic accuracy and evaluation were difficult to achieve due to the methodologies applied, which encountered inherent limitations. Despite rigorous efforts, challenges persisted in accurately assessing and enhancing the predictive capabilities of the models.
3. Among the evaluated models, the Voting Classifier demonstrated the most promising performance, characterized by higher values and consistent training and validation results. Its robust performance suggests its potential utility in clinical applications.
4. TPOT exhibited highly positive outcomes comparable to those achieved by conventional ML algorithms, particularly in terms of AUC performance. These findings highlight TPOT's effectiveness in automating the model selection and optimization process, contributing to enhanced diagnostic accuracy and efficiency.

Regarding the final findings discovered in the regression problem:

On the other hand, in addressing the regression problem to predict the time to recurrence, a similar methodology was employed. The process commenced with data exploration, followed by preparatory steps and feature selection based on the analysis. Subsequently, various models with diverse parameters were applied to identify the most effective regression model for predicting the time to recurrence. Following this practical phase, the results were scrutinized to discern the more reliable and accurate models for addressing the problem at hand.

The following key findings emerged:

1. ML and GP algorithms have exhibited suboptimal results and discrepancies between train and validation datasets. These values may be attributed to the small size of the dataset, which hindered the models' ability to generalize effectively.
2. Challenges persisted in achieving significant improvements in prognostic accuracy and evaluation due to the methodologies applied. Despite diligent efforts, limitations

inherent in the approaches utilized impeded substantial enhancements in predictive performance.

3. Among the evaluated models, the Gradient Boosting Regressor stood out for its higher values and demonstrated consistency between training and validation datasets. However, it's noteworthy to mention that while it exhibits higher values, this doesn't necessarily imply excellence; rather, the results are somewhat lacking in robustness and generalizability.

These findings underscore the potential of advanced ML techniques in improving BC prognosis and suggest avenues for further research and clinical application in this critical domain.

5.1. LIMITATIONS

The limitations faced during in the process were outlined as follows:

- **Lack of Reliable Literature Review:** The study encountered challenges in accessing comprehensive and reliable literature relevant to the research topic. Despite extensive efforts to gather pertinent sources, the scarcity of up-to-date and credible literature limited the depth of contextual understanding and potentially influenced the comprehensiveness of the analysis.
- **Data Quality, Size & Availability:** The availability of high-quality and relevant datasets proved to be a significant constraint in the research process. The scarcity of data within the specific domain under study hindered the robustness and generalizability of the findings. Variations in data quality, size, and relevance posed challenges to conducting comprehensive analyses, ultimately impacting the validity of the results.
- **Computational Complexity vs. Resource Requirements:** The computational complexity associated with the methodologies employed posed practical challenges, particularly in terms of resource requirements. Execution times and computational demands were significant considerations, especially when dealing with large datasets or intricate algorithms. Resource constraints, including limitations in computing power and memory, affected the efficiency and scalability of the analytical processes.
- **Exhaustive Algorithm Selection and Tuning:** The process of selecting and tuning algorithms proved to be demanding and time-consuming. Despite meticulous efforts to identify suitable algorithms, the complexity of the task and the need for thorough tuning to optimize performance presented challenges. The exhaustive nature of algorithm selection and tuning potentially impacted the efficiency and effectiveness of the analytical procedures.

5.2. FUTURE WORK

To further advance this research, several opportunities can be pursued. Firstly, future efforts should focus on refining the developed models by training them on larger and more diverse datasets. This can enhance the accuracy and robustness of the models, improving their generalizability and performance in real-world applications. Secondly, another promising direction is to explore the integration of emerging technologies. Incorporating advancements in advanced data analytics, deep learning can provide new dimensions of data and more sophisticated analytical tools. This integration has the potential to significantly improve the diagnosis and prognosis of BC. By addressing these areas, the research can be significantly extended, offering even more reliable, rich, and efficient solutions in the fight against BC.

5.3. FINAL THOUGHTS

In the introduction of this final document, a central question emerged: **How can ML and GP be used effectively to enhance the diagnosis and prognosis of BC?** Throughout this document, theoretical and practical processes have been explored with the aim of answering this question comprehensively.

As previously mentioned, BC remains one of the most lethal cancers for women. However, the emergence of ML and GP in the medical field presents powerful and promising solutions for enhancing its diagnosis and prognosis. The integration of medical expertise and AI holds transformative potential, poised to revolutionize the approach to diagnosing and prognosing BC outcomes. This synergy represents a significant advancement in optimizing, facilitating, accelerating and refining both diagnostic and prognostic procedures within healthcare.

This thesis focuses on the implementation of ML and GP algorithms. In this context, several ML algorithms were employed, and the results demonstrate significant success in distinguishing between malignant and benign breast cancer. Although the results were less promising in predicting recurrence and the time to recurrence for recurrent cases, as well as disease-free time for non-recurrent cases, prognosis capabilities of these methods still showed potential benefits.

As presented by all document, integrating the abovementioned two fields, it is possible to achieve accurate and rapid diagnoses by identifying patterns and features that indicate cancerous conditions efficiently. Additionally, ML and GP can provide reliable, easy, and accurate results without the need for extensive procedures. This approach facilitates personalized treatments plans, thereby improving the quality of care for each patient.

Using ML and GP, we can improve diagnostic accuracy, speed up the diagnostic process and adapt treatments to each patient, thus improving outcomes and quality of life. Additionally, it

is possible to predict with some accuracy whether a particular patient will develop cancer again and forecast the likelihood of recurrence.

In conclusion, the application and the integration of ML and GP in BC holds great promise for transforming and enhancing prognostic processes. By leveraging these advanced techniques, it was possible to develop more accurate and reliable methods for diagnosis and prognosis. The potential of these technologies to revolutionize healthcare underscores the importance of continued research and innovation in this area.

BIBLIOGRAPHICAL REFERENCES

- Bozorgi, M. (2018). *Application of Machine Learning in Cancer Research* (Publication No. 10839508) [Doctoral dissertation, University of Nevada, Las Vegas]. ProQuest Dissertations & Theses Global. <https://doi.org/10.34917/14139867>
- Donga, H. (2022). *Comparing Machine Learning Models: For Diagnosis of Breast Cancer*. [Unpublished bachelor dissertation]. Blekinge Institute of Technology. <https://www.diva-portal.org/smash/get/diva2:1679145/FULLTEXT02>
- Ferroni, P., Zanzotto, F. M., Riondino, S., Scarpato, N., Guadagni, F., & Roselli, M. (2019). Breast Cancer Prognosis Using a Machine Learning Approach. *Cancers*, 11(3), 328. <https://doi.org/10.3390/cancers11030328>
- Hammi, A. (2022). *Diagnosing breast cancer using Machine Learning methods*. [Unpublished bachelor dissertation]. Vrije Universiteit Amsterdam.
- Kottur, S. (2023). *Breast Cancer Classification* [Master dissertation, California State University]. ScholarWorks Repository. <https://scholarworks.calstate.edu/concern/theses/8623j506r>
- Mangasarian, O., Street, N., & Wolberg, W. (1970). Breast Cancer Diagnosis and Prognosis Via Linear Programming. *Operations Research*, 43(4), 548-725. <https://doi.org/10.1287/opre.43.4.570>
- Mohammed, S. A., Darrab, S., Noaman, S. A., & Saake, G. (2020). Analysis of Breast Cancer Detection Using Different Machine Learning Techniques. In Y. Tan, Y. Shi, & M. Tuba (Eds.), *Data Mining and Big Data* (pp. 108–117). Springer. https://doi.org/10.1007/978-981-15-7205-0_10
- Naji, M. A., Filali, S. E., Aarika, K., Benlahmar, E. H., Abdelouhahid, R. A., & Debauche, O. (2021). Machine Learning Algorithms For Breast Cancer Prediction And Diagnosis. *Procedia Computer Science*, 191, 487–492. <https://doi.org/10.1016/j.procs.2021.07.062>
- Naseem, U., Rashid, J., Ali, L., Kim, J., Haq, Q. E. U., Awan, M. J., & Imran, M. (2022). An Automatic Detection of Breast Cancer Diagnosis and Prognosis Based on Machine Learning Using Ensemble of Classifiers. *IEEE Access*, 10, 78242–78252. <https://doi.org/10.1109/ACCESS.2022.3174599>
- Nassif, A. B., Talib, M. A., Nasir, Q., Afadar, Y., & Elgendy, O. (2022). Breast cancer detection using artificial intelligence techniques: A systematic literature review. *Artificial Intelligence in Medicine*, 127, 102276. <https://doi.org/10.1016/j.artmed.2022.102276>
- Omondiagbe, D. A., Veeramani, S., & Sidhu, A. S. (2019). Machine Learning Classification Techniques for Breast Cancer Diagnosis. *IOP Conference Series: Materials Science and Engineering*, 495, 012033. <https://doi.org/10.1088/1757-899X/495/1/012033>

- Osareh, A., & Shadgar, B. (2010). Machine learning techniques to diagnose breast cancer. *2010 5th International Symposium on Health Informatics and Bioinformatics*, 114–120. <https://doi.org/10.1109/HIBIT.2010.5478895>
- Rabiei, R., Ayyoubzadeh, S. M., Sohrabei, S., Esmaeili, M., & Atashi, A. (2022). Prediction of Breast Cancer using Machine Learning Approaches. *Journal of Biomedical Physics & Engineering*, 12(3), 297–308. <https://doi.org/10.31661/jbpe.v0i0.2109-1403>
- Sharma, A., Kulshrestha, S., & Daniel, S. (2017). Machine learning approaches for breast cancer diagnosis and prognosis. *2017 International Conference on Soft Computing and Its Engineering Applications (icSoftComp)*, 1–5. <https://doi.org/10.1109/ICSOFTCOMP.2017.8280082>
- Shubair, R. (2016). Comparative Study of Machine Learning Algorithms for Breast Cancer Detection and Diagnosis. *2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA)*, 1-4. <https://doi.org/10.1109/ICEDSA.2016.7818560>
- Street, N., Wolberg, W., & Mangasarian, O. (1993). Nuclear Feature Extraction For Breast Tumor Diagnosis. *Biomedical Image Processing and Biomedical Visualization, 1905*. <https://doi.org/10.1117/12.148698>
- Vanneschi, L., Farinaccio, A., Mauri, G., Antoniotti, M., Provero, P., & Giacobini, M. (2011). A comparison of machine learning techniques for survival prediction in breast cancer. *BioData Mining*, 4(1), 12. <https://doi.org/10.1186/1756-0381-4-12>
- Wolberg, W. H., Street, W. N., Heisey, D. M., & Mangasarian, O. L. (1995). Computer-derived nuclear features distinguish malignant from benign breast cytology. *Human Pathology*, 26(7), 792–796. [https://doi.org/10.1016/0046-8177\(95\)90229-5](https://doi.org/10.1016/0046-8177(95)90229-5)
- Wolberg, W. H., Street, W. N., Heisey, D. M., & Mangasarian, O. L. (1995). Computer-Derived Nuclear «Grade» and Breast Cancer Prognosis. *Analytical and Quantitative Cytology and Histology*, 17(4), 257-264.
- Wolberg, W. H., Street, W. N., Heisey, D. M., & Mangasarian, O. L. (1995). Computerized Breast Cancer Diagnosis and Prognosis From Fine-Needle Aspirates. *Archives of Surgery*, 130(5), 511–516. <https://doi.org/10.1001/archsurg.1995.01430050061010>
- Wolberg, W. H., Street, W. N., & Mangasarian, O. L. (1994). Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates. *Cancer Letters*, 77(2–3), 163–171. [https://doi.org/10.1016/0304-3835\(94\)90099-x](https://doi.org/10.1016/0304-3835(94)90099-x)
- Wolberg, W. H., Street, W. N., & Mangasarian, O. L. (1995). Image Analysis and Machine Learning Applied to Breast Cancer Diagnosis and Prognosis. *Analytical and Quantitative Cytology and Histology*, 17(2), 77-87.

Yu, M. (2017). *Breast cancer prediction using machine learning algorithm* [Master dissertation, University of Texas at Austin]. UT Electronic Theses and Dissertations. <http://hdl.handle.net/2152/62802>

Yue, W., Wang, Z., Chen, H., Payne, A., & Liu, X. (2018). Machine Learning with Applications in Breast Cancer Diagnosis and Prognosis. *Designs*, 2(2), Article 2. <https://doi.org/10.3390/designs2020013>

APPENDIX A

A.1 DIAGNOSTIC METHODOLOGY – BUSINESS UNDERSTANDING

Table A.1 – Real-Valued features

Feature	Detail
radius	Mean of distances from center to points on the perimeter
texture	Standard deviation of gray-scale values
perimeter	Mean of total distance around the outside of nuclear cell shape
area	Amount of surface covered by the nuclear cell shape
smoothness	Local variation in radius lengths
compactness	Ration of the square of the perimeter to the area, minus 1
concavity	Severity of concave portions of the contour
concave points	Number of concave portions of the contour
symmetry	The longest chord passing the through the center
fractal dimension	"Coastline approximation" - 1

A.2 DIAGNOSTIC METHODOLOGY – DATA UNDERSTANDING

diagnosis	2	compactness_se	541
radius_mean	456	concavity_se	533
texture_mean	479	concave_points_se	507
perimeter_mean	522	symmetry_se	498
area_mean	539	fractal_dimension_se	545
smoothness_mean	474	radius_worst	457
compactness_mean	537	texture_worst	511
concavity_mean	537	perimeter_worst	514
concave_points_mean	542	area_worst	544
symmetry_mean	432	smoothness_worst	411
fractal_dimension_mean	499	compactness_worst	529
radius_se	540	concavity_worst	539
texture_se	519	concave_points_worst	492
perimeter_se	533	symmetry_worst	500
area_se	528	fractal_dimension_worst	535
smoothness_se	547	dtype: int64	

Figure A.1 – Distinct observations in WDBC dataset

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
diagnosis	569	2	B	357	NaN	NaN	NaN	NaN	NaN	NaN	NaN
radius_mean	569.0	NaN	NaN	NaN	14.127292	3.524049	6.981	11.7	13.37	15.78	28.11
texture_mean	569.0	NaN	NaN	NaN	19.289649	4.301036	9.71	16.17	18.84	21.8	39.28
perimeter_mean	569.0	NaN	NaN	NaN	91.969033	24.298981	43.79	75.17	86.24	104.1	188.5
area_mean	569.0	NaN	NaN	NaN	654.889104	351.914129	143.5	420.3	551.1	782.7	2501.0
smoothness_mean	569.0	NaN	NaN	NaN	0.09636	0.014064	0.05263	0.08637	0.09587	0.1053	0.1634
compactness_mean	569.0	NaN	NaN	NaN	0.104341	0.052813	0.01938	0.06492	0.09263	0.1304	0.3454
concavity_mean	569.0	NaN	NaN	NaN	0.088799	0.07972	0.0	0.02956	0.06154	0.1307	0.4268
concave_points_mean	569.0	NaN	NaN	NaN	0.048919	0.038803	0.0	0.02031	0.0335	0.074	0.2012
symmetry_mean	569.0	NaN	NaN	NaN	0.181162	0.027414	0.106	0.1619	0.1792	0.1957	0.304
fractal_dimension_mean	569.0	NaN	NaN	NaN	0.062798	0.00706	0.04996	0.0577	0.06154	0.06612	0.09744
radius_se	569.0	NaN	NaN	NaN	0.405172	0.277313	0.1115	0.2324	0.3242	0.4789	2.873
texture_se	569.0	NaN	NaN	NaN	1.216853	0.551648	0.3602	0.8339	1.108	1.474	4.885
perimeter_se	569.0	NaN	NaN	NaN	2.866059	2.021855	0.757	1.606	2.287	3.357	21.98
area_se	569.0	NaN	NaN	NaN	40.337079	45.491006	6.802	17.85	24.53	45.19	542.2
smoothness_se	569.0	NaN	NaN	NaN	0.007041	0.003003	0.001713	0.005169	0.00638	0.008146	0.03113
compactness_se	569.0	NaN	NaN	NaN	0.025478	0.017908	0.002252	0.01308	0.02045	0.03245	0.1354
concavity_se	569.0	NaN	NaN	NaN	0.031894	0.030186	0.0	0.01509	0.02589	0.04205	0.396
concave_points_se	569.0	NaN	NaN	NaN	0.011796	0.00617	0.0	0.007638	0.01093	0.01471	0.05279
symmetry_se	569.0	NaN	NaN	NaN	0.020542	0.008266	0.007882	0.01516	0.01873	0.02348	0.07895
fractal_dimension_se	569.0	NaN	NaN	NaN	0.003795	0.002646	0.000895	0.002248	0.003187	0.004558	0.02984
radius_worst	569.0	NaN	NaN	NaN	16.26919	4.833242	7.93	13.01	14.97	18.79	36.04
texture_worst	569.0	NaN	NaN	NaN	25.677223	6.146258	12.02	21.08	25.41	29.72	49.54
perimeter_worst	569.0	NaN	NaN	NaN	107.261213	33.602542	50.41	84.11	97.66	125.4	251.2
area_worst	569.0	NaN	NaN	NaN	880.583128	569.356993	185.2	515.3	686.5	1084.0	4254.0
smoothness_worst	569.0	NaN	NaN	NaN	0.132369	0.022832	0.07117	0.1166	0.1313	0.146	0.2226
compactness_worst	569.0	NaN	NaN	NaN	0.254265	0.157336	0.02729	0.1472	0.2119	0.3391	1.058
concavity_worst	569.0	NaN	NaN	NaN	0.272188	0.208624	0.0	0.1145	0.2267	0.3829	1.252
concave_points_worst	569.0	NaN	NaN	NaN	0.114606	0.065732	0.0	0.06493	0.09993	0.1614	0.291
symmetry_worst	569.0	NaN	NaN	NaN	0.290076	0.061867	0.1565	0.2504	0.2822	0.3179	0.6638
fractal_dimension_worst	569.0	NaN	NaN	NaN	0.083946	0.018061	0.05504	0.07146	0.08004	0.09208	0.2075

Figure A.2 – Descriptive statistics in WDBC dataset

#	Column	Non-Null Count	Dtype
0	diagnosis	569 non-null	object
1	radius_mean	569 non-null	float64
2	texture_mean	569 non-null	float64
3	perimeter_mean	569 non-null	float64
4	area_mean	569 non-null	float64
5	smoothness_mean	569 non-null	float64
6	compactness_mean	569 non-null	float64
7	concavity_mean	569 non-null	float64
8	concave_points_mean	569 non-null	float64
9	symmetry_mean	569 non-null	float64
10	fractal_dimension_mean	569 non-null	float64
11	radius_se	569 non-null	float64
12	texture_se	569 non-null	float64
13	perimeter_se	569 non-null	float64
14	area_se	569 non-null	float64
15	smoothness_se	569 non-null	float64
16	compactness_se	569 non-null	float64
17	concavity_se	569 non-null	float64
18	concave_points_se	569 non-null	float64
19	symmetry_se	569 non-null	float64
20	fractal_dimension_se	569 non-null	float64
21	radius_worst	569 non-null	float64
22	texture_worst	569 non-null	float64
23	perimeter_worst	569 non-null	float64
24	area_worst	569 non-null	float64
25	smoothness_worst	569 non-null	float64
26	compactness_worst	569 non-null	float64
27	concavity_worst	569 non-null	float64
28	concave_points_worst	569 non-null	float64
29	symmetry_worst	569 non-null	float64
30	fractal_dimension_worst	569 non-null	float64

dtypes: float64(30), object(1)
memory usage: 158.4+ KB

Figure A.3 – Types of the columns in WDBC dataset

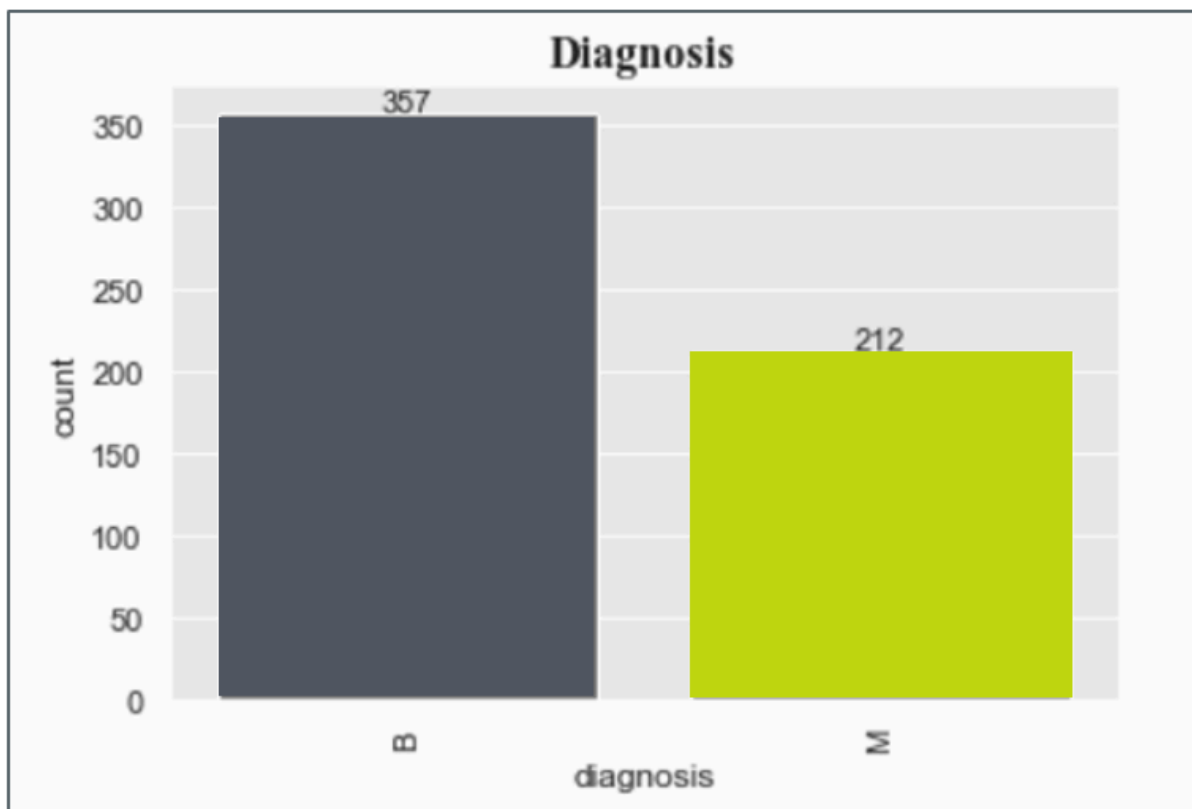


Figure A.4 – Number of people with different breast cancer diagnoses

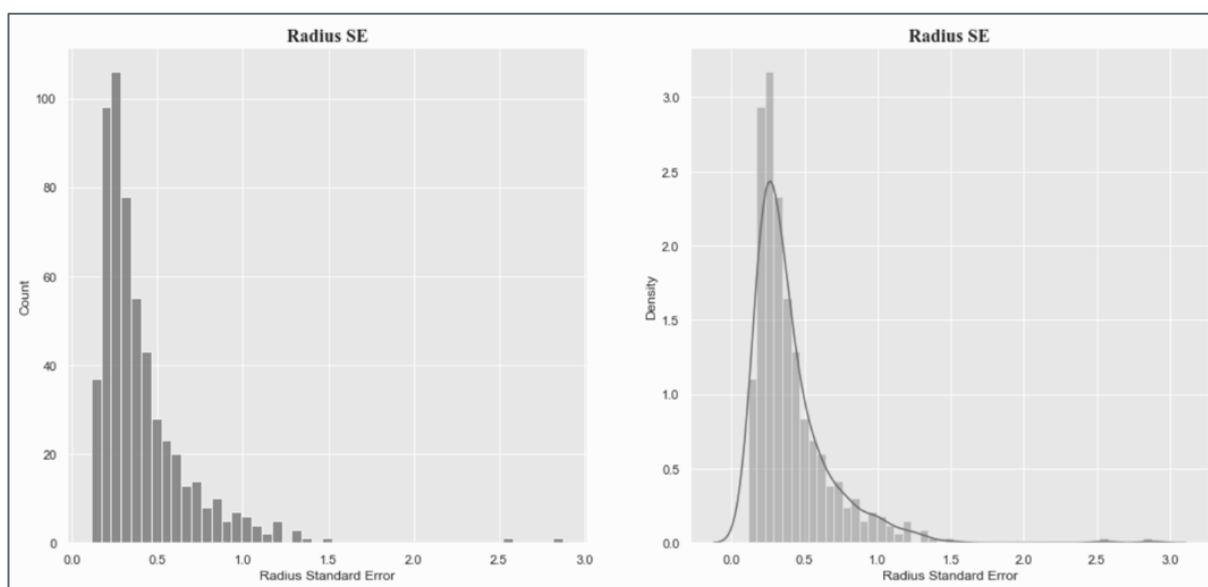


Figure A.5 – Radius SE distribution – histogram and displot

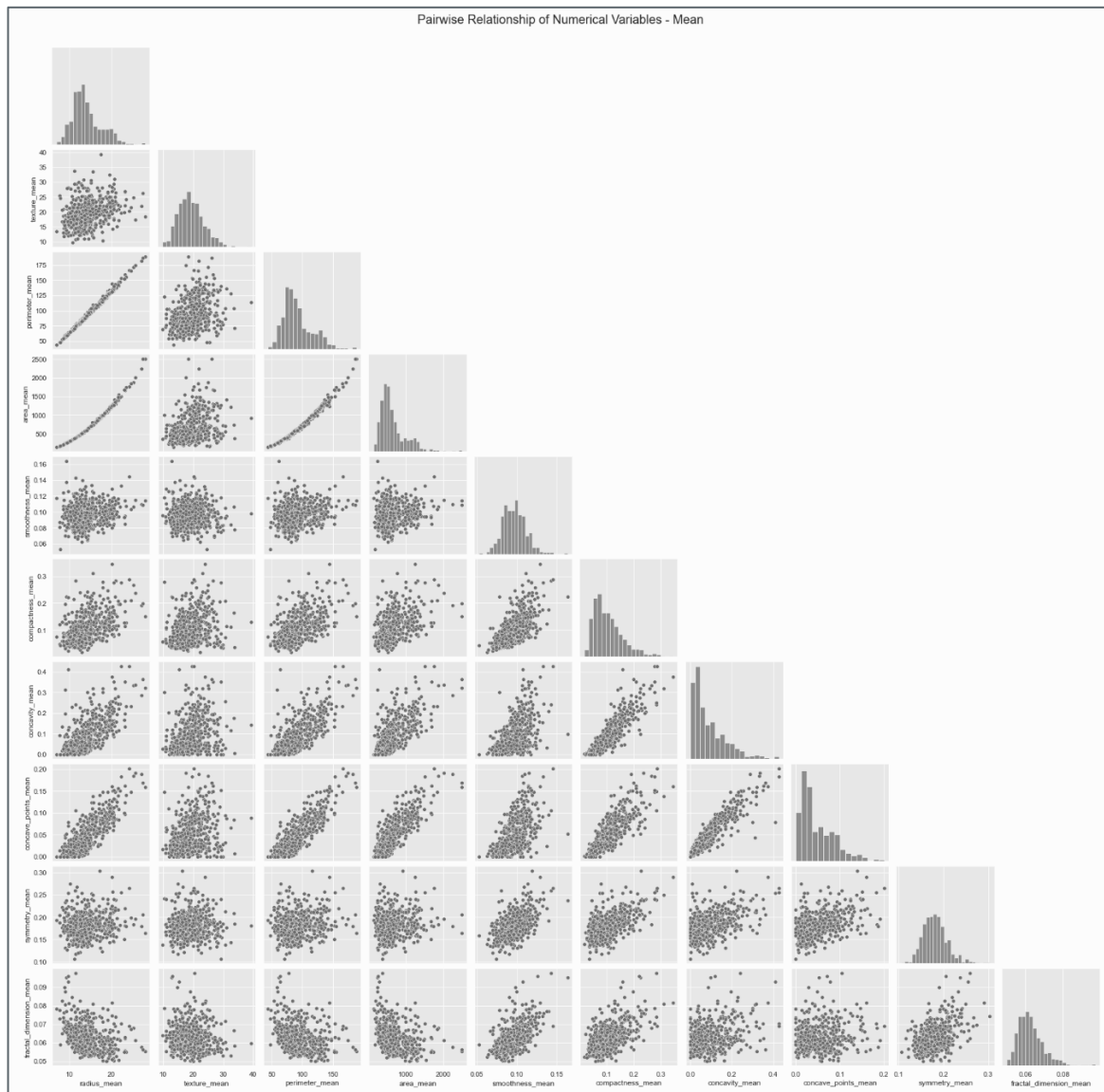


Figure A.6 – Relationship between mean numerical variable

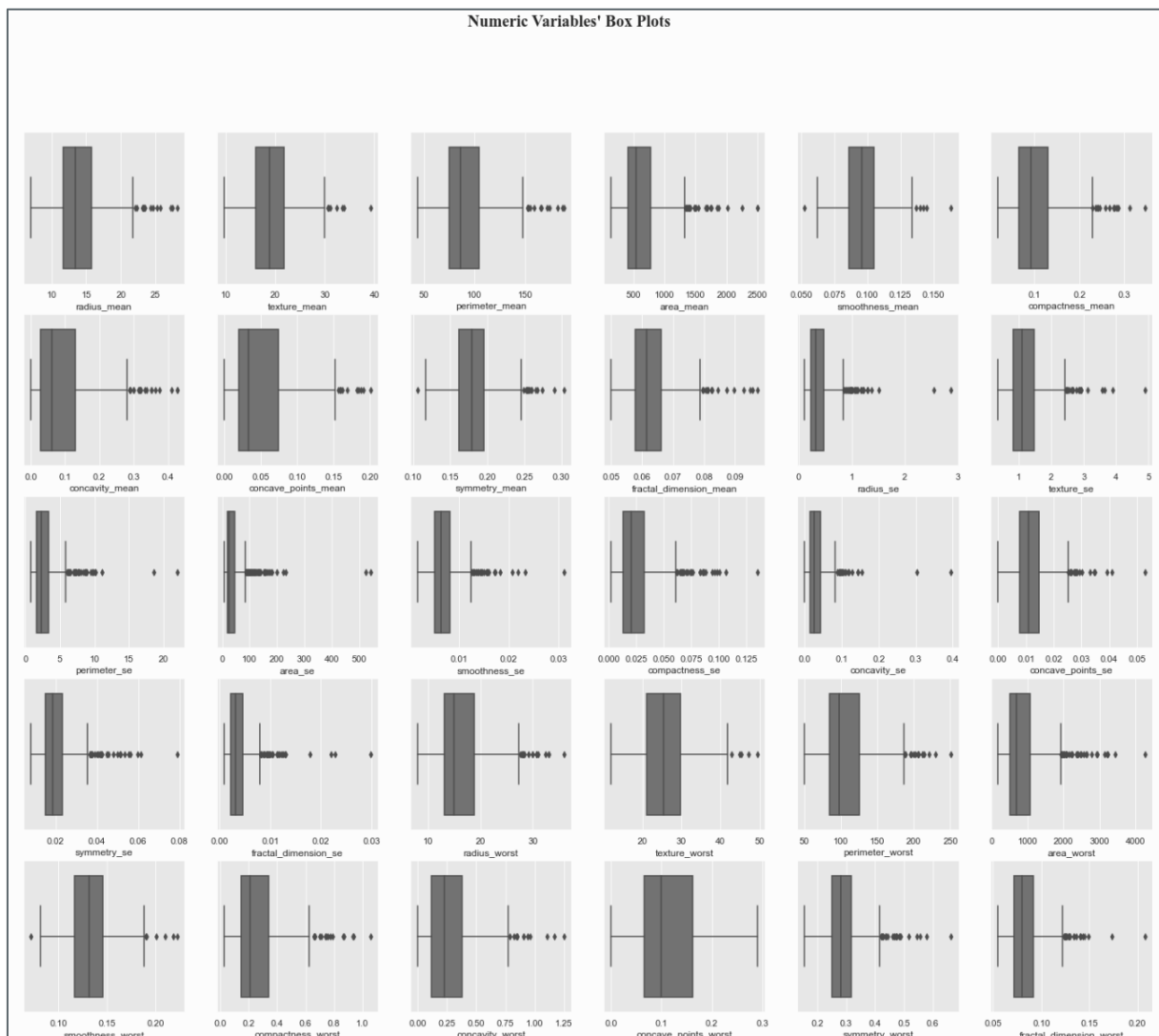


Figure A.7 – Relationship between mean numerical variable

A.3 DIAGNOSTIC METHODOLOGY – DATA PREPROCESSING

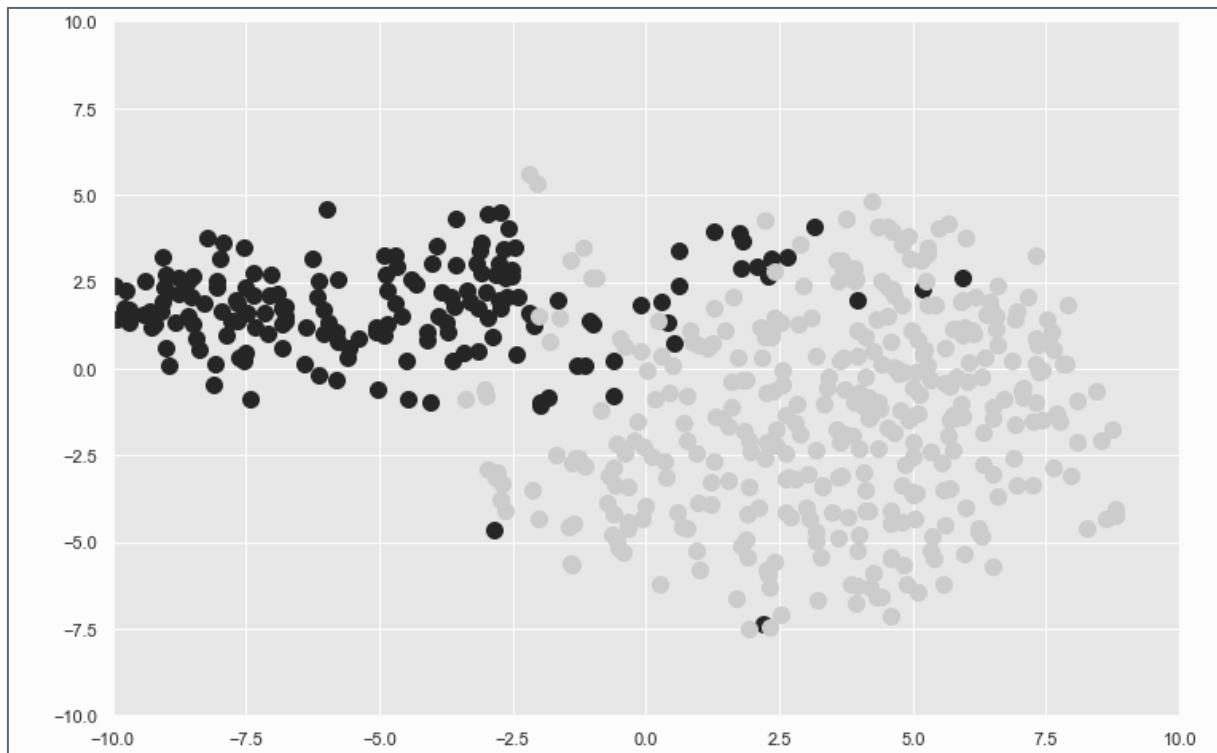


Figure A.8 – Imbalanced Data

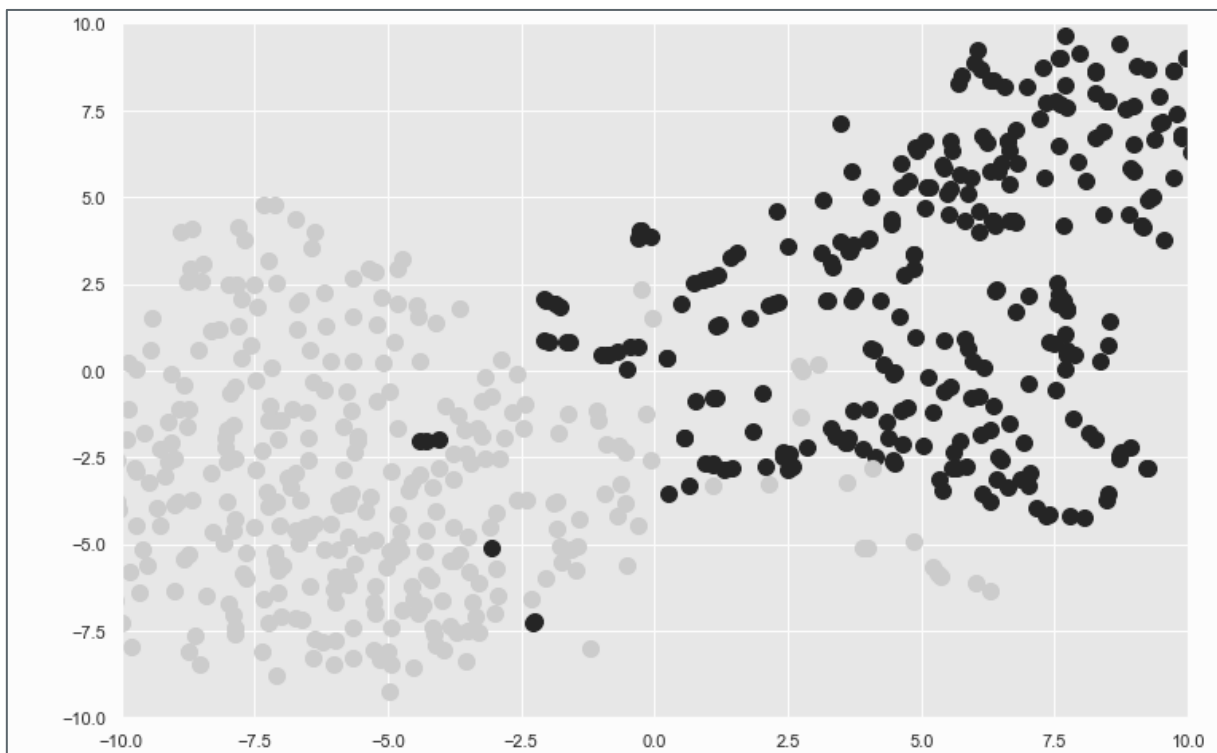


Figure A.9 – Balanced Data

	Pearson	Spearman	Kendall	Univariate Variables	ANOVA	Mutual Information	RFE	Lasso	Elastic Net	PCA	...	MSE	MAE	FRIEDMAN	AdaBoost	Randc Fore
smoothness_mean	0	0	1	0	0	0	0	0	0	0	...	0	1	0	1	
compactness_worst	0	0	0	1	0	1	0	0	0	1	...	0	1	0	0	
symmetry_se	1	1	1	1	0	0	0	0	0	0	...	0	0	0	1	
concave_points_se	1	1	1	1	0	1	0	0	0	0	...	0	0	0	0	
fractal_dimension_mean	0	0	1	1	0	0	0	0	0	1	...	0	1	0	1	

5 rows × 22 columns

Figure A.10 – Part of final dataframe with the most important features

A.4 DIAGNOSTIC METHODOLOGY – MODELLING

Table A.2 – Lists created with different number of features

List	Features
First list	<i>texture_worst, area_worst, concave_points_worst</i>
Second list	<i>perimeter_worst, worst_volume, texture_worst, area_worst, concave_points_worst</i>
Third List	<i>se_volume, smootheness_worst, concave_points_mean, area_se, perimeter_worst, worst_volume, texture_worst, area_worst, concave_points_worst</i>

```
#Function to score
def IndividualModel_Scoring(X_train, X_val, y_train, y_tval, model, model_name, index):
    """
    X_train, X_val, y_train, y_val -> The sets splitted,
    model -> model object
    model_name -> Name for the output, for the results
    """
    #To define the output dataset:
    df_out = pd.DataFrame(columns = ['Model', 'Scale Type', 'Training', 'Validation'])

    #To fit the model and data
    model_fit = model.fit(X_train, y_train)
    pred_y_train = model_fit.predict(X_train)
    pred_y_test = model_fit.predict(X_val)

    #To append the results
    line = [ model_name, 'Robust Scaler', accuracy_score(y_train, pred_y_train),
            accuracy_score(y_val, pred_y_test)]

    appending_lines(df_out, line, index)

    return df_out
```

Figure A.11 – Function IndividualModel_Scoring(X_train, X_val, y_train, y_tval, model, model_name, index)

```

#To use a validation method (in this case, KFold)
def model_scores(n_split, X, y, model):
    seconds = []
    score_train = []
    score_val = []

    accuracy_train = []
    accuracy_val = []

    specificity_train = []
    specificity_val = []

    sensitivity_train = []
    sensitivity_val = []

    #To loop for partition
    for train_index, val_index in (KFold(n_splits = n_split)).split(X):

        X_train, X_val = X.iloc[train_index], X.iloc[val_index]
        y_train, y_val = y.iloc[train_index], y.iloc[val_index]

        #To start time
        begin = time.perf_counter()

        # Fitting model
        model = model.fit(X_train, y_train)

        # Predicting
        y_pred_train = model.predict(X_train)
        y_pred_test = model.predict(X_val)

        # End time
        end = time.perf_counter()
        seconds.append(end-begin)

        accuracy_train.append(accuracy_score(y_train, y_pred_train))
        accuracy_val.append(accuracy_score(y_val, y_pred_test))

        score_train.append(f1_score(y_train, y_pred_train))
        score_val.append(f1_score(y_val, y_pred_test))

        specificity_train.append(specificity_score(y_train, y_pred_train))
        specificity_val.append(specificity_score(y_val, y_pred_test))

        sensitivity_train.append(sensitivity_score(y_train, y_pred_train))
        sensitivity_val.append(sensitivity_score(y_val, y_pred_test))

    df_F1_Score = pd.DataFrame()
    df_accuracy = pd.DataFrame()
    df_specificity = pd.DataFrame()
    df_sensitivity = pd.DataFrame()

    #F1 Score
    df_F1_Score['Time'] = seconds
    df_F1_Score['Training'] = score_train
    df_F1_Score['Val'] = score_val

    #Accuracy Score
    df_accuracy['Time'] = seconds
    df_accuracy['Training'] = accuracy_train
    df_accuracy['Val'] = accuracy_val

    #Specificity Score
    df_specificity['Time'] = seconds
    df_specificity['Training'] = specificity_train
    df_specificity['Val'] = specificity_val

    #Sensitivity Score
    df_sensitivity['Time'] = seconds
    df_sensitivity['Training'] = sensitivity_train
    df_sensitivity['Val'] = sensitivity_val

    return df_F1_Score, df_accuracy, df_specificity, df_sensitivity

```

Figure A.12 – Function model_scores(n_split, X, y, model)

```

#To get the mean and sdv
def dataframe_mean_std(df):

    #To calculate mean and stdv
    df = df.agg({'mean', 'std'},axis = 0)

    #To transform all columns in str
    for col in df.columns:
        df[col] = np.round(df[col],3)
        df[col] = df[col].apply(str)

    #Aux dataframe
    dfout = df.copy()

    for c in df.columns:
        dfout[c] = df[c].loc['mean'] + ' +/- ' + df[c].loc['std']

    #To drop the row
    dfout.drop(['mean'], inplace = True)

    #To reset the index
    dfout.reset_index(drop = True, inplace = True)

    return dfout

```

Figure A.13 – Function dataframe_mean_std(df)

B.1 PROGNOSTIC METHODOLOGY – BUSINESS UNDERSTANDING

Table B.2 – Real-Valued features

Feature	Detail
radius	Mean of distances from center to points on the perimeter
texture	Standard deviation of gray-scale values
perimeter	Mean of total distance around the outside of nuclear cell shape
area	Amount of surface covered by the nuclear cell shape
smoothness	Local variation in radius lengths
compactness	Ration of the square of the perimeter to the area, minus 1
concavity	Severity of concave portions of the contour
concave points	Number of concave portions of the contour
symmetry	The longest chord passing the through the center
fractal dimension	"Coastline approximation" - 1

B.2 PROGNOSTIC METHODOLOGY – DATA UNDERSTANDING

outcome	2	concavity_se	192
time	95	concave_points_se	187
radius_mean	177	symmetry_se	189
texture_mean	193	fractal_dimension_se	195
perimeter_mean	181	radius_worst	182
area_mean	192	texture_worst	187
smoothness_mean	179	perimeter_worst	183
compactness_mean	192	area_worst	191
concavity_mean	196	smoothness_worst	172
concave_points_mean	189	compactness_worst	191
symmetry_mean	175	concavity_worst	197
fractal_dimension_mean	194	concave_points_worst	185
radius_se	196	symmetry_worst	192
texture_se	191	fractal_dimension_worst	189
perimeter_se	192	tumor_size	39
area_se	196	lymph_node_status	23
smoothness_se	196	dtype: int64	
compactness_se	193		

Figure B.1 – Distinct observations in WPBC dataset

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
outcome	198	2	N	151	NaN	NaN	NaN	NaN	NaN	NaN	NaN
time	198.0	NaN	NaN	NaN	46.732323	34.46287	1.0	14.0	39.5	72.75	125.0
radius_mean	198.0	NaN	NaN	NaN	17.412323	3.161676	10.95	15.0525	17.29	19.58	27.22
texture_mean	198.0	NaN	NaN	NaN	22.27601	4.29829	10.38	19.4125	21.75	24.655	39.28
perimeter_mean	198.0	NaN	NaN	NaN	114.856566	21.383402	71.9	98.16	113.7	129.65	182.1
area_mean	198.0	NaN	NaN	NaN	970.040909	352.149215	361.6	702.525	929.1	1193.5	2250.0
smoothness_mean	198.0	NaN	NaN	NaN	0.102681	0.012522	0.07497	0.0939	0.1019	0.110975	0.1447
compactness_mean	198.0	NaN	NaN	NaN	0.142648	0.049898	0.04605	0.1102	0.13175	0.1722	0.3114
concavity_mean	198.0	NaN	NaN	NaN	0.156243	0.070572	0.02398	0.10685	0.15135	0.2005	0.4268
concave_points_mean	198.0	NaN	NaN	NaN	0.086776	0.033877	0.02031	0.06367	0.086075	0.103925	0.2012
symmetry_mean	198.0	NaN	NaN	NaN	0.192754	0.027437	0.1308	0.174075	0.18935	0.209325	0.304
fractal_dimension_mean	198.0	NaN	NaN	NaN	0.062706	0.00724	0.05025	0.056718	0.061715	0.066715	0.09744
radius_se	198.0	NaN	NaN	NaN	0.603346	0.310112	0.1938	0.3882	0.53325	0.7509	1.819
texture_se	198.0	NaN	NaN	NaN	1.26445	0.526467	0.3621	0.9213	1.1685	1.46325	3.503
perimeter_se	198.0	NaN	NaN	NaN	4.255394	2.194128	1.153	2.7425	3.767	5.21275	13.28
area_se	198.0	NaN	NaN	NaN	70.228737	47.982255	13.99	35.365	58.455	92.4775	316.0
smoothness_se	198.0	NaN	NaN	NaN	0.006762	0.002974	0.002667	0.005001	0.006193	0.007973	0.03113
compactness_se	198.0	NaN	NaN	NaN	0.031199	0.017613	0.007347	0.019803	0.02788	0.038335	0.1354
concavity_se	198.0	NaN	NaN	NaN	0.04075	0.020869	0.01094	0.02681	0.03691	0.04897	0.1438
concave_points_se	198.0	NaN	NaN	NaN	0.015099	0.005504	0.005174	0.011422	0.014175	0.017665	0.03927
symmetry_se	198.0	NaN	NaN	NaN	0.020555	0.009578	0.007882	0.014795	0.017905	0.02288	0.06041
fractal_dimension_se	198.0	NaN	NaN	NaN	0.003987	0.001938	0.001087	0.002748	0.003719	0.00463	0.01256
radius_worst	198.0	NaN	NaN	NaN	21.021818	4.242997	12.84	17.6325	20.525	23.73	35.13
texture_worst	198.0	NaN	NaN	NaN	30.139091	6.017777	16.67	26.21	30.135	33.555	49.54
perimeter_worst	198.0	NaN	NaN	NaN	140.347778	28.892279	85.1	118.075	136.5	159.875	232.2
area_worst	198.0	NaN	NaN	NaN	1404.958586	586.006972	508.1	947.275	1295.0	1694.25	3903.0
smoothness_worst	198.0	NaN	NaN	NaN	0.143921	0.022004	0.08191	0.129325	0.14185	0.154875	0.2226
compactness_worst	198.0	NaN	NaN	NaN	0.365102	0.163965	0.05131	0.2487	0.3513	0.423675	1.058
concavity_worst	198.0	NaN	NaN	NaN	0.436685	0.173625	0.02398	0.32215	0.40235	0.54105	1.17
concave_points_worst	198.0	NaN	NaN	NaN	0.178778	0.045181	0.02899	0.15265	0.17925	0.207125	0.2903
symmetry_worst	198.0	NaN	NaN	NaN	0.323404	0.075161	0.1565	0.27595	0.3103	0.3588	0.6638
fractal_dimension_worst	198.0	NaN	NaN	NaN	0.090828	0.021172	0.05504	0.076577	0.08689	0.101375	0.2075
tumor_size	198.0	NaN	NaN	NaN	2.847475	1.937964	0.4	1.5	2.5	3.5	10.0
lymph_node_status	198	23	0	87	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure B.2 – Descriptive statistics in WPBC dataset

#	Column	Non-Null Count	Dtype
0	outcome	198 non-null	object
1	time	198 non-null	int64
2	radius_mean	198 non-null	float64
3	texture_mean	198 non-null	float64
4	perimeter_mean	198 non-null	float64
5	area_mean	198 non-null	float64
6	smoothness_mean	198 non-null	float64
7	compactness_mean	198 non-null	float64
8	concavity_mean	198 non-null	float64
9	concave_points_mean	198 non-null	float64
10	symmetry_mean	198 non-null	float64
11	fractal_dimension_mean	198 non-null	float64
12	radius_se	198 non-null	float64
13	texture_se	198 non-null	float64
14	perimeter_se	198 non-null	float64
15	area_se	198 non-null	float64
16	smoothness_se	198 non-null	float64
17	compactness_se	198 non-null	float64
18	concavity_se	198 non-null	float64
19	concave_points_se	198 non-null	float64
20	symmetry_se	198 non-null	float64
21	fractal_dimension_se	198 non-null	float64
22	radius_worst	198 non-null	float64
23	texture_worst	198 non-null	float64
24	perimeter_worst	198 non-null	float64
25	area_worst	198 non-null	float64
26	smoothness_worst	198 non-null	float64
27	compactness_worst	198 non-null	float64
28	concavity_worst	198 non-null	float64
29	concave_points_worst	198 non-null	float64
30	symmetry_worst	198 non-null	float64
31	fractal_dimension_worst	198 non-null	float64
32	tumor_size	198 non-null	float64
33	lymph_node_status	198 non-null	object

dtypes: float64(31), int64(1), object(2)
memory usage: 62.2+ KB

Figure B.3 – Types of the columns in WPBC dataset

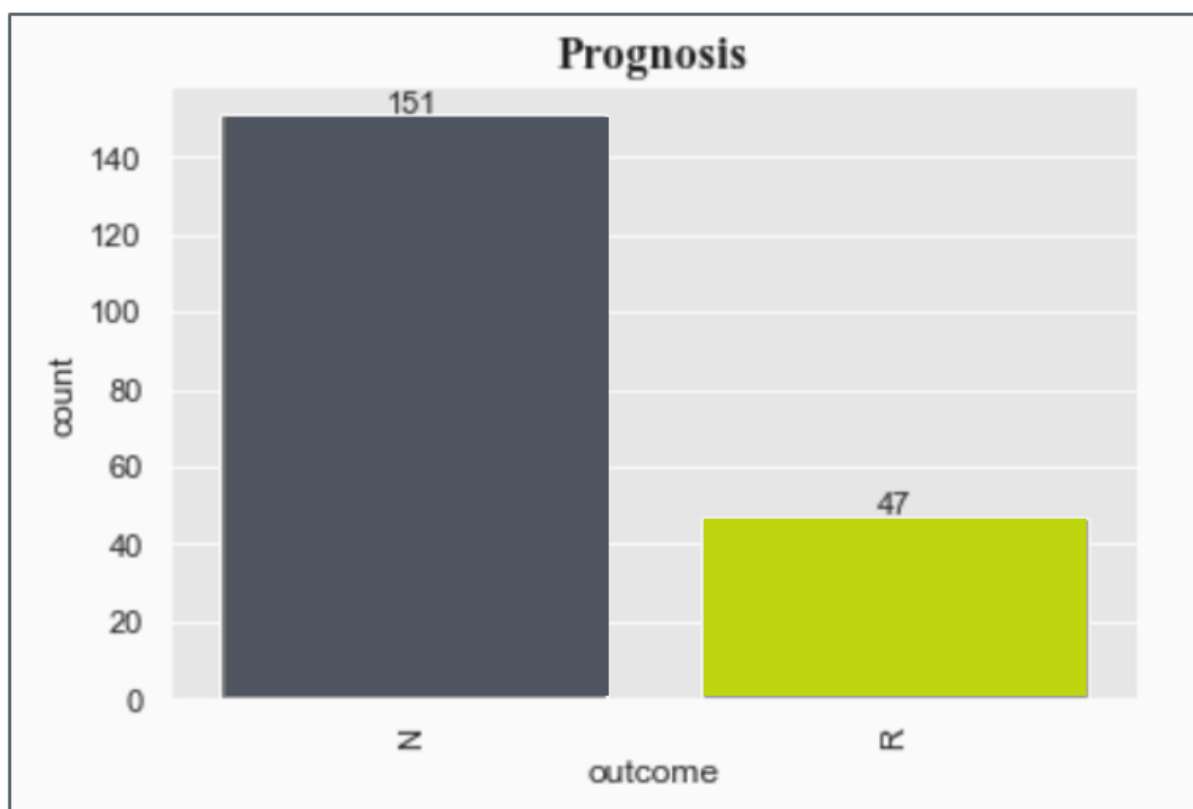


Figure B.4 – Number of people with different breast cancer prognoses

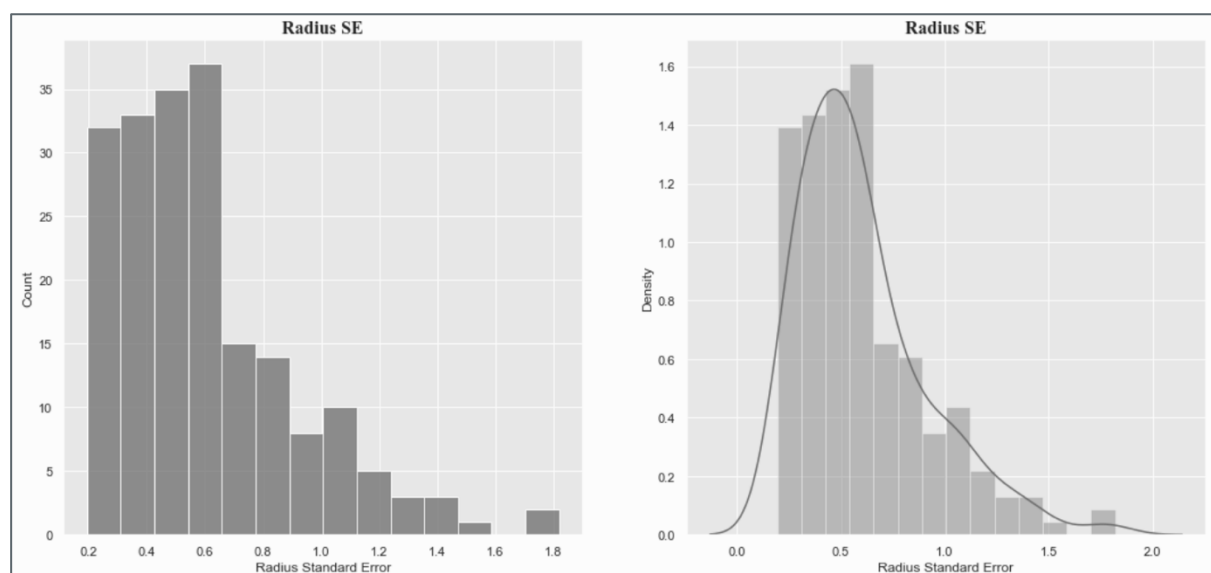


Figure B.5 – Radius SE distribution – histogram and displot

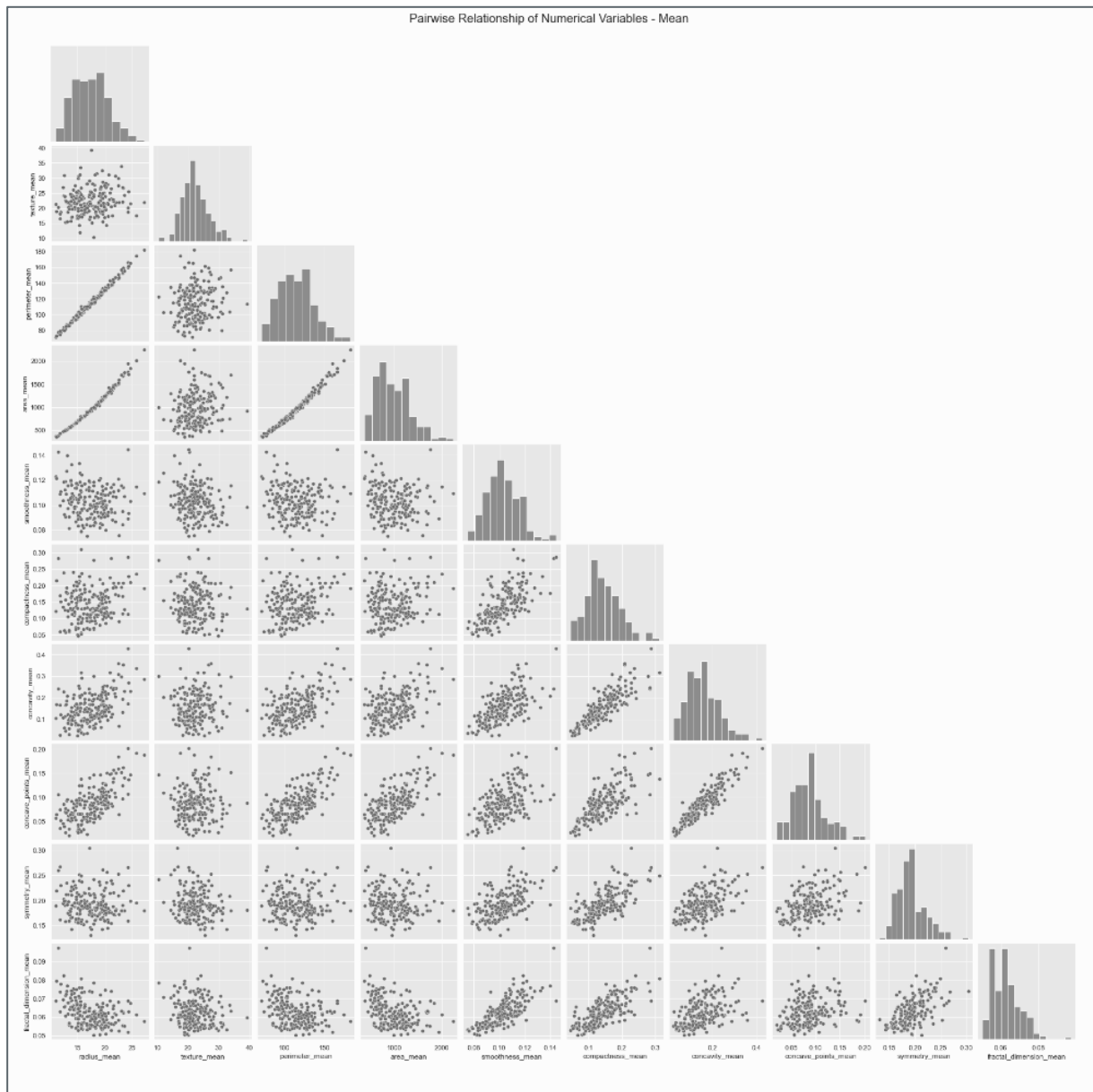


Figure B.6 – Relationship between mean numerical variable

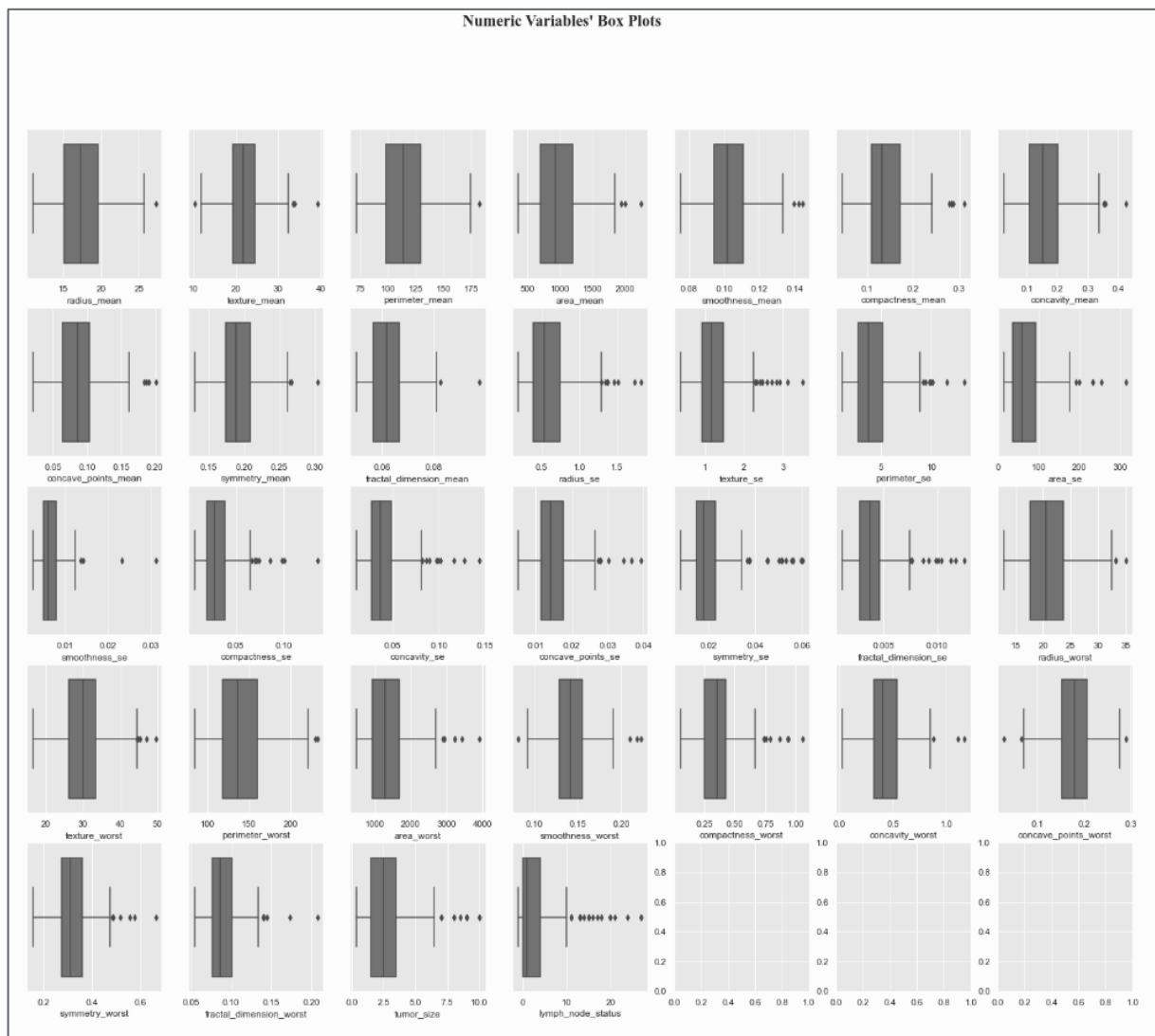


Figure B.7 – Relationship between mean numerical variable

B.3 PROGNOSTIC METHODOLOGY – DATA PREPROCESSING

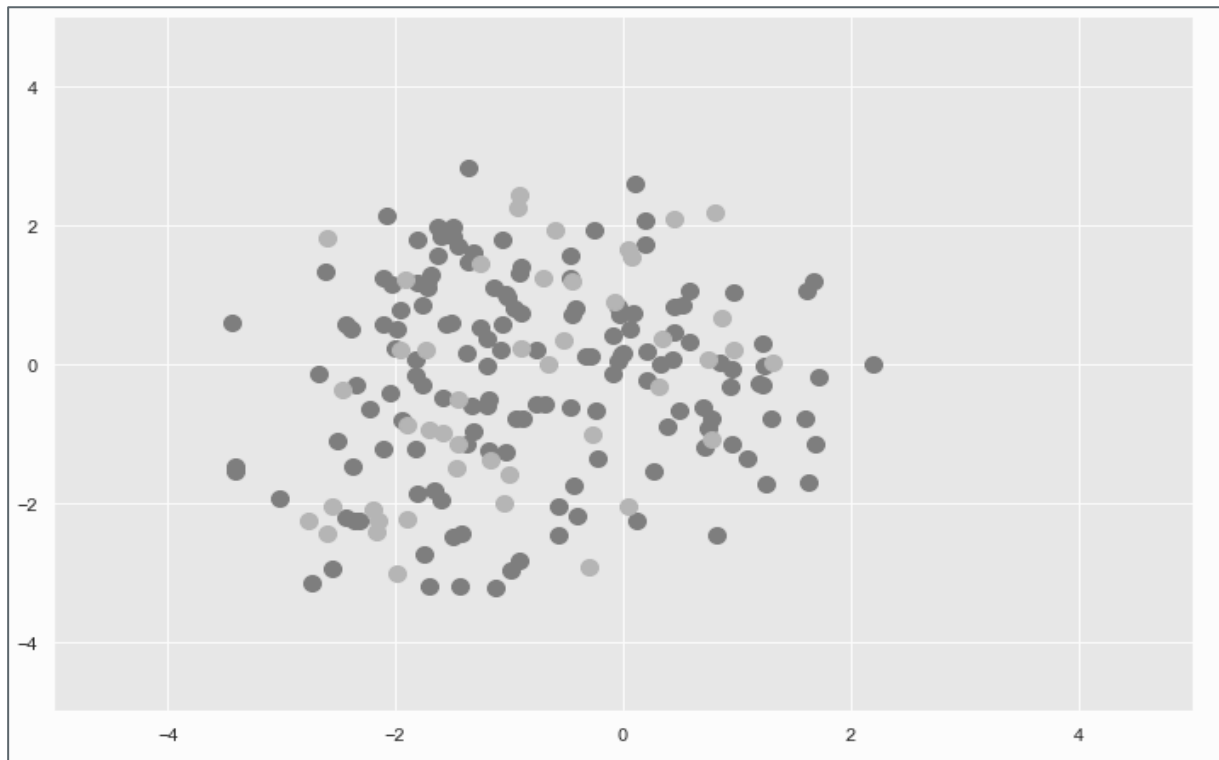


Figure B.8 – Imbalanced Data

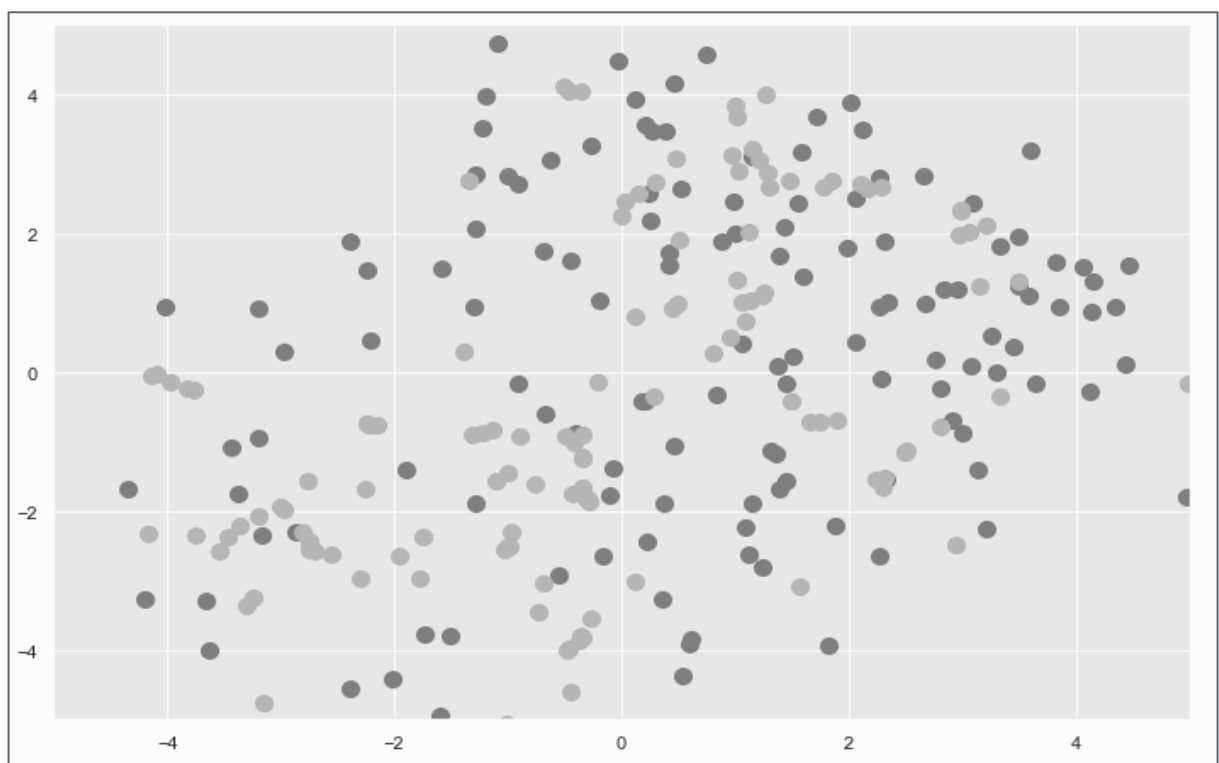


Figure B.9 – Balanced Data

	Pearson	Spearman	Kendall	Univariate Variables	ANOVA	Mutual Information	RFE	Lasso	Elastic Net	PCA	...	MSE	MAE	FRIEDMAN	AdaBoost	Random Forest	F
smoothness_worst	1	1	1	1	0	1	1	1	1	0	...	1	1	1	1	1	
smoothness_mean	0	0	1	0	0	1	1	1	1	1	...	1	1	1	1	1	
concave_points_se	1	1	1	0	0	1	1	1	1	0	...	1	1	1	1	1	
texture_se	1	1	1	1	0	1	1	1	1	0	...	1	1	1	1	1	
tumor_size	1	1	1	1	1	1	1	1	1	0	...	1	1	1	1	1	

Figure B.10 – Part of final dataframe with the most important features in Classification Problem

	Pearson	Spearman	Kendall	Univariate Variables	ANOVA	RFE	Boruta	Lasso	Elastic Net	PCA	...	MAE	FRIEDMAN	AdaBoost	Random Forest	Li Regres
smoothness_worst	1	1	1	1	0	1	0	0	0	1	...	1	1	1	1	
texture_se	1	1	1	1	0	0	0	1	1	0	...	1	1	1	1	
perimeter_mean	0	0	0	0	1	1	1	0	1	1	...	1	1	1	1	
texture_mean	0	0	1	1	0	0	1	1	1	0	...	1	1	1	1	
fractal_dimension_worst	1	1	1	1	0	1	0	1	1	1	...	0	1	1	1	

Figure B.11 – Part of final dataframe with the most important features in Regression Problem

B.4 PROGNOSTIC METHODOLOGY – MODELLING

Table B.2 – Lists with final features in Classification Problem

List	Features
First list	<i>concave_points_se, texture_se, tumor_size</i>
Second list	<i>concave_points_se, texture_se, smoothness_mean</i>
Third List	<i>texture_mean, area_worst, concavity_worst, fractal_dimension_worst</i>
Fourth List	<i>smoothness_worst, smoothness_mean, texture_se, tumor_size, concave_points_se</i>
Fifth List	<i>smoothness_worst, concave_points_se, texture_worst, concave_points_worst, symmetry_worst, lymph_node_status, smoothness_mean, texture_se, tumor_size</i>

Table B.3 – Lists with final features in Regression Problem

List	Features
First list	<i>fractal_dimension_worst, texture_mean, perimeter_mean</i>
Second list	<i>smoothness_worst, texture_se, perimeter_mean, texture_mean, fractal_dimension_worst</i>
Third List	<i>Symmetry_se, se_volume, symmetry_mean, concave_points_se, smoothness_worst, texture_Se, perimeter_mean, texture_mean, fractal_dimension_worst</i>

```
#Function to score
def IndividualModel_Scoring(X_train, X_val, y_train, y_val, model, model_name, index):
    """
    X_train, X_val, y_train, y_val -> The sets splitted,
    model -> model object
    model_name -> Name for the output, for the results
    """
    #To define the output dataset:
    df_out = pd.DataFrame(columns = ['Model', 'Scale Type', 'MSE - Train', 'MSE - Test', 'Score R^2 - Train', 'Score R^2 - Test'])

    #To fit the model and data
    model = model.fit(X_train, y_train)
    pred_y_train = model.predict(X_train)
    pred_y_test = model.predict(X_val)

    train_mse = mean_squared_error(y_train, pred_y_train)
    test_mse = mean_squared_error(y_val, pred_y_test)

    #score
    train_score = (model.score(X_train, y_train))
    test_score = (model.score(X_val, y_val))

    #To append the results
    line = [ model_name, 'Robust Scaler', train_mse, test_mse, train_score, test_score]

    appending_lines(df_out, line, index)

    return df_out
```

Figure B.12 – Function IndividualModel_Scoring(X_train, X_val, y_train, y_tval, model, model_name, index)

```
def MAPE(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

```
#To use a validation method (in this case, KFold)
def model_scores(n_split, X, y, model):
    seconds = []

    mae_train = []
    mae_val = []

    mse_train = []
    mse_val = []

    rmse_train = []
    rmse_val = []

    rsquared_train = []
    rsquared_val = []

    mape_train = []
    mape_val = []

    #To loop for partition
    for train_index, val_index in (KFold(n_splits = n_split)).split(X):

        X_train, X_val = X.iloc[train_index], X.iloc[val_index]
        y_train, y_val = y.iloc[train_index], y.iloc[val_index]

        #To start time
        begin = time.perf_counter()

        # Fitting model
        model = model.fit(X_train, y_train)

        # Predicting
        y_pred_train = model.predict(X_train)
        y_pred_test = model.predict(X_val)

        # End time
        end = time.perf_counter()
        seconds.append(end-begin)

        =====

        #mae
        mae_train.append(mean_absolute_error(y_train, y_pred_train))
        mae_val.append(mean_absolute_error(y_val, y_pred_test))

        #mse
        mse_train.append(mean_squared_error(y_train, y_pred_train))
        mse_val.append(mean_squared_error(y_val, y_pred_test))

        #rmse
        rmse_train.append(mean_squared_error(y_train, y_pred_train, squared=False))
        rmse_val.append(mean_squared_error(y_val, y_pred_test, squared=False))

        #r2
        rsquared_train.append(r2_score(y_train, y_pred_train))
        rsquared_val.append(r2_score(y_val, y_pred_test))

        #mape - Mean Absolute Percentage Error
        mape_train.append(MAPE(y_train, y_pred_train))
        mape_val.append(MAPE(y_val, y_pred_test))

        =====

    df_mae = pd.DataFrame()
    df_mse = pd.DataFrame()
    df_rmse = pd.DataFrame()
    df_rsquared = pd.DataFrame()
    df_mape = pd.DataFrame()

    #MAE
    df_mae['Time'] = seconds
    df_mae['Training'] = mae_train
    df_mae['Val'] = mae_val

    #MSE
    df_mse['Time'] = seconds
    df_mse['Training'] = mse_train
    df_mse['Val'] = mse_val

    #RMSE
    df_rmse['Time'] = seconds
    df_rmse['Training'] = rmse_train
    df_rmse['Val'] = rmse_val

    #R2 Squared
    df_rsquared['Time'] = seconds
    df_rsquared['Training'] = rsquared_train
    df_rsquared['Val'] = rsquared_val

    #Mape
    df_mape['Time'] = seconds
    df_mape['Training'] = mape_train
    df_mape['Val'] = mape_val

    return df_mae, df_mse, df_rmse, df_rsquared, df_mape
```

Figure B.13 – Function model_scores(n_split, X, y, model) in Regression model

C.1 DIAGNOSTIC RESULTS - NAÏVE BAYES

Table C.1 – Accuracy of Naïve Bayes with different numbers of features

	Model	Scale Type	Training	Validation
0	Naïve Bayes – All Features	Robust Scaler	0.941884	0.944186
1	Naïve Bayes – 3 Features	Robust Scaler	0.965932	0.972093
2	Naïve Bayes – 5 Features	Robust Scaler	0.959920	0.948837
3	Naïve Bayes – 9 Features	Robust Scaler	0.943888	0.972093

Table C.2 – F1-Score, Accuracy, Specificity, Sensitivity of Naïve Bayes of Final Naïve Bayes model with 3 features

	Time	Training	Validation	Model
0	0.004 +/- 0.001	0.968 +/- 0.004	0.952 +/- 0.034	F1-Score
1	0.004 +/- 0.001	0.968 +/- 0.004	0.965 +/- 0.018	Accuracy
2	0.004 +/- 0.001	0.968 +/- 0.004	0.771 +/- 0.406	Specificity
3	0.004 +/- 0.001	0.967 +/- 0.004	0.964 +/- 0.035	Sensitivity

C.2 DIAGNOSTIC RESULTS - LOGISTIC REGRESSION (LR)

Table C.3 – Accuracy of Logistic Regression with different numbers of features

	Model	Scale Type	Training	Validation
0	Logistic Regression – All Features	Robust Scaler	0.977956	0.990698
1	Logistic Regression – 3 Features	Robust Scaler	0.969940	0.976744
2	Logistic Regression – 5 Features	Robust Scaler	0.967936	0.972093
3	Logistic Regression – 9 Features	Robust Scaler	0.975952	0.986047

Table C.4 – Accuracy of Logistic Regression with different parameters

	Model	Scale Type	Training	Validation
0	LR 9Features – Newton	Robust Scaler	0.975952	0.986047
1	LR 9Features – Lib	Robust Scaler	0.973948	0.986047
2	LR 9Features – Sag	Robust Scaler	0.975952	0.986047
3	LR 9Features – Saga	Robust Scaler	0.975952	0.986047

Table C.5 – F1-Score, Accuracy, Specificity, Sensitivity of Logistic Regression of Final Logistic Regression model with 9 features

	Time	Training	Validation	Model
0	0.011 +/- 0.001	0.974 +/- 0.002	0.962 +/- 0.030	F1-Score
1	0.011 +/- 0.001	0.974 +/- 0.003	0.972 +/- 0.013	Accuracy
2	0.011 +/- 0.001	0.982 +/- 0.004	0.784 +/- 0.413	Specificity
3	0.011 +/- 0.001	0.967 +/- 0.004	0.963 +/- 0.035	Sensitivity

C.3 DIAGNOSTIC RESULTS – DECISION TREE (DT)

Table C.6 – Accuracy of Decision Tree with different number of features

	Model	Scale Type	Training	Validation
0	Decision Tree – All Features	Robust Scaler	1.000000	0.925581
1	Decision Tree – 3 Features	Robust Scaler	1.000000	0.948837
2	Decision Tree – 5 Features	Robust Scaler	1.000000	0.944186
3	Decision Tree – 9 Features	Robust Scaler	1.000000	0.967442

Table C.7 – Accuracy of Decision Tree with different *Criterion* parameter

	Model	Training	Validation
0	DT 9Features – Default Gini	1.000000	0.972093
1	DT 9Features – Entropy	1.000000	0.925581

Table C.8 – Accuracy of Decision Tree with different *Splitter* parameter

	Model	Training	Validation
0	DT 9Features – Default Best	1.000000	0.953488
1	DT 9Features – Random	1.000000	0.939535

Table C.9 – Accuracy of Decision Tree with different *Max Depth* parameter

	Model	Training	Validation
0	DT 9Features – Default None	1.000000	0.958140
1	DT 9Features – 5	0.995992	0.967442
2	DT 9Features – 10	1.000000	0.967442
3	DT 9Features – 20	1.000000	0.958140
4	DT 9Features – 27	1.000000	0.972093
5	DT 9Features – 50	1.000000	0.967442
6	DT 9Features – 68	1.000000	0.958140
7	DT 9Features – 100	1.000000	0.967442

Table C.10 – Accuracy of Decision Tree with different *Min Samples Split* parameter

	Model	Training	Validation
0	DT 9Features – Default 2	0.995992	0.967442
1	DT 9Features – 5	0.993988	0.967442
2	DT 9Features – 10	0.981964	0.948837
3	DT 9Features – 20	0.963928	0.934884
4	DT 9Features – 43	0.939880	0.902326
5	DT 9Features – 100	0.939880	0.902326
6	DT 9Features – 250	0.939880	0.902326
7	DT 9Features – 500	0.501002	0.497674
8	DT 9Features – 1000	0.501002	0.497674

Table C.11 – Accuracy of Decision Tree with different *Min Samples Leaf* parameter

	Model	Training	Validation
0	DT 9Features – Default 1	0.995992	0.967442
1	DT 9Features – 5	0.977956	0.958140
2	DT 9Features – 10	0.963928	0.934884
3	DT 9Features – 50	0.937876	0.879070
4	DT 9Features – 75	0.937876	0.879070
5	DT 9Features – 100	0.937876	0.879070
6	DT 9Features – 250	0.501002	0.497674
7	DT 9Features – 500	0.501002	0.497674

Table C.12 – Accuracy of Decision Tree with different *Min Weight Fraction Leaf* parameter

	Model	Training	Validation
0	DT 9Features – Default 0.0	0.977956	0.958140
1	DT 9Features – 0.08	0.937876	0.879070
2	DT 9Features – 0.15	0.937876	0.879070
3	DT 9Features – 0.30	0.937876	0.879070
4	DT 9Features – 0.50	0.501002	0.497674

Table C.13 – Accuracy of Decision Tree with different *Min Impurity Decrease* parameter

	Model	Training	Validation
--	--------------	-----------------	-------------------

0	DT 9Features – Default 0.0	0.977956	0.958140
1	DT 9Features – 0.015	0.963928	0.934884
2	DT 9Features – 0.020	0.939880	0.902326
3	DT 9Features – 0.025	0.939880	0.902326

Table C.14 – Accuracy of Decision Tree best model with the best parameters

	Model	Scale Type	Training	Validation
0	Decision Tree – Best Parameters	Robust Scaler	0.977956	0.958140

Table C.15 – F1-Score, Accuracy, Specificity, Sensitivity of Decision Tree of Final Decision Tree model with 9 features

	Time	Training	Validation	Model
0	0.006 +/- 0.001	0.981 +/- 0.004	0.934 +/- 0.056	F1-Score
1	0.006 +/- 0.001	0.981 +/- 0.003	0.951 +/- 0.033	Accuracy
2	0.006 +/- 0.001	0.982 +/- 0.006	0.764 +/- 0.404	Specificity
3	0.006 +/- 0.001	0.980 +/- 0.007	0.944 +/- 0.054	Sensitivity

C.4 DIAGNOSTIC RESULTS – RANDOM FOREST (RF)

Table C.16 – Accuracy of Random Forest with different number of features

	Model	Scale Type	Training	Validation
0	Random Forest – All Features	Robust Scaler	1.000000	0.972093
1	Random Forest – 3 Features	Robust Scaler	1.000000	0.976744
2	Random Forest – 5 Features	Robust Scaler	1.000000	0.962791
3	Random Forest – 9 Features	Robust Scaler	1.000000	0.962791

Table C.17 – Accuracy of Random Forest with different *N Estimators* parameter

	Model	Training	Validation
0	RF 3Features – Default 2	0.969940	0.897674
1	RF 3Features – 5	0.991984	0.958140
2	RF 3Features – 10	0.997996	0.976744
3	RF 3Features – 100	1.000000	0.967442
4	RF 3Features – 300	1.000000	0.962791
5	RF 3Features – 570	1.000000	0.962791

Table C.18 – Accuracy of Random Forest with different *Criterion* parameter

	Model	Training	Validation
0	RF 3Features – Default Gini	0.995992	0.962791
1	RF 3Features – Entropy	0.997996	0.962791

Table C.19 – Accuracy of Random Forest with different *Max Depth* parameter

	Model	Training	Validation
0	RF 3Features – Default None	0.995992	0.967442
1	RF 3Features – 1	0.871743	0.855814
2	RF 3Features – 2	0.973948	0.948837
3	RF 3Features – 4	0.985972	0.953488
4	RF 3Features – 5	0.983968	0.958140
5	RF 3Features – 8	0.997996	0.953488

Table C.20 – Accuracy of Random Forest with different *Min Samples Leaf* parameter

	Model	Training	Validation
0	RF 3Features – Default 1	0.985972	0.958140
1	RF 3Features – 2	0.985972	0.967442
2	RF 3Features – 4	0.979960	0.944186
3	RF 3Features – 5	0.979960	0.944837

Table C.21 – Accuracy of Random Forest with different *Bootstrap* parameter

	Model	Training	Validation
0	RF 3Features – Default True	0.987976	0.967442
1	RF 3Features – False	0.989980	0.962791

Table C.22 – Accuracy of Random Forest with different *Max Samples* parameter

	Model	Training	Validation
0	RF 3Features – Default None	0.985972	0.953488
1	RF 3Features – 0.30	0.975952	0.962791
2	RF 3Features – 0.60	0.979960	0.958140
3	RF 3Features – 0.80	0.981964	0.967442
4	RF 3Features – 0.90	0.983968	0.972093

Table C.23 – Accuracy of Random Forest best model with the best parameters

	Model	Scale Type	Training	Validation
0	Random Forest – Best Parameters	Robust Scaler	0.985972	0.962791

Table C.24 – F1-Score, Accuracy, Specificity, Sensitivity of Random Forest of Final Random Forest model with 3 features

	Time	Training	Validation	Model
0	0.022 +/- 0.002	0.982 +/- 0.003	0.962 +/- 0.028	F1-Score
1	0.022 +/- 0.002	0.982 +/- 0.003	0.971 +/- 0.017	Accuracy
2	0.022 +/- 0.002	0.987 +/- 0.005	0.781 +/- 0.412	Specificity

3	0.022 +/- 0.002	0.978 +/- 0.007	0.967 +/- 0.048	Sensitivity
---	-----------------	-----------------	-----------------	--------------------

C.5 DIAGNOSTIC RESULTS – NEURAL NETWORK (NN)

Table C.25 – Accuracy of Neural Network with different number of features

	Model	Scale Type	Training	Validation
0	Neural Network – All Features	Robust Scaler	0.993988	0.981395
1	Neural Network – 3 Features	Robust Scaler	0.971944	0.976744
2	Neural Network – 5 Features	Robust Scaler	0.967936	0.972093
3	Neural Network – 9 Features	Robust Scaler	0.979960	0.986047

Table C.26 – Accuracy of Neural Network with different *Hidden Layer Sizes* parameter

	Model	Training	Validation
0	NN 3Features – Default (100,)	0.971944	0.976744
1	NN 3Features – (25,25)	0.973948	0.976744
2	NN 3Features – (50,100)	0.981964	0.976744
3	NN 3Features – (6,)	0.955912	0.972093
4	NN 3Features – (5,2)	0.963928	0.981395
5	NN 3Features – (10,5)	0.969940	0.976744
6	NN 3Features – (50,100,50)	0.987976	0.976744
7	NN 3Features – (100, 150, 50, 75)	0.997996	0.972093
8	NN 3Features – (250, 250)	0.985972	0.976744
9	NN 3Features – (250, 220, 200)	1.000000	0.976744

Table C.27 – Accuracy of Neural Network with different *Activation* parameter

	Model	Training	Validation
0	NN 3Features – Default <i>relu</i>	0.973948	0.976744
1	NN 3Features – <i>identity</i>	0.965932	0.976744
2	NN 3Features – <i>logistic</i>	0.971944	0.981395
3	NN 3Features – <i>tanh</i>	0.965932	0.976744

Table C.28 – Accuracy of Neural Network with different *Solver* parameter

	Model	Training	Validation
0	NN 3Features – Default <i>adam</i>	0.971944	0.976744

1	NN 3Features – <i>sgd</i>	0.967936	0.976744
2	NN 3Features – <i>lbfgs</i>	1.000000	0.948837

Table C.29 – Accuracy of Neural Network with different *Alpha* parameter

	Model	Training	Validation
0	NN 3Features – Default 0.0001	0.973948	0.976744
1	NN 3Features – 0.001	0.973948	0.976744
2	NN 3Features – 0.01	0.975952	0.976744
3	NN 3Features – 0.05	0.973948	0.976744
4	NN 3Features – 0.30	0.969940	0.976744

Table C.30 – Accuracy of Neural Network with different *Learning Rate* parameter

	Model	Training	Validation
0	NN 3Features – Default <i>constant</i>	0.971944	0.976744
1	NN 3Features – <i>invscaling</i>	0.973948	0.972093
2	NN 3Features – <i>adaptive</i>	0.971944	0.976744

Table C.31 – Accuracy of Neural Network with different *Learning Rate Init* parameter

	Model	Training	Validation
0	NN 3Features – Default 0.001	0.971944	0.976744
1	NN 3Features – 0.01	0.985972	0.962791
2	NN 3Features – 0.0001	0.961924	0.967662

Table C.32 – Accuracy of Neural Network with different *Max Iter* parameter

	Model	Training	Validation
0	NN 3Features – Default 200	0.975952	0.976744
1	NN 3Features – 40	0.963928	0.976744
2	NN 3Features – 75	0.969940	0.976744
3	NN 3Features – 100	0.973948	0.976744
4	NN 3Features – 400	0.971944	0.976744
5	NN 3Features – 1000	0.973948	0.976744

Table C.33 – Accuracy of Neural Network with different *Shuffle* parameter

	Model	Training	Validation
0	NN 3Features – Default True	0.975952	0.976744
1	NN 3Features – False	0.971944	0.981395

Table C.34 – Accuracy of Neural Network best model with the best parameters

	Model	Scale Type	Training	Validation
0	Neural Network – Best Parameters	Robust Scaler	0.973948	0.976744

Table C.35 – F1-Score, Accuracy, Specificity, Sensitivity of Neural Network of Final Neural Network model with 3 features

	Time	Training	Validation	Model
0	0.373 +/- 0.052	0.975 +/- 0.003	0.968 +/- 0.015	F1-Score
1	0.373 +/- 0.052	0.975 +/- 0.004	0.973 +/- 0.014	Accuracy
2	0.373 +/- 0.052	0.978 +/- 0.004	0.785 +/- 0.414	Specificity
3	0.373 +/- 0.052	0.972 +/- 0.003	0.966 +/- 0.031	Sensitivity

C.6 DIAGNOSTIC RESULTS – PROBABILISTIC NEURAL NETWORK (PROBABILISTIC NN)

Table C.36 – Accuracy of Probabilistic Neural Network with different number of features

	Model	Scale Type	Training	Validation
0	Probabilistic NN – All Features	Robust Scaler	1.000000	0.981395
1	Probabilistic NN – 3 Features	Robust Scaler	0.973948	0.976744
2	Probabilistic NN – 5 Features	Robust Scaler	0.975952	0.972093
3	Probabilistic NN – 9 Features	Robust Scaler	0.989980	0.986047

Table C.37 – Accuracy of Probabilistic Neural Network best model with the best parameters

	Model	Scale Type	Training	Validation
0	Neural Network – Best Parameters	Robust Scaler	0.989980	0.986046

Table C.38 – F1-Score, Accuracy, Specificity, Sensitivity of Probabilistic Neural Network of Final Probabilistic Neural Network model with 9 features

	Time	Training	Validation	Model
0	0.069 +/- 0.015	0.992 +/- 0.002	0.981 +/- 0.011	F1-Score
1	0.069 +/- 0.015	0.992 +/- 0.002	0.979 +/- 0.014	Accuracy
2	0.069 +/- 0.015	0.998 +/- 0.001	0.787 +/- 0.415	Specificity
3	0.069 +/- 0.015	0.986 +/- 0.003	0.973 +/- 0.028	Sensitivity

C.7 DIAGNOSTIC RESULTS – LINEAR DISCRIMINANT ANALYSIS (LDA)

Table C.39 – Accuracy of Linear Discriminant Analysis with different number of features

	Model	Scale Type	Training	Validation
0	LDA – All Features	Robust Scaler	0.975952	0.972093
1	LDA – 3 Features	Robust Scaler	0.941884	0.972093
2	LDA – 5 Features	Robust Scaler	0.963928	0.972093
3	LDA – 9 Features	Robust Scaler	0.969940	0.986047

Table C.40 – Accuracy of Linear Discriminant Analysis best model with the best parameters

	Model	Scale Type	Training	Validation
0	LDA – Best Parameters	Robust Scaler	0.975952	0.972093

Table C.41 – F1-Score, Accuracy, Specificity, Sensitivity of Linear Discriminant Analysis of Final Linear Discriminant Analysis model with all features

	Time	Training	Validation	Model
0	0.010 +/- 0.001	0.976 +/- 0.004	0.959 +/- 0.020	F1-Score
1	0.010 +/- 0.001	0.977 +/- 0.003	0.964 +/- 0.019	Accuracy
2	0.010 +/- 0.001	0.996 +/- 0.002	0.794 +/- 0.419	Specificity
3	0.010 +/- 0.001	0.957 +/- 0.009	0.936 +/- 0.046	Sensitivity

C.8 DIAGNOSTIC RESULTS – SUPPORT VECTOR MACHINE (SVM)

Table C.42 – Accuracy of Support Vector Machine with different number of features

	Model	Scale Type	Training	Validation
0	SVM – All Features	Robust Scaler	0.977962	0.995349
1	SVM – 3 Features	Robust Scaler	0.971944	0.981395
2	SVM – 5 Features	Robust Scaler	0.967936	0.976744
3	SVM – 9 Features	Robust Scaler	0.971944	0.990698

Table C.43 – Accuracy of Support Vector Machine with different C parameter

	Model	Training	Validation
0	SVM 5Features – Default 1	0.967936	0.976744
1	SVM 5Features – 0.01	0.955912	0.976744
2	SVM 5Features – 0.4	0.967936	0.976744
3	SVM 5Features – 50	0.975952	0.972093
4	SVM 5Features – 10	0.971944	0.972093
5	SVM 5Features – 100	0.979960	0.972093

Table C.44 – Accuracy of Support Vector Machine with different *Kernel* parameter

	Model	Training	Validation
0	SVM 5Features – <i>linear</i>	0.965932	0.976744
1	SVM 5Features – <i>poly</i>	0.955912	0.976744
2	SVM 5Features – Default <i>rbf</i>	0.971944	0.972093
3	SVM 5Features – <i>sigmoid</i>	0.921844	0.860465

Table C.45 – Accuracy of Support Vector Machine with different *Gamma* parameter

	Model	Training	Validation
0	SVM 5Features – Default <i>scale</i>	0.971944	0.972093
1	SVM 5Features – <i>auto</i>	0.971944	0.976744

Table C.46 – Accuracy of Support Vector Machine with different *Max Iter* parameter

	Model	Training	Validation
0	SVM 5Features – (-1)	0.971944	0.972093
1	SVM 5Features – 3	0.695391	0.646512
2	SVM 5Features – 15	0.723447	0.688372
3	SVM 5Features – 50	0.971944	0.976744
4	SVM 5Features – 100	0.973948	0.972093
5	SVM 5Features – 145	0.973948	0.972093
6	SVM 5Features – 235	0.971944	0.972093
7	SVM 5Features – 500	0.971944	0.972093
8	SVM 5Features – 750	0.971944	0.972093
9	SVM 5Features – 1000	0.971944	0.972093
10	SVM 5Features – 2500	0.971944	0.972093

Table C.47 – Accuracy of Support Vector Machine best model with the best parameters

	Model	Scale Type	Training	Validation
0	SVM – Best Parameters	Robust Scaler	0.971944	0.972093

Table C.48 – F1-Score, Accuracy, Specificity, Sensitivity of Support Vector Machine of Final Support Vector Machine model with 5 features

	Time	Training	Validation	Model
0	0.007 +/- 0.001	0.974 +/- 0.003	0.962 +/- 0.017	F1-Score
1	0.007 +/- 0.001	0.974 +/- 0.002	0.969 +/- 0.011	Accuracy
2	0.007 +/- 0.001	0.979 +/- 0.002	0.785 +/- 0.414	Specificity
3	0.007 +/- 0.001	0.969 +/- 0.005	0.958 +/- 0.034	Sensitivity

C.9 DIAGNOSTIC RESULTS – MULTI-LAYER PERCEPTRON WITH GRID SEARCH (MLP w/ GS)

Table C.49 – Accuracy of Multi-Layer Perceptron with Grid Search with different number of features

	Model	Scale Type	Training	Validation
0	MLP w/ GS – All Features	Robust Scaler	0.995992	0.981395
1	MLP w/ GS – 3 Features	Robust Scaler	0.973948	0.976744
2	MLP w/ GS – 5 Features	Robust Scaler	0.969940	0.976744
3	MLP w/ GS – 9 Features	Robust Scaler	0.979960	0.986047

Table C.50 – *Parameter Space*

Parameter	Values
<i>hidden_layer_sizes</i>	[(25,25), (50,100), (5,2), (10,10,10), (45,2,11), (25,11,7,5,3,)]
<i>activation</i>	['tanh', 'relu', 'logistic']
<i>solver</i>	['adam', 'lbfgs', 'sgd']
<i>alpha</i>	[0.0001, 0.001]
<i>learning_rate</i>	['invscaling', 'constant']

Table C.51 – Best Parameters found from *Parameter Space* with *Grid Search*

Parameter	Best Value
<i>hidden_layer_sizes</i>	(25,25)
<i>activation</i>	<i>tanh</i>
<i>solver</i>	<i>adam</i>
<i>alpha</i>	0.0001
<i>learning_rate</i>	<i>invscaling</i>

Table C.52 – Accuracy of Multi-Layer Perceptron with Grid Search best model with the best parameters

	Model	Scale Type	Training	Validation
0	MLP w/ GS – Best Parameters	Robust Scaler	0.993988	0.995349

Table C.53 – F1-Score, Accuracy, Specificity, Sensitivity of Multi-Layer Perceptron with Grid Search of Final Multi-Layer Perceptron with Grid Search model with all features

	Time	Training	Validation	Model
0	0.452 +/- 0.035	0.997 +/- 0.002	0.983 +/- 0.014	F1-Score
1	0.452 +/- 0.035	0.997 +/- 0.002	0.985 +/- 0.014	Accuracy
2	0.452 +/- 0.035	1.000 +/- 0.000	0.789 +/- 0.416	Specificity
3	0.452 +/- 0.035	0.994 +/- 0.004	0.981 +/- 0.028	Sensitivity

C.10 DIAGNOSTIC RESULTS – K-NEAREST NEIGHBOR (KNN)

Table C.54 – Accuracy of K-Nearest Neighbor with different number of features

	Model	Scale Type	Training	Validation
0	K-Nearest Neighbor – All Features	Robust Scaler	0.971944	0.967442
1	K-Nearest Neighbor – 3 Features	Robust Scaler	0.975952	0.967442
2	K-Nearest Neighbor – 5 Features	Robust Scaler	0.977956	0.962791
3	K-Nearest Neighbor – 9 Features	Robust Scaler	0.979960	0.976744

Table C.55 – Accuracy of K-Nearest Neighbor with different *N Neighbors* parameter

	Model	Training	Validation
0	KNN 9Features – Default 5	0.979960	0.976744
1	KNN 9Features – 3	0.977956	0.972093
2	KNN 9Features – 7	0.979960	0.981395
3	KNN 9Features – 10	0.971944	0.981395
4	KNN 9Features – 15	0.965932	0.990698
5	KNN 9Features – 25	0.965932	0.990698
6	KNN 9Features – 38	0.961924	0.995349
7	KNN 9Features – 50	0.961924	0.995349
8	KNN 9Features – 75	0.963928	0.986047
9	KNN 9Features – 100	0.953908	0.981395
10	KNN 9Features – 224	0.889780	0.902326

Table C.56 – Accuracy of K-Nearest Neighbor with different *Weights* parameter

	Model	Training	Validation
0	KNN 9Features – Default <i>uniform</i>	0.979960	0.976744
1	KNN 9Features – distance	1.000000	0.976744

Table C.57 – Accuracy of K-Nearest Neighbor best model with the best parameters

	Model	Scale Type	Training	Validation
0	KNN – Best Parameters	Robust Scaler	0.979960	0.976744

Table C.58 – F1-Score, Accuracy, Specificity, Sensitivity of K-Nearest Neighbor of Final K-Nearest Neighbor model with 9 features

	Time	Training	Validation	Model
0	0.027 +/- 0.002	0.982 +/- 0.003	0.964 +/- 0.030	F1-Score
1	0.027 +/- 0.002	0.982 +/- 0.002	0.973 +/- 0.012	Accuracy
2	0.027 +/- 0.002	0.987 +/- 0.002	0.786 +/- 0.414	Specificity
3	0.027 +/- 0.002	0.978 +/- 0.005	0.962 +/- 0.027	Sensitivity

C.11 DIAGNOSTIC RESULTS – ENSEMBLE BAGGING CLASSIFIER (BC)

Table C.59 – Accuracy of Ensemble Bagging Classifier with different number of features

	Model	Scale Type	Training	Validation
0	Bagging Classifier – All Features	Robust Scaler	1.000000	0.962791
1	Bagging Classifier – 3 Features	Robust Scaler	0.995992	0.953488
2	Bagging Classifier – 5 Features	Robust Scaler	0.995992	0.953488
3	Bagging Classifier – 9 Features	Robust Scaler	0.995992	0.962791

Table C.60 – Accuracy of Ensemble Bagging Classifier with different *Base Estimator* parameter

	Model	Training	Validation
0	BC 9Features – Default None	0.993988	0.944186
1	BC 9Features – <i>KNeighborsClassifier()</i>	0.975952	0.986047
2	BC 9Features – <i>DummyClassifier()</i>	0.515030	0.483721
3	BC 9Features – <i>Perceptron (tol=1e-3)</i>	0.969940	0.990698
4	BC 9Features – <i>SVC (gamma="scale")</i>	0.971944	0.995349
5	BC 9Features – LR_model_3Features	0.975952	0.986047
6	BC 9Features – DT_final_model	0.979960	0.944186
7	BC 9Features – SVC_final_model	0.979960	0.986047
8	BC 9Features – RF_final_model	0.977956	0.958140

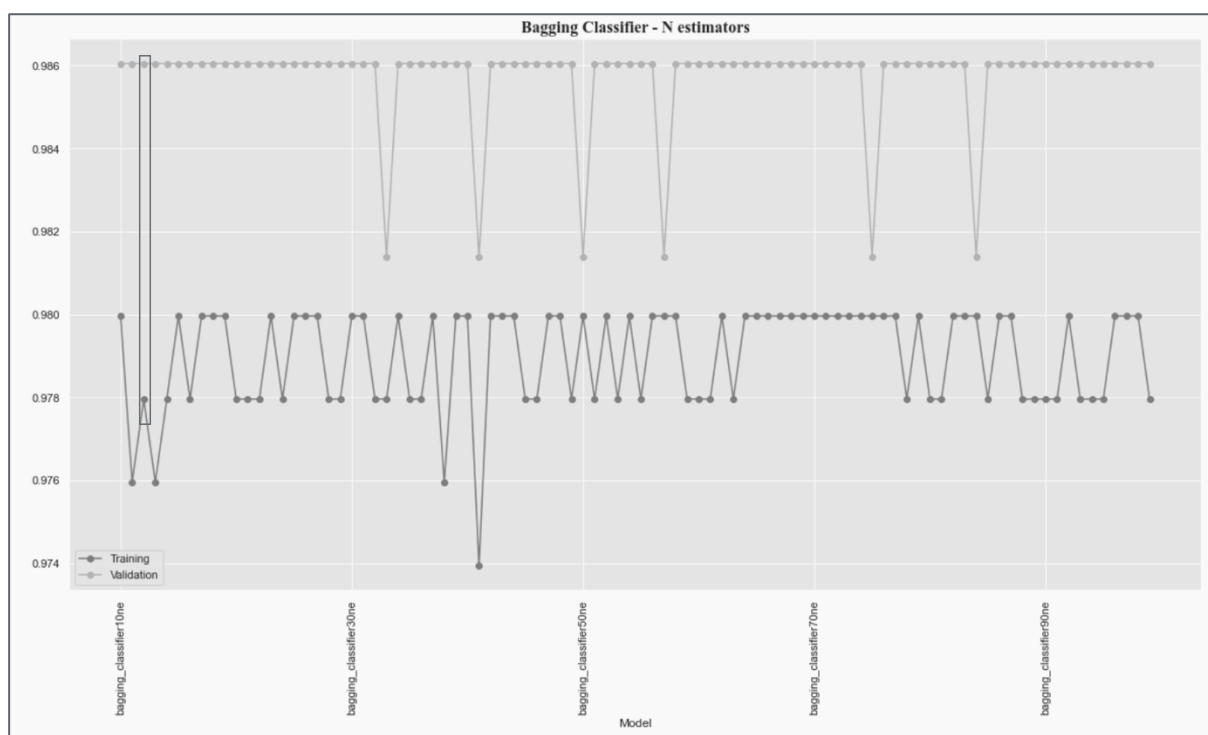


Figure C.1 – Accuracy of Ensemble Bagging Classifier with different *N Estimators* parameter

Table C.61 – Accuracy of Ensemble Bagging Classifier with different *Max Samples* parameter

	Model	Training	Validation
0	BC 9Features – Default 1.00	0.979960	0.986047
1	BC 9Features – 0.50	0.977956	0.986047
2	BC 9Features – 0.75	0.973948	0.986047
3	BC 9Features – 10	0.947896	0.972093
4	BC 9Features – 50	0.969940	0.990698
5	BC 9Features – 100	0.969940	0.990698
6	BC 9Features – 250	0.971944	0.986047
7	BC 9Features – 400	0.977956	0.986047
8	BC 9Features – 499	0.975952	0.986047

Table C.62 – Accuracy of Ensemble Bagging Classifier with different parameters

	Model	Training	Validation
0	BC 9Features – Default	0.975952	0.981395
1	BC 9Features – bootstrap = <i>False</i> , bootstrap_features = <i>True</i>	0.973948	0.986047
2	BC 9Features – oob_score = <i>True</i>	0.973948	0.986047
3	BC 9Features – warm_start = <i>True</i>	0.975952	0.990698

Table C.63 – Accuracy of Ensemble Bagging Classifier with different *Random State* parameter

	Model	Training	Validation
0	BC 9Features – Default None	0.975952	0.986047
1	BC 9Features – 1	0.997996	0.967442
2	BC 9Features – 6	1.000000	0.962791
3	BC 9Features – 11	0.997996	0.948837
4	BC 9Features – 16	1.000000	0.962791
5	BC 9Features – 21	0.997996	0.953488
6	BC 9Features – 26	0.997996	0.972093
7	BC 9Features – 31	0.995992	0.958140

8	BC 9Features – 36	0.997996	0.962791
9	BC 9Features – 41	1.000000	0.958140
10	BC 9Features – 46	0.995992	0.958140
11	BC 9Features – 51	0.997996	0.962791
12	BC 9Features – 56	0.997996	0.958140
13	BC 9Features – 61	0.991984	0.962791
14	BC 9Features – 66	1.000000	0.962791
15	BC 9Features – 71	0.995992	0.953488
16	BC 9Features – 76	0.997996	0.967442
17	BC 9Features – 81	0.995992	0.962791
18	BC 9Features – 86	1.000000	0.958140
19	BC 9Features – 91	1.000000	0.962791
20	BC 9Features – 96	0.997996	0.967442

Table C.64 – Accuracy of Ensemble Bagging Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Bagging Classifier – Best Parameters	Robust Scaler	0.979960	0.986047

Table C.65 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Bagging Classifier of Final Ensemble Bagging Classifier model with 9 features

	Time	Training	Validation	Model
0	0.060 +/- 0.005	0.980 +/- 0.003	0.963 +/- 0.022	F1-Score
1	0.060 +/- 0.005	0.980 +/- 0.002	0.972 +/- 0.009	Accuracy
2	0.060 +/- 0.005	0.991 +/- 0.004	0.785 +/- 0.414	Specificity
3	0.060 +/- 0.005	0.969 +/- 0.007	0.960 +/- 0.033	Sensitivity

C.12 DIAGNOSTIC RESULTS – ENSEMBLE ADABOOST CLASSIFIER (AC)

Table C.66 – Accuracy of Ensemble Adaboost Classifier with different number of features

	Model	Scale Type	Training	Validation
0	Adaboost Classifier – All Features	Robust Scaler	1.000000	0.976744
1	Adaboost Classifier – 3 Features	Robust Scaler	1.000000	0.967442
2	Adaboost Classifier – 5 Features	Robust Scaler	1.000000	0.953488
3	Adaboost Classifier – 9 Features	Robust Scaler	1.000000	0.981395

Table C.67 – Accuracy of Ensemble Adaboost Classifier with different *Base Estimator* parameter

	Model	Training	Validation
0	AC All Features – Default None	1.000000	0.976744
1	AC All Features – <i>DecisionTreeClassifier()</i>	1.000000	0.948837
2	AC All Features – <i>DecisionTreeClassifier((class_weight='balanced'))</i>	1.000000	0.925581
3	AC All Features – <i>LogisticRegression(solver='liblinear'))</i>	0.973948	0.986047
4	AC All Features – LR_model_newton_3Features	0.973948	0.986047
5	AC All Features – DT_model_final	1.000000	0.972093
6	AC All Features – (algorithm = 'SAMME', base_estimator = SVC_model_final)	0.925852	0.906977

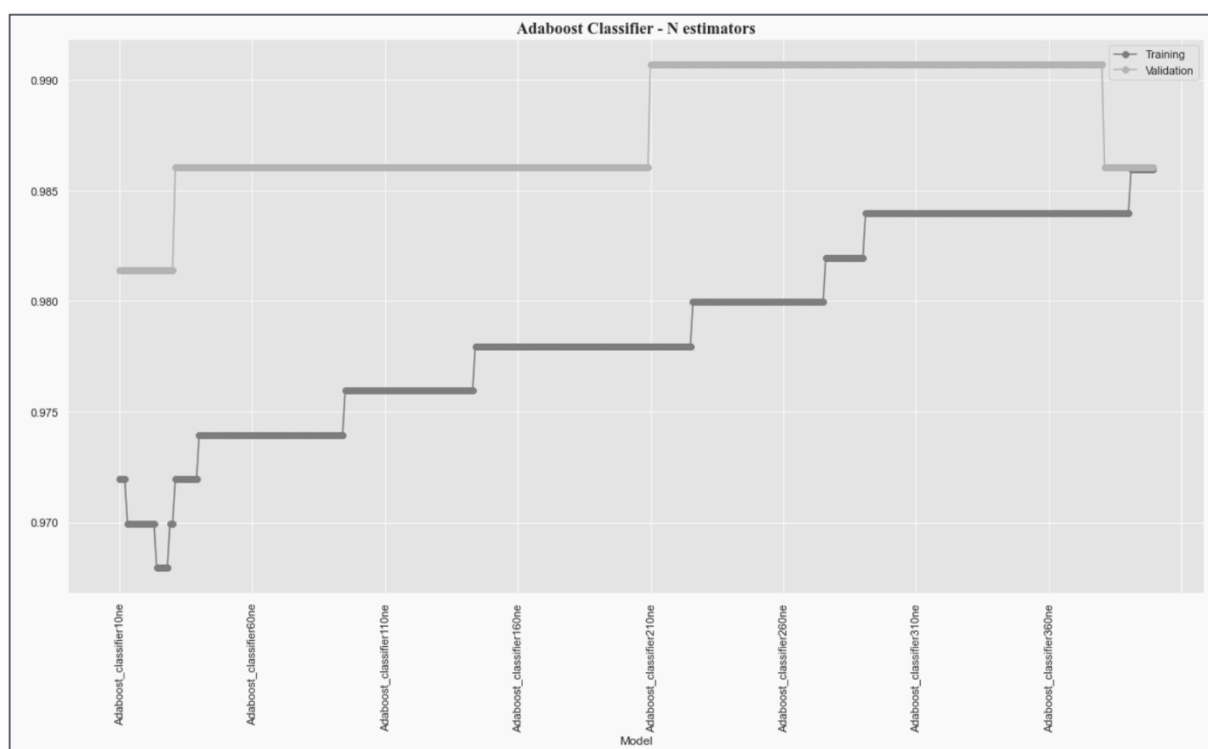


Figure C.2 – Accuracy of Ensemble Adaboost Classifier with different *N Estimators* parameter

Table C.68 – Accuracy of Ensemble Adaboost Classifier with different *Learning Rate* parameter

	Model	Training	Validation
0	AC All Features – 0.10	0.973948	0.986047
1	AC All Features – 0.20	0.973948	0.986047
2	AC All Features – 0.3	0.975952	0.986047
3	AC All Features – 0.40	0.977956	0.986047
4	AC All Features – 0.50	0.977956	0.986047
5	AC All Features – 0.60	0.979960	0.990698
6	AC All Features – 0.70	0.981964	0.990698
7	AC All Features – 0.80	0.983968	0.990698
8	AC All Features – 0.90	0.983968	0.990698
9	AC All Features – Default 1.00	0.985972	0.986047

Table C.69 – Accuracy of Ensemble Adaboost Classifier with different *Algorithm* parameter

	Model	Training	Validation
0	AC All Features – Default <i>SAMME.R</i>	0.985972	0.986047
1	AC All Features – <i>SAMME</i>	0.991984	0.976744

Table C.70 – Accuracy of Ensemble Adaboost Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Adaboost Classifier – Best Parameters	Robust Scaler	0.985972	0.986047

Table C.71 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Adaboost Classifier of Final Ensemble Adaboost Classifier model with all features

	Time	Training	Validation	Model
0	1.369 +/- 0.095	0.989 +/- 0.002	0.969 +/- 0.017	F1-Score
1	1.369 +/- 0.095	0.989 +/- 0.002	0.975 +/- 0.014	Accuracy
2	1.369 +/- 0.095	0.992 +/- 0.003	0.778 +/- 0.410	Specificity
3	1.369 +/- 0.095	0.985 +/- 0.003	0.973 +/- 0.030	Sensitivity

C.13 DIAGNOSTIC RESULTS – ENSEMBLE VOTING CLASSIFIER (VC)

```
#To define the estimators
estimators = [('dt', DT_model_final), ('svc', SVC_model_final)]

estimators_1 = [('nv', modelNB), ('lr', LR_model_newton_3Features), ('rf', RF_model_final), ('dt', DT_model_final), ('n
estimators_2 = [('gb', gbc_model_final), ('bc', BC_model_final), ('svc', SVC_model_final)]
estimators_3 = [('gb', gbc_model_final), ('dt', DT_model_final), ('svc', SVC_model_final)]
estimators_4 = [('gb', gbc_model_final), ('ac', AdC_model_final), ('bc', BC_model_final)]
estimators_5 = [('gb', gbc_model_final), ('bc', BC_model_final)]
estimators_6 = [('ac', AdC_model_final), ('bc', BC_model_final)]
estimators_7 = [('lr', LR_model_newton_3Features), ('bc', BC_model_final)]
estimators_8 = [('ac', AdC_model_final), ('lr', LR_model_newton_3Features)]
estimators_9 = [('svc', SVC_model_final), ('bc', BC_model_final)]
estimators_10 = [('ac', AdC_model_final), ('svc', SVC_model_final)]
estimators_11 = [('ac', AdC_model_final), ('svc', SVC_model_final), ('bc', BC_model_final)]
estimators_12 = [('ac', AdC_model_final), ('svc', SVC_model_final), ('bc', BC_model_final), ('lr', LR_model_newton_3Featu
```

Figure C.3 – Estimators used in the Voting Classifier model

Table C.72 – Accuracy of Ensemble Voting Classifier with different number of features

	Model	Scale Type	Training	Validation
0	Voting Classifier – All Features	Robust Scaler	0.993988	0.995349
1	Voting Classifier – 3 Features	Robust Scaler	0.981964	0.967442
2	Voting Classifier – 5 Features	Robust Scaler	0.977956	0.967442
3	Voting Classifier – 9 Features	Robust Scaler	0.979960	0.986047

Table C.73 – Accuracy of Ensemble Voting Classifier with different *Weights* parameter

	Model	Training	Validation
0	VC All Features – Default None	0.993988	0.995349
1	VC All Features – [1,1,2]	0.993988	0.995349
2	VC All Features – [2,1,2]	0.993988	0.995349
3	VC All Features – [1,2,2]	0.993988	0.995349
4	VC All Features – [2,0,0]	1.000000	0.972093
5	VC All Features – [0,2,0]	0.985972	0.986047
6	VC All Features – [0,0,2]	0.993988	0.995349

Table C.74 – Accuracy of Ensemble Voting Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Voting Classifier – Best Parameters	Robust Scaler	0.993988	0.995349

Table C.75 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Voting Classifier of Final Ensemble Voting Classifier model with all features

	Time	Training	Validation	Model
0	1.925 +/- 0.166	0.994 +/- 0.001	0.975 +/- 0.021	F1-Score
1	1.925 +/- 0.166	0.994 +/- 0.001	0.980 +/- 0.012	Accuracy
2	1.925 +/- 0.166	1.000 +/- 0.000	0.782 +/- 0.413	Specificity
3	1.925 +/- 0.166	0.989 +/- 0.003	0.980 +/- 0.028	Sensitivity

C.14 DIAGNOSTIC RESULTS – ENSEMBLE GRADIENT BOOSTING CLASSIFIER (GBC)

Table C.76 – Accuracy of Ensemble Gradient Boosting Classifier with different number of features

	Model	Scale Type	Training	Validation
0	Gradient Boosting Classifier – All Features	Robust Scaler	1.000000	0.967442
1	Gradient Boosting Classifier – 3 Features	Robust Scaler	1.000000	0.958140
2	Gradient Boosting Classifier – 5 Features	Robust Scaler	1.000000	0.958140
3	Gradient Boosting Classifier – 9 Features	Robust Scaler	1.000000	0.967442

Table C.77 – Accuracy of Ensemble Gradient Boosting Classifier with different *Min Impurity Decrease* parameter

	Model	Training	Validation
0	GBC 9Features – Default 0.0	1.000000	0.981395
1	GBC 9Features – 1e-06	1.000000	0.981395
2	GBC 9Features – 1e-07	1.000000	0.976744
3	GBC 9Features – 1e-08	1.000000	0.981395

Table C.78 – Accuracy of Ensemble Gradient Boosting Classifier with different *Loss* parameter

	Model	Training	Validation
0	GBC 9Features – Default <i>log_loss</i>	1.000000	0.972093
1	GBC 9Features – <i>exponential</i>	1.000000	0.972093

Table C.79 – Accuracy of Ensemble Gradient Boosting Classifier with different *Learning Rate* parameter

	Model	Training	Validation
0	GBC 9Features – Default 0.1000	1.000000	0.967442
1	GBC 9Features – 0.2125	1.000000	0.972093

2	GBC 9Features – 0.3250	1.000000	0.976744
3	GBC 9Features – 0.4375	1.000000	0.976744
4	GBC 9Features – 0.5500	1.000000	0.981395
5	GBC 9Features – 0.6625	1.000000	0.972093
6	GBC 9Features – 0.7750	1.000000	0.972093
7	GBC 9Features – 0.8875	1.000000	0.972093
8	GBC 9Features – 1.0000	1.000000	0.967442

Table C.80 – Accuracy of Ensemble Gradient Boosting Classifier with different *Subsample* parameter

	Model	Training	Validation
0	GBC 9Features – Default 1.0	1.000000	0.981395
1	GBC 9Features – 0.25	0.995992	0.958140
2	GBC 9Features – 0.50	1.000000	0.972093
3	GBC 9Features – 0.70	1.000000	0.972093
4	GBC 9Features – 0.75	1.000000	0.962791
5	GBC 9Features – 0.80	1.000000	0.976744
6	GBC 9Features – 0.90	1.000000	0.972093

Table C.81 – Accuracy of Ensemble Gradient Boosting Classifier with different *Criterion* parameter

	Model	Training	Validation
0	GBC 9Features – Default <i>friedman_mse</i>	1.000000	0.981395
1	GBC 9Features – <i>mse</i>	1.000000	0.981395
2	GBC 9Features – <i>mae</i>	0.975952	0.925581

Table C.82 – Accuracy of Ensemble Gradient Boosting Classifier with different *Max Depth* parameter

	Model	Training	Validation
0	GBC 9Features – Default 3	1.000000	0.981395
1	GBC 9Features – 1	1.000000	0.972093
2	GBC 9Features – 2	1.000000	0.967442
3	GBC 9Features – 4	1.000000	0.967442

4	GBC 9Features – 6	1.000000	0.967442
5	GBC 9Features – 10	1.000000	0.967442

Table C.83 – Accuracy of Ensemble Gradient Boosting Classifier with different *Min Samples Leaf* parameter

	Model	Training	Validation
0	GBC 9Features – Default 1	1.000000	0.981395
1	GBC 9Features – 8	1.000000	0.976744
2	GBC 9Features – 50	1.000000	0.976744
3	GBC 9Features – 30	1.000000	0.981395
4	GBC 9Features – 60	1.000000	0.972093
5	GBC 9Features – 75	1.000000	0.981395
6	GBC 9Features – 80	1.000000	0.986047
7	GBC 9Features – 100	1.000000	0.972093
8	GBC 9Features – 500	1.000000	0.497674

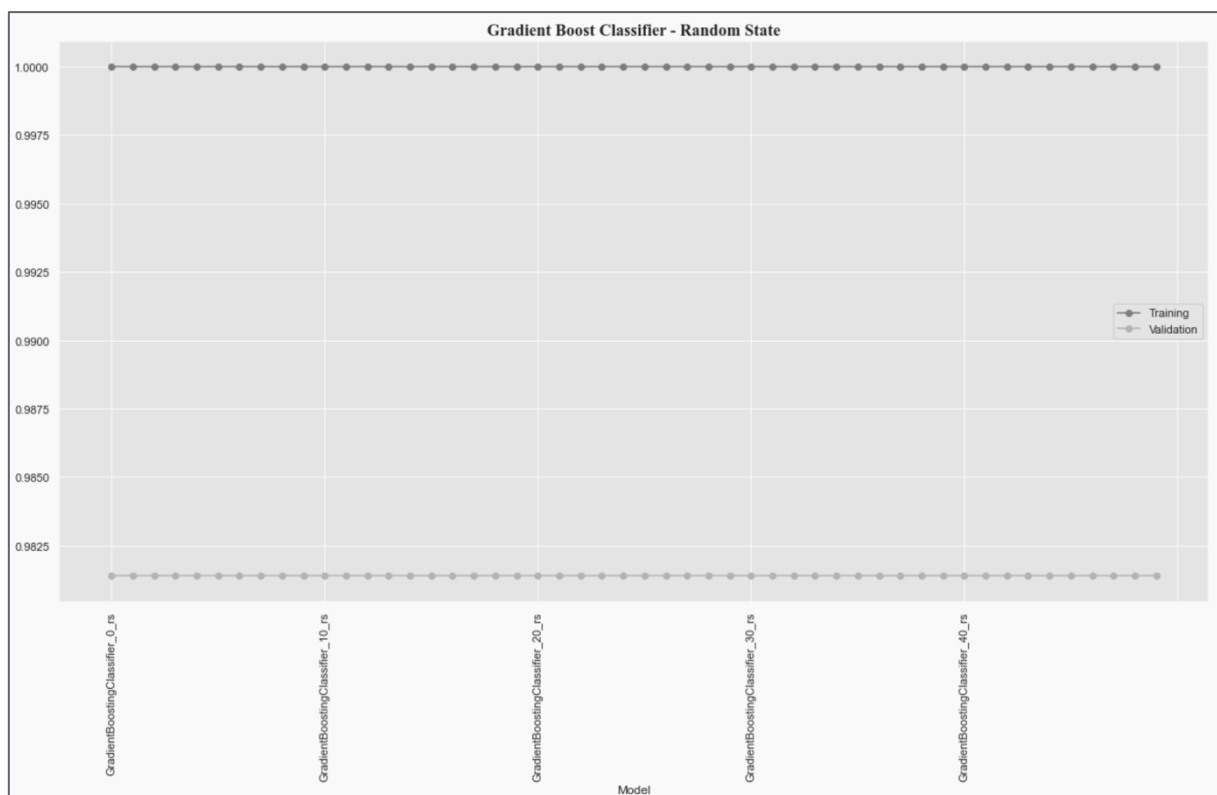


Figure C.4 – Accuracy of Ensemble Gradient Boosting Classifier with different *Random State* parameter

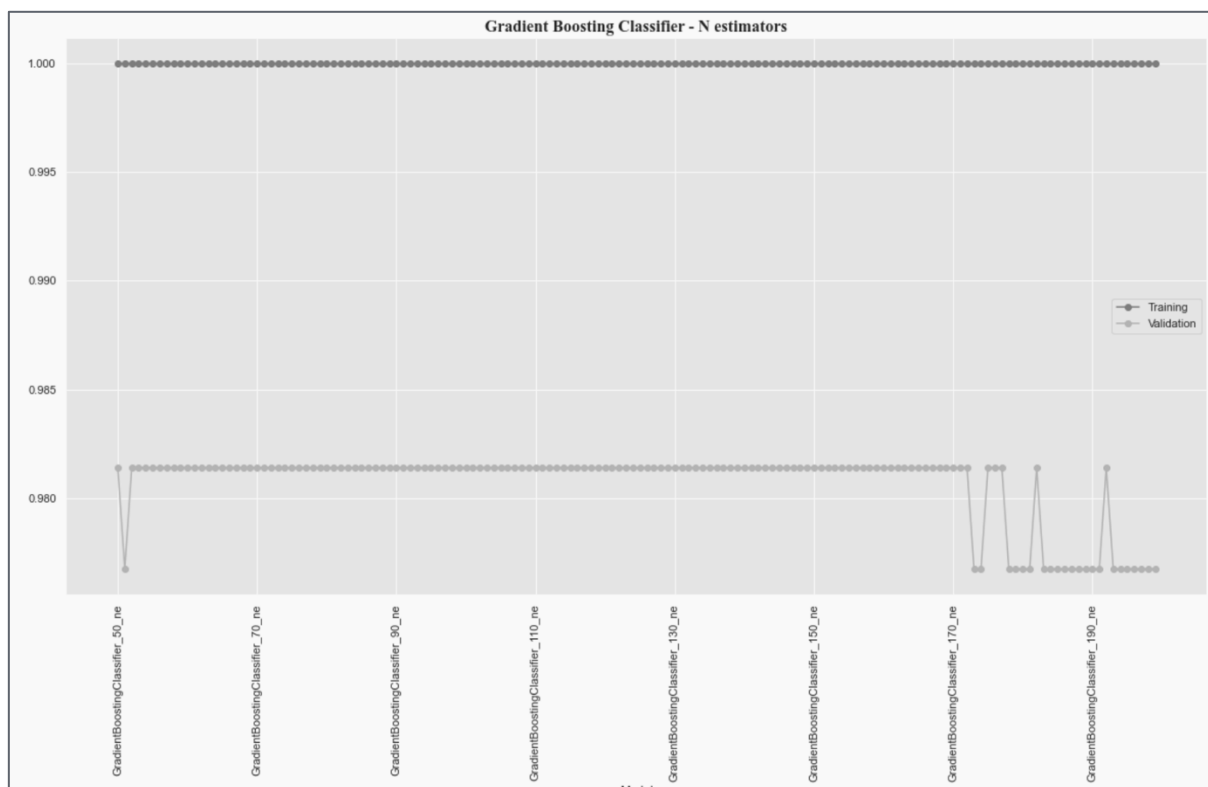


Figure C.5 – Accuracy of Ensemble Gradient Boosting Classifier with different *N Estimators* parameter

Table C.84 – Accuracy of Ensemble Gradient Boosting Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Gradient Boosting Classifier – Best Parameters	Robust Scaler	1.000000	0.981395

Table C.85 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Gradient Boosting Classifier of Final Ensemble Gradient Boosting Classifier model with 9 features

	Time	Training	Validation	Model
0	0.194 +/- 0.025	1.000 +/- 0.000	0.955 +/- 0.031	F1-Score
1	0.194 +/- 0.025	1.000 +/- 0.000	0.966 +/- 0.021	Accuracy
2	0.194 +/- 0.025	1.000 +/- 0.000	0.774 +/- 0.408	Specificity
3	0.194 +/- 0.025	1.000 +/- 0.000	0.963 +/- 0.047	Sensitivity

C.15 DIAGNOSTIC RESULTS – ENSEMBLE STACKING CLASSIFIER (SC)

```
#To define the estimators
estimators = [('dt', DT_model_final), ('svc', SVC_model_final)]

estimators_1 = [('nv', modelNB), ('lr', LR_model_newton_3Features), ('rf', RF_model_final), ('dt', DT_model_final), ('n
estimators_2 = [('gb', gbc_model_final), ('bc', BC_model), ('svc', SVC_model_final)]
estimators_3 = [('gb', gbc_model_final), ('dt', DT_model_final), ('svc', SVC_model_final)]
estimators_4 = [('gb', gbc_model_final), ('ac', AdC_model_final), ('bc', BC_model)]
estimators_5 = [('gb', gbc_model_final), ('bc', BC_model)]
estimators_6 = [('ac', AdC_model_final), ('bc', BC_model)]
estimators_7 = [('lr', LR_model_newton_3Features), ('gc', BC_model)]
estimators_8 = [('ac', AdC_model_final), ('lr', LR_model_newton_3Features)]
estimators_9 = [('svc', SVC_model_final), ('gc', BC_model)]
estimators_10 = [('ac', AdC_model_final), ('svc', SVC_model_final)]
estimators_11 = [('ac', AdC_model_final), ('svc', SVC_model_final), ('gc', BC_model)]
estimators_12 = [('ac', AdC_model_final), ('svc', SVC_model_final), ('gc', BC_model), ('lr', LR_model_newton_3Features)]
estimators_13 = [('gc', BC_model), ('dt', DT_model_final)]
estimators_14 = [('gc', BC_model), ('ac', AdC_model_final)]
```

Figure C.6 – Estimators used in the Stacking Classifier model

Table C.86 – Accuracy of Ensemble Stacking Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Stacking Classifier – Best Parameters	Robust Scaler	0.995992	0.995349

Table C.87 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Stacking Classifier of Final Ensemble Stacking Classifier model

	Time	Training	Validation	Model
0	2.975 +/- 0.088	0.996 +/- 0.002	0.980 +/- 0.015	F1-Score
1	2.975 +/- 0.088	0.996 +/- 0.002	0.983 +/- 0.011	Accuracy
2	2.975 +/- 0.088	1.000 +/- 0.000	0.791 +/- 0.417	Specificity
3	2.975 +/- 0.088	0.991 +/- 0.004	0.977 +/- 0.027	Sensitivity

C.16 DIAGNOSTIC RESULTS – ENSEMBLE EXTREME GRADIENT BOOSTING CLASSIFIER (XGBC w/ GS)

Table C.88 – Accuracy of Ensemble Extreme Gradient Boosting Classifier with Grid Search with different number of features

	Model	Scale Type	Training	Validation
0	XGBC w/ GS – All Features	Robust Scaler	1.000000	0.962791
1	XGBC w/ GS – 3 Features	Robust Scaler	1.000000	0.962791
2	XGBC w/ GS – 5 Features	Robust Scaler	1.000000	0.958140
3	XGBC w/ GS – 9 Features	Robust Scaler	1.000000	0.986047

Table C.89 – *Parameter Space*

Parameter	Values
<i>max_depth</i>	[6, 0, 3, 10, None, 17]
<i>learning_rate</i>	[0.3, 0.01, 0.1, 0.5, 0.75, 1.0]
<i>n_estimators</i>	[100, 3, 5, 50, 75, 200, 750, 1000]
<i>subsample</i>	[1, 0.1, 0.25, 0.5, 0.8]
<i>colsample_bytree</i>	[1, 0.2, 0.5, 0.8]

Table C.90 – Best Parameters found from *Parameter Space* with *Grid Search*

Parameter	Best Value
<i>max_depth</i>	6
<i>learning_rate</i>	0.3
<i>n_estimators</i>	100
<i>subsample</i>	1
<i>colsample_bytree</i>	1

Table C.91 – Accuracy of Ensemble Extreme Gradient Boosting Classifier with Grid Search best model with the best parameters

Model	Scale Type	Training	Validation
-------	------------	----------	------------

0	XGBC w/ GS – Best Parameters	Robust Scaler	1.000000	0.986047
---	------------------------------	---------------	----------	----------

Table C.92 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Extreme Gradient Boosting Classifier with Grid Search of Final Ensemble Extreme Gradient Boosting Classifier with Grid Search model with 9 features

	Time	Training	Validation	Model
0	0.075 +/- 0.042	1.000 +/- 0.000	0.953 +/- 0.037	F1-Score
1	0.075 +/- 0.042	1.000 +/- 0.000	0.966 +/- 0.021	Accuracy
2	0.075 +/- 0.042	1.000 +/- 0.000	0.775 +/- 0.409	Specificity
3	0.075 +/- 0.042	1.000 +/- 0.000	0.962 +/- 0.045	Sensitivity

C.17 DIAGNOSTIC RESULTS – TREE-BASED PIPELINE OPTIMIZATION TOOL (TPOT)

Table C.93 – Accuracy of TPOT with different number of features (default parameters)

	Model	Scale Type	Training	Validation
0	TPOT – All Features	Robust Scaler	1.000000	0.962791
1	TPOT – 3 Features	Robust Scaler	1.000000	0.948837
2	TPOT – 5 Features	Robust Scaler	0.993988	0.976744
3	TPOT – 9 Features	Robust Scaler	0.983968	0.976744

Table C.94 – Accuracy of TPOT with 20 *Generations* parameter

	Model	Scale Type	Training	Validation
0	TPOT – All Features	Robust Scaler	1.000000	0.986040
1	TPOT – 3 Features	Robust Scaler	1.000000	0.976743
2	TPOT – 5 Features	Robust Scaler	1.000000	0.976743
3	TPOT – 9 Features	Robust Scaler	1.000000	0.976743

Table C.95 – Accuracy of TPOT with 50 *Generations* parameter

	Model	Scale Type	Training	Validation
0	TPOT – All Features	Robust Scaler	1.000000	0.967440
1	TPOT – 3 Features	Robust Scaler	1.000000	0.976743
2	TPOT – 5 Features	Robust Scaler	0.971944	0.972093
3	TPOT – 9 Features	Robust Scaler	1.000000	0.986047

Table C.96 – Accuracy of TPOT with 100 *Generations* parameter

	Model	Scale Type	Training	Validation
0	TPOT – All Features	Robust Scaler	1.000000	0.972089
1	TPOT – 3 Features	Robust Scaler	1.000000	0.953482
2	TPOT – 5 Features	Robust Scaler	1.000000	0.976738
3	TPOT – 9 Features	Robust Scaler	0.997996	0.976743

Table C.97 – Accuracy of TPOT with different *Population Size* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 100	0.983968	0.976744
1	TPOT 9Features – 25	0.985972	0.976744
2	TPOT 9Features – 10	0.979960	0.976744
3	TPOT 9Features – 50	0.983968	0.986047
4	TPOT 9Features – 75	0.981964	0.986047
5	TPOT 9Features – 95	0.983968	0.976744

Table C.98 – Accuracy of TPOT with different *Mutation Rate* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 0.9	0.987976	0.981395
1	TPOT 9Features – 0.1	0.967936	0.986047
2	TPOT 9Features – 0.3	0.983968	0.981395
3	TPOT 9Features – 0.5	1.000000	0.986047
4	TPOT 9Features – 0.7	0.987976	0.972093

Table C.99 – Accuracy of TPOT with different *Crossover Rate* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 0.1	0.973948	0.981395
1	TPOT 9Features – 0.3	0.985972	0.986047
2	TPOT 9Features – 0.4	0.983968	0.976744
3	TPOT 9Features – 0.5	0.973948	0.986047
4	TPOT 9Features – 0.7	1.000000	0.962791

Table C.100 – Accuracy of TPOT with different *cv* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 5	1.000000	0.986047
1	TPOT 9Features – 3	0.969940	0.981395
2	TPOT 9Features – 7	1.000000	0.958140
3	TPOT 9Features – 10	0.983968	0.976744
4	TPOT 9Features – 25	1.000000	0.962791
5	TPOT 9Features – 50	1.000000	0.976744

6	TPOT 9Features – 75	0.985972	0.972093
7	TPOT 9Features – 250	0.977956	0.986047

Table C.101 – Accuracy of TPOT with different *Random State* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 10	0.983968	0.976744
1	TPOT 9Features – 15	0.985972	0.972093
2	TPOT 9Features – 42	0.983968	0.976744
3	TPOT 9Features – 123	0.987976	0.962791

```
HBox(children=(FloatProgress(value=0.0, description='Optimization Progress', max=450.0, style=ProgressStyle(de...

Generation 1 - Current best internal CV score: 0.977959595959596
Generation 2 - Current best internal CV score: 0.977959595959596
Generation 3 - Current best internal CV score: 0.977959595959596
Generation 4 - Current best internal CV score: 0.977959595959596
Generation 5 - Current best internal CV score: 0.977959595959596
Best pipeline: MLPClassifier(input_matrix, alpha=0.01, learning_rate_init=0.001)
```

Figure C.7 – Accuracy of TPOT best model with the best parameters

Table C.102 – F1-Score, Accuracy, Specificity, Sensitivity of TPOT of Final TPOT Classifier model

	Time	Training	Validation	Model
0	220.702 +/- 48.739	0.988 +/- 0.007	0.958 +/- 0.027	F1-Score
1	220.702 +/- 48.739	0.988 +/- 0.007	0.968 +/- 0.011	Accuracy
2	220.702 +/- 48.739	0.991 +/- 0.005	0.773 +/- 0.408	Specificity
3	220.702 +/- 48.739	0.985 +/- 0.001	0.974 +/- 0.027	Sensitivity

C.18 DIAGNOSTIC RESULTS – FINAL RESULTS

Table C.103 – Accuracy of all models

	Model	Training	Validation
0	Gaussian Naïve Bayes	0.965932	0.972033
1	Logistic Regression	0.975952	0.986047
2	Decision Tree	0.977956	0.958140
3	Random Forest	0.985972	0.962791
4	Neural Network	0.973948	0.976744
5	Probabilistic Neural Network	0.989980	0.986046
6	Linear Discriminant Analysis	0.975952	0.972093
7	Support Vector Machine	0.971944	0.972093
8	Multi-Layer Perceptron with Grid Search	0.993988	0.995349
9	K-Nearest Neighbors	0.979960	0.976744
10	Bagging Classifier	0.979960	0.986047
11	Adaboost Classifier	0.985972	0.986047
12	Voting Classifier	0.993988	0.995349
13	Gradient Boosting Classifier	1.000000	0.981395
14	Stacking Classifier	0.995992	0.995349
15	Extreme Gradient Boosting Classifier	1.000000	0.986047
16	TPOT	0.983968	0.981395

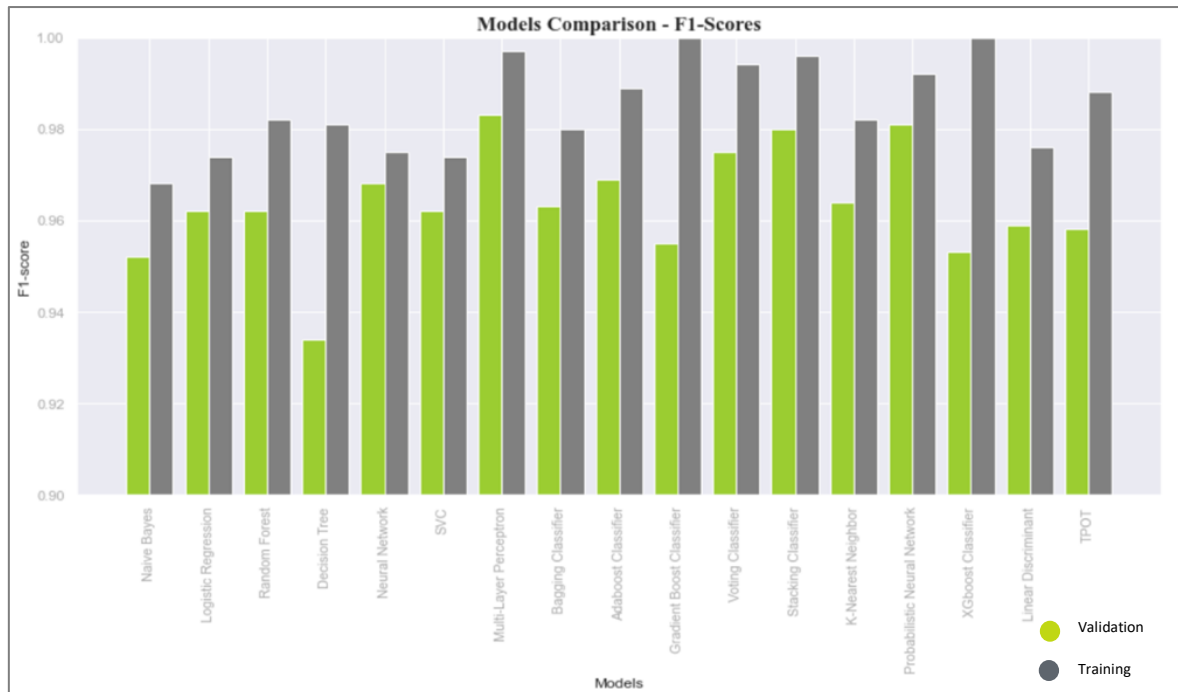


Figure C.8 – F-1 Scores of all models

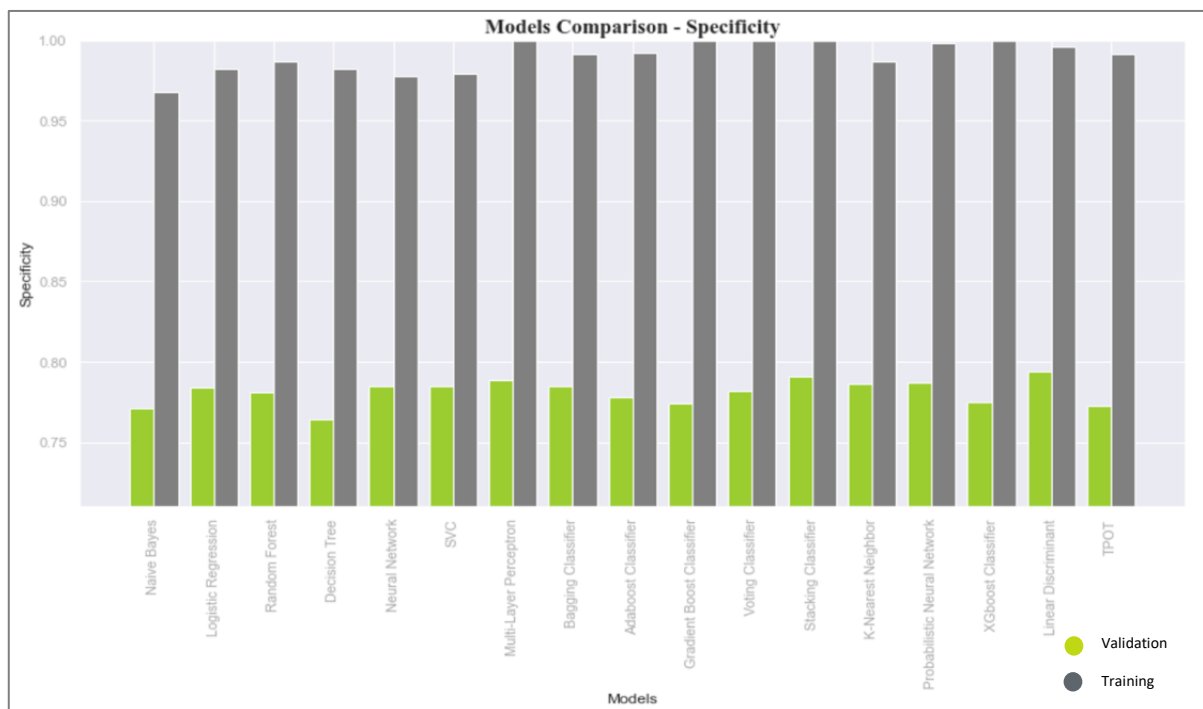


Figure C.9 – Specificity of all models

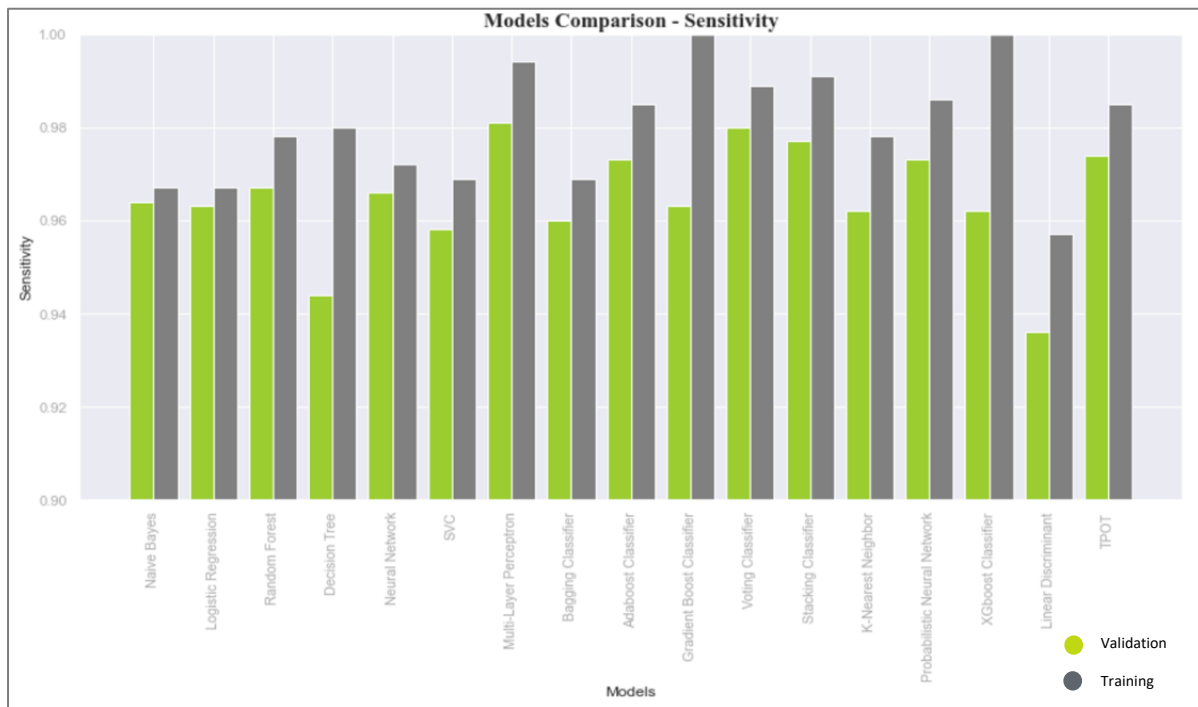


Figure C.10 – Sensitivity of all models

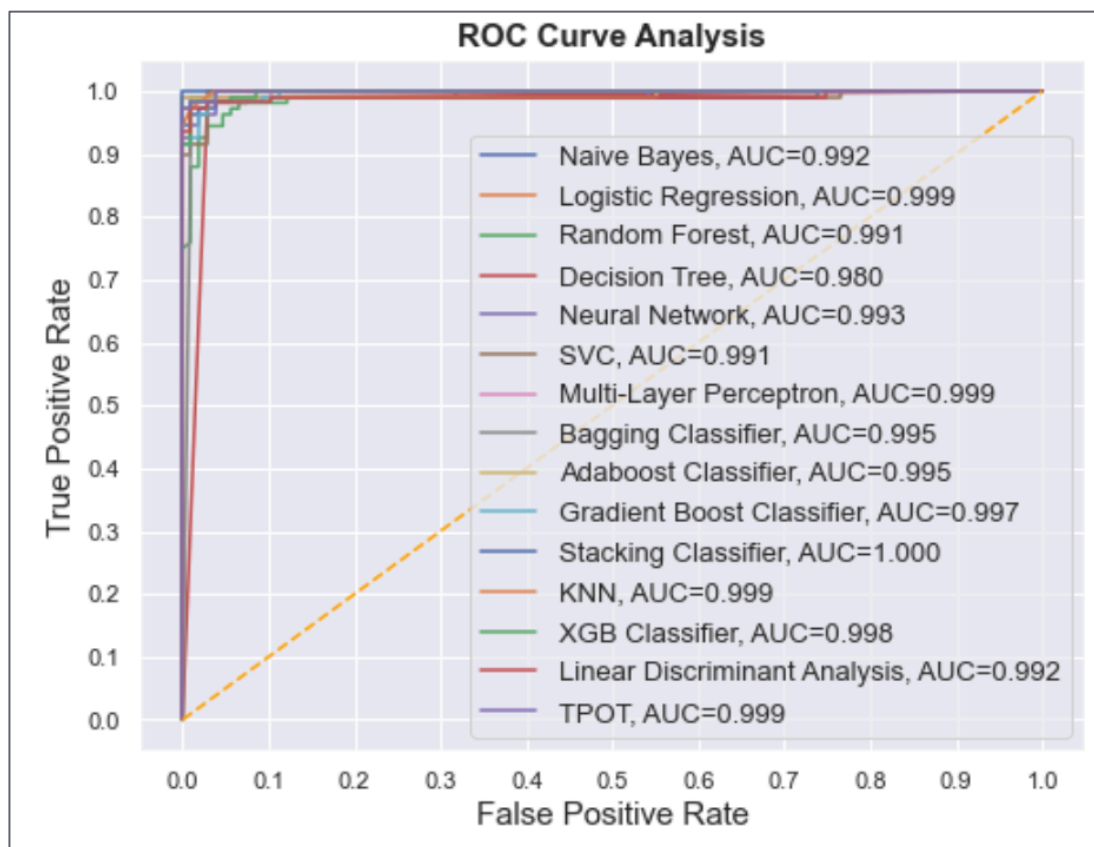


Figure C.11 – ROC Curve

D.1 PROGNOSTIC RESULTS – CLASSIFICATION PROBLEM – NAÏVE BAYES

Table D.1 – Accuracy of Naïve Bayes with different numbers of features

	Model	Scale Type	Training	Validation
0	Naïve Bayes – All Features	Robust Scaler	0.701923	0.666667
1	Naïve Bayes – 3 Features	Robust Scaler	0.668269	0.666667
2	Naïve Bayes – 3 F – no <i>Size</i>	Robust Scaler	0.668269	0.666667
3	Naïve Bayes – 4 Features	Robust Scaler	0.629808	0.644444
4	Naïve Bayes – 5 Features	Robust Scaler	0.711538	0.700000
3	Naïve Bayes – 9 Features	Robust Scaler	0.764423	0.655556

Table D.2 – F1-Score, Accuracy, Specificity, Sensitivity of Naïve Bayes of Final Naïve Bayes model with 5 features

	Time	Training	Validation	Model
0	0.005 +/- 0.001	0.700 +/- 0.020	0.712 +/- 0.210	F1-Score
1	0.005 +/- 0.001	0.705 +/- 0.021	0.682 +/- 0.196	Accuracy
2	0.005 +/- 0.001	0.569 +/- 0.030	0.390 +/- 0.324	Specificity
3	0.005 +/- 0.001	0.839 +/- 0.030	0.785 +/- 0.157	Sensitivity

D.2 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – LOGISTIC REGRESSION (LR)

Table D.3 – Accuracy of Logistic Regression with different numbers of features

	Model	Scale Type	Training	Validation
0	Logistic Regression – All Features	Robust Scaler	0.778846	0.722222
1	Logistic Regression – 3 Features	Robust Scaler	0.673077	0.600000
2	Logistic Regression – 3 F – no Size	Robust Scaler	0.673077	0.600000
3	Logistic Regression – 4 Features	Robust Scaler	0.581731	0.633333
4	Logistic Regression – 5 Features	Robust Scaler	0.668269	0.666667
5	Logistic Regression – 9 Features	Robust Scaler	0.692308	0.655556

Table D.4 – Accuracy of Logistic Regression with different parameters

	Model	Scale Type	Training	Validation
0	LR 5Features – Newton	Robust Scaler	0.668269	0.666667
1	LR 5Features – Lib	Robust Scaler	0.668269	0.666667
2	LR 5Features – Sag	Robust Scaler	0.668269	0.666667
3	LR 5Features – Saga	Robust Scaler	0.668269	0.666667

Table D.5 – F1-Score, Accuracy, Specificity, Sensitivity of Logistic Regression of Final Logistic Regression model with 5 features

	Time	Training	Validation	Model
0	0.010 +/- 0.001	0.644 +/- 0.031	0.626 +/- 0.134	F1-Score
1	0.010 +/- 0.001	0.648 +/- 0.027	0.563 +/- 0.166	Accuracy
2	0.010 +/- 0.001	0.665 +/- 0.072	0.444 +/- 0.361	Specificity
3	0.010 +/- 0.001	0.616 +/- 0.130	0.564 +/- 0.179	Sensitivity

D.3 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – DECISION TREE (DT)

Table D.6 – Accuracy of Decision Tree with different number of features

	Model	Scale Type	Training	Validation
0	Decision Tree – All Features	Robust Scaler	1.000000	0.777778
1	Decision Tree – 3 Features	Robust Scaler	1.000000	0.655556
2	Decision Tree – 3 F – no Size	Robust Scaler	1.000000	0.688889
3	Decision Tree – 4 Features	Robust Scaler	1.000000	0.700000
4	Decision Tree – 5 Features	Robust Scaler	1.000000	0.733333
5	Decisin Tree – 9 Features	Robust Scaler	1.000000	0.788889

Table D.7 – Accuracy of Decision Tree with different *Criterion* parameter

	Model	Training	Validation
0	DT 9Features – Default Gini	1.000000	0.800000
1	DT 9Features – Entropy	1.000000	0.733333

Table D.8 – Accuracy of Decision Tree with different *Splitter* parameter

	Model	Training	Validation
0	DT 9Features – Default Best	1.000000	0.800000
1	DT 9Features – Random	1.000000	0.711111

Table D.9 – Accuracy of Decision Tree with different *Max Depth* parameter

	Model	Training	Validation
0	DT 9Features – Default None	1.000000	0.777778
1	DT 9Features – 5	0.899038	0.722222
2	DT 9Features – 10	0.985577	0.744444
3	DT 9Features – 20	1.000000	0.777778
4	DT 9Features – 27	1.000000	0.811111
5	DT 9Features – 50	1.000000	0.766667
6	DT 9Features – 68	1.000000	0.788889
7	DT 9Features – 100	1.000000	0.811111

Table D.10 – Accuracy of Decision Tree with different *Min Samples Split* parameter

	Model	Training	Validation
0	DT 9Features – Default 2	0.899038	0.711111
1	DT 9Features – 5	0.899038	0.733333
2	DT 9Features – 10	0.894231	0.733333
3	DT 9Features – 20	0.831731	0.744444
4	DT 9Features – 43	0.826923	0.722222
5	DT 9Features – 100	0.793269	0.733333
6	DT 9Features – 250	0.500000	0.500000
7	DT 9Features – 500	0.500000	0.500000
8	DT 9Features – 1000	0.500000	0.500000

Table D.11 – Accuracy of Decision Tree with different *Min Samples Leaf* parameter

	Model	Training	Validation
0	DT 9Features – Default 1	0.793269	0.733333
1	DT 9Features – 5	0.793269	0.733333
2	DT 9Features – 10	0.793269	0.733333
3	DT 9Features – 50	0.692308	0.633333
4	DT 9Features – 75	0.687500	0.622222
5	DT 9Features – 100	0.682692	0.700000
6	DT 9Features – 250	0.500000	0.500000
7	DT 9Features – 500	0.500000	0.500000

Table D.12 – Accuracy of Decision Tree with different *Min Weight Fraction Leaf* parameter

	Model	Training	Validation
0	DT 9Features – Default 0.0	0.682692	0.700000
1	DT 9Features – 0.08	0.682692	0.700000
2	DT 9Features – 0.15	0.682692	0.700000
3	DT 9Features – 0.30	0.682692	0.700000
4	DT 9Features – 0.50	0.663462	0.600000

Table D.13 – Accuracy of Decision Tree with different *Min Impurity Decrease* parameter

	Model	Training	Validation
0	DT 9Features – Default 0.0	0.682692	0.700000
1	DT 9Features – 0.015	0.682692	0.700000
2	DT 9Features – 0.020	0.682692	0.700000
3	DT 9Features – 0.025	0.682692	0.700000

Table D.14 – Accuracy of Decision Tree best model with the best parameters

	Model	Scale Type	Training	Validation
0	Decision Tree – Best Parameters	Robust Scaler	0.6882692	0.700000

Table A.15 – F1-Score, Accuracy, Specificity, Sensitivity of Decision Tree of Final Decision Tree model with 9 features

	Time	Training	Validation	Model
0	0.004 +/- 0.000	0.688 +/- 0.010	0.729 +/- 0.115	F1-Score
1	0.004 +/- 0.000	0.688 +/- 0.010	0.685 +/- 0.086	Accuracy
2	0.004 +/- 0.000	0.691 +/- 0.010	0.485 +/- 0.338	Specificity
3	0.004 +/- 0.000	0.684 +/- 0.016	0.634 +/- 0.200	Sensitivity

D.4 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – RANDOM FOREST (RF)

Table D.16 – Accuracy of Random Forest with different number of features

	Model	Scale Type	Training	Validation
0	Random Forest – All Features	Robust Scaler	1.000000	0.877778
1	Random Forest – 3 Features	Robust Scaler	1.000000	0.777778
2	Random Forest – 3 F – no Size	Robust Scaler	1.000000	0.733333
3	Random Forest – 4 Features	Robust Scaler	1.000000	0.777778
4	Random Forest – 5 Features	Robust Scaler	1.000000	0.833333
5	Random Forest – 9 Features	Robust Scaler	1.000000	0.844444

Table D.17 – Accuracy of Random Forest with different *N Estimators* parameter

	Model	Training	Validation
0	RF All Features – Default 2	0.899038	0.688889
1	RF All Features – 5	0.985577	0.855556
2	RF All Features – 10	0.990385	0.877778
3	RF All Features – 100	1.000000	0.877778
4	RF All Features – 300	1.000000	0.877778
5	RF All Features – 570	1.000000	0.877778

Table D.18 – Accuracy of Random Forest with different *Criterion* parameter

	Model	Training	Validation
0	RF All Features – Default Gini	0.990385	0.844444
1	RF All Features – Entropy	0.995192	0.822222

Table D.19 – Accuracy of Random Forest with different *Max Depth* parameter

	Model	Training	Validation
0	RF All Features – Default None	1.000000	0.833333
1	RF All Features – 1	0.730769	0.711111
2	RF All Features – 2	0.798077	0.711111
3	RF All Features – 4	0.903846	0.755556
4	RF All Features – 5	0.942308	0.788889

5	RF All Features – 8	0.985577	0.833333
---	---------------------	----------	----------

Table D.20 – Accuracy of Random Forest with different *Min Samples Leaf* parameter

	Model	Training	Validation
0	RF All Features – Default 1	0.682692	0.611111
1	RF All Features – 2	0.711528	0.655556
2	RF All Features – 4	0.759615	0.744444
3	RF All Features – 5	0.668269	0.600000

Table D.21 – Accuracy of Random Forest with different *Bootstrap* parameter

	Model	Training	Validation
0	RF All Features – Default True	0.706731	0.644444
1	RF All Features – False	0.668269	0.600000

Table D.22 – Accuracy of Random Forest with different *Max Samples* parameter

	Model	Training	Validation
0	RF All Features – Default None	0.725962	0.700000
1	RF All Features – 0.30	0.701923	0.600000
2	RF All Features – 0.60	0.740385	0.700000
3	RF All Features – 0.80	0.735577	0.777778
4	RF All Features – 0.90	0.697115	0.688889

Table D.23 – Accuracy of Random Forest best model with the best parameters

	Model	Scale Type	Training	Validation
0	Random Forest – Best Parameters	Robust Scaler	0.754808	0.744444

Table D.24 – F1-Score, Accuracy, Specificity, Sensitivity of Random Forest of Final Random Forest model with all features

Time	Training	Validation	Model
------	----------	------------	-------

0	0.021 +/- 0.002	0.701 +/- 0.027	0.651 +/- 0.081	F1-Score
1	0.021 +/- 0.002	0.708 +/- 0.024	0.590 +/- 0.096	Accuracy
2	0.021 +/- 0.002	0.649 +/- 0.108	0.400 +/- 0.293	Specificity
3	0.021 +/- 0.002	0.751 +/- 0.157	0.653 +/- 0.175	Sensitivity

D.5 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – NEURAL NETWORK (NN)

Table D.25 – Accuracy of Neural Network with different number of features

	Model	Scale Type	Training	Validation
0	Neural Network – All Features	Robust Scaler	0.903846	0.766667
1	Neural Network – 3 Features	Robust Scaler	0.721154	0.677778
2	Neural Network – 3 F – no Size	Robust Scaler	0.725962	0.677778
3	Neural Network – 4 Features	Robust Scaler	0.639423	0.644444
4	Neural Network – 5 Features	Robust Scaler	0.739769	0.700000
5	Neural Network – 9 Features	Robust Scaler	0.822115	0.755556

Table D.26 – Accuracy of Neural Network best model with the best parameters

	Model	Scale Type	Training	Validation
0	Neural Network – Best Parameters	Robust Scaler	0.807692	0.766667

Table D.27 – F1-Score, Accuracy, Specificity, Sensitivity of Neural Network of Final Neural Network model with 5 features

	Time	Training	Validation	Model
0	0.993 +/- 0.049	0.954 +/- 0.012	0.818 +/- 0.140	F1-Score
1	0.993 +/- 0.049	0.954 +/- 0.012	0.796 +/- 0.131	Accuracy
2	0.993 +/- 0.049	0.927 +/- 0.019	0.523 +/- 0.368	Specificity
3	0.993 +/- 0.049	0.980 +/- 0.015	0.816 +/- 0.196	Sensitivity

D.6 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – SUPPORT VECTOR MACHINE (SVM)

Table D.28 – Accuracy of Support Vector Machine with different number of features

	Model	Scale Type	Training	Validation
0	SVM – All Features	Robust Scaler	0.884615	0.744444
1	SVM – 3 Features	Robust Scaler	0.783654	0.677778
2	SVM – 3 F – no <i>Size</i>	Robust Scaler	0.783654	0.677778
3	SVM – 4 Features	Robust Scaler	0.711538	0.644444
4	SVM – 5 Features	Robust Scaler	0.774038	0.744444
5	SVM – 9 Features	Robust Scaler	0.879808	0.833333

Table D.29 – Accuracy of Support Vector Machine best model with the best parameters

	Model	Scale Type	Training	Validation
0	SVM – Best Parameters	Robust Scaler	0.855769	0.833333

Table D.30 – F1-Score, Accuracy, Specificity, Sensitivity of Support Vector Machine of Final Support Vector Machine model with 9 features

	Time	Training	Validation	Model
0	0.008 +/- 0.000	0.859 +/- 0.018	0.815 +/- 0.145	F1-Score
1	0.008 +/- 0.000	0.859 +/- 0.018	0.793 +/- 0.137	Accuracy
2	0.008 +/- 0.000	0.813 +/- 0.029	0.512 +/- 0.370	Specificity
3	0.008 +/- 0.000	0.036 +/- 0.036	0.826 +/- 0.152	Sensitivity

D.7 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – MULTI-LAYER PERCEPTRON WITH GRID SEARCH (MLP w/ GS)

Table D.31 – Accuracy of Multi-Layer Perceptron with Grid Search with different number of features

	Model	Scale Type	Training	Validation
0	MLP w/ GS – All Features	Robust Scaler	0.913462	0.777778
1	MLP w/ GS – 3 Features	Robust Scaler	0.745192	0.688889
2	MLP w/ GS – 3 F – no Size	Robust Scaler	0.730769	0.700000
3	MLP w/ GS – 4 Features	Robust Scaler	0.658654	0.700000
4	MLP w/ GS – 5 Features	Robust Scaler	0.730769	0.700000
5	MLP w/ GS – 9 Features	Robust Scaler	0.836538	0.766667

Table D.32 – *Parameter Space*

Parameter	Values
<i>hidden_layer_sizes</i>	[(25,25), (50,100), (5,2), (10,10,10), (45,2,11), (25,11,7,5,3,)]
<i>activation</i>	['tanh', 'relu', 'logistic']
<i>solver</i>	['adam', 'lbfgs', 'sgd']
<i>alpha</i>	[0.0001, 0.001]
<i>learning_rate</i>	['invscaling', 'constant']

Table D.33 – Best Parameters found from *Parameter Space* with *Grid Search*

Parameter	Best Value
<i>hidden_layer_sizes</i>	(25,25)
<i>activation</i>	<i>tanh</i>
<i>solver</i>	<i>adam</i>
<i>alpha</i>	0.0001
<i>learning_rate</i>	<i>invscaling</i>

Table D.34 – Accuracy of Multi-Layer Perceptron with Grid Search best model with the best parameters

	Model	Scale Type	Training	Validation
0	MLP w/ GS – Best Parameters	Robust Scaler	0.673077	0.677778

Table D.35 – F1-Score, Accuracy, Specificity, Sensitivity of Multi-Layer Perceptron with Grid Search of Final Multi-Layer Perceptron with Grid Search model with 5 features

	Time	Training	Validation	Model
0	0.222 +/- 0.018	0.712 +/- 0.010	0.723 +/- 0.103	F1-Score
1	0.222 +/- 0.018	0.714 +/- 0.010	0.674 +/- 0.079	Accuracy
2	0.222 +/- 0.018	0.655 +/- 0.047	0.439 +/- 0.311	Specificity
3	0.222 +/- 0.018	0.767 +/- 0.054	0.749 +/- 0.185	Sensitivity

D.8 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – K-NEAREST NEIGHBOR (KNN)

Table D.36 – Accuracy of K-Nearest Neighbor with different number of features

	Model	Scale Type	Training	Validation
0	K-Nearest Neighbor – All Features	Robust Scaler	0.774038	0.744444
1	K-Nearest Neighbor – 3 Features	Robust Scaler	0.793269	0.677778
2	K-Nearest Neighbor – 3 F – no <i>Size</i>	Robust Scaler	0.793269	0.677778
3	K-Nearest Neighbor – 4 Features	Robust Scaler	0.764423	0.666667
4	K-Nearest Neighbor – 5 Features	Robust Scaler	0.774038	0.666667
5	K-Nearest Neighbor – 9 Features	Robust Scaler	0.812500	0.733333

Table D.37 – Accuracy of K-Nearest Neighbor best model with the best parameters

	Model	Scale Type	Training	Validation
0	KNN – Best Parameters	Robust Scaler	0.764423	0.744444

Table D.38 – F1-Score, Accuracy, Specificity, Sensitivity of K-Nearest Neighbor of Final K-Nearest Neighbor model with all features

	Time	Training	Validation	Model
0	0.020 +/- 0.000	0.757 +/- 0.019	0.664 +/- 0.244	F1-Score
1	0.020 +/- 0.000	0.767 +/- 0.017	0.646 +/- 0.228	Accuracy
2	0.020 +/- 0.000	0.578 +/- 0.052	0.274 +/- 0.225	Specificity
3	0.020 +/- 0.000	0.949 +/- 0.020	0.894 +/- 0.143	Sensitivity

D.9 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE BAGGING CLASSIFIER (BC)

Table D.39 – Accuracy of Ensemble Bagging Classifier with different number of features

	Model	Scale Type	Training	Validation
0	Bagging Classifier – All Features	Robust Scaler	0.975962	0.855556
1	Bagging Classifier – 3 Features	Robust Scaler	1.000000	0.744444
2	Bagging Classifier – 3 F – no Size	Robust Scaler	0.995192	0.677778
3	Bagging Classifier – 4 Features	Robust Scaler	0.975962	0.800000
4	Bagging Classifier – 5 Features	Robust Scaler	0.990385	0.755556
5	Bagging Classifier – 9 Features	Robust Scaler	0.985577	0.788889

Table D.40 – Accuracy of Ensemble Bagging Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Bagging Classifier – Best Parameters	Robust Scaler	0.745192	0.744444

Table D.41 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Bagging Classifier of Final Ensemble Bagging Classifier model with all features

	Time	Training	Validation	Model
0	0.162 +/- 0.005	0.735 +/- 0.024	0.756 +/- 0.221	F1-Score
1	0.162 +/- 0.005	0.745 +/- 0.025	0.747 +/- 0.224	Accuracy
2	0.162 +/- 0.005	0.550 +/- 0.027	0.385 +/- 0.308	Specificity
3	0.162 +/- 0.005	0.939 +/- 0.014	0.898 +/- 0.145	Sensitivity

D.10 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE ADABOOST CLASSIFIER (AC)

Table D.42 – Accuracy of Ensemble Adaboost Classifier with different number of features

	Model	Scale Type	Training	Validation
0	Adaboost Classifier – All Features	Robust Scaler	0.995192	0.822222
1	Adaboost Classifier – 3 Features	Robust Scaler	0.817308	0.722222
2	Adaboost Classifier – 3 F – no Size	Robust Scaler	0.817308	0.722222
3	Adaboost Classifier – 4 Features	Robust Scaler	0.908654	0.577778
4	Adaboost Classifier – 5 Features	Robust Scaler	0.918269	0.744444
5	Adaboost Classifier – 9 Features	Robust Scaler	0.995192	0.711111

Table D.43 – Accuracy of Ensemble Adaboost Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Adaboost Classifier – Best Parameters	Robust Scaler	0.706731	0.677778

Table D.44 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Adaboost Classifier of Final Ensemble Adaboost Classifier model

	Time	Training	Validation	Model
0	0.022 +/- 0.005	0.371 +/- 0.020	0.093 +/- 0.098	F1-Score
1	0.022 +/- 0.005	0.534 +/- 0.018	0.197 +/- 0.162	Accuracy
2	0.022 +/- 0.005	0.400 +/- 0.516	0.100 +/- 0.316	Specificity
3	0.022 +/- 0.005	0.600 +/- 0.516	0.600 +/- 0.516	Sensitivity

D.11 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE VOTING CLASSIFIER (VC)

```
#To define the estimators
estimators = [('dt', DT_model_final), ('svc', SVC_model_final)]

estimators_1 = [('nv', modelNB), ('lr', LR_model_newton), ('rf', RF_model_final), ('dt', DT_model_final), ('nn', NN_model)]
estimators_2 = [('gb', gbc_model_final), ('bc', BC_model_final), ('svc', SVC_model_final)]
estimators_3 = [('gb', gbc_model_final), ('dt', DT_model_final), ('svc', SVC_model_final)]
estimators_4 = [('gb', gbc_model_final), ('ac', AdC_model_final), ('bc', BC_model_final)]
estimators_5 = [('gb', gbc_model_final), ('bc', BC_model_final)]
estimators_6 = [('ac', AdC_model_final), ('bc', BC_model_final)]
estimators_7 = [('lr', LR_model_newton), ('bc', BC_model_final)]
estimators_8 = [('ac', AdC_model_final), ('lr', LR_model_newton)]
estimators_9 = [('svc', SVC_model_final), ('bc', BC_model_final)]
estimators_10 = [('ac', AdC_model_final), ('svc', SVC_model_final)]
estimators_11 = [('ac', AdC_model_final), ('svc', SVC_model_final), ('bc', BC_model_final)]
estimators_12 = [('ac', AdC_model_final), ('svc', SVC_model_final), ('bc', BC_model_final), ('lr', LR_model_newton)]
```

Figure D.1 – Estimators used in the Voting Classifier model

Table D.45 – Accuracy of Ensemble Voting Classifier with different number of features

	Model	Scale Type	Training	Validation
0	Voting Classifier – All Features	Robust Scaler	0.841346	0.822222
1	Voting Classifier – 3 Features	Robust Scaler	0.778846	0.722222
2	Voting Classifier – 3 F – no Size	Robust Scaler	0.778846	0.722222
3	Voting Classifier – 4 Features	Robust Scaler	0.716346	0.611111
4	Voting Classifier – 5 Features	Robust Scaler	0.788462	0.733333
5	Voting Classifier – 9 Features	Robust Scaler	0.855769	0.855556

Table D.46 – Accuracy of Ensemble Voting Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Voting Classifier – Best Parameters	Robust Scaler	0.855769	0.955556

Table D.47 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Voting Classifier of Final Ensemble Voting Classifier model with 9 features

Time	Training	Validation	Model
------	----------	------------	-------

0	0.152 +/- 0.009	0.873 +/- 0.016	0.828 +/- 0.140	F1-Score
1	0.152 +/- 0.009	0.874 +/- 0.016	0.806 +/- 0.131	Accuracy
2	0.152 +/- 0.009	0.842 +/- 0.023	0.534 +/- 0.384	Specificity
3	0.152 +/- 0.009	0.902 +/- 0.036	0.822 +/- 0.151	Sensitivity

D.12 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE GRADIENT BOOSTING CLASSIFIER (GBC)

Table D.48 – Accuracy of Ensemble Gradient Boosting Classifier with different number of features

	Model	Scale Type	Training	Validation
0	Gradient Boosting Classifier – All Features	Robust Scaler	1.000000	0.822222
1	Gradient Boosting Classifier – 3 Features	Robust Scaler	0.985577	0.733333
2	Gradient Boosting Classifier – 3 F – no Size	Robust Scaler	0.985577	0.733333
3	Gradient Boosting Classifier – 4 Features	Robust Scaler	0.990385	0.633333
4	Gradient Boosting Classifier – 5 Features	Robust Scaler	0.990385	0.777778
5	Gradient Boosting Classifier – 9 Features	Robust Scaler	0.995192	0.788889

Table D.49 – Accuracy of Ensemble Gradient Boosting Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Gradient Boosting Classifier – Best Parameters	Robust Scaler	0.740385	0.722222

Table D.50 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Gradient Boosting Classifier of Final Ensemble Gradient Boosting Classifier model with all features

	Time	Training	Validation	Model
0	0.088 +/- 0.031	0.798 +/- 0.011	0.714 +/- 0.117	F1-Score
1	0.088 +/- 0.031	0.799 +/- 0.011	0.661 +/- 0.119	Accuracy
2	0.088 +/- 0.031	0.755 +/- 0.042	0.450 +/- 0.317	Specificity
3	0.088 +/- 0.031	0.836 +/- 0.054	0.695 +/- 0.195	Sensitivity

D.13 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE STACKING CLASSIFIER (SC)

```
#To define the estimators
estimators = [('dt', DT_model_final), ('svc', SVC_model_final)]

estimators_1 = [('nv', modelNB), ('lr', LR_model_newton), ('rf', RF_model_final), ('dt', DT_model_final), ('nn', NN_model)]
estimators_2 = [('gb', gbc_model_final), ('bc', BC_model), ('svc', SVC_model_final)]
estimators_3 = [('gb', gbc_model_final), ('dt', DT_model_final), ('svc', SVC_model_final)]
estimators_4 = [('gb', gbc_model_final), ('ac', AdC_model_final), ('bc', BC_model)]
estimators_5 = [('gb', gbc_model_final), ('bc', BC_model)]
estimators_6 = [('ac', AdC_model_final), ('bc', BC_model)]
estimators_7 = [('lr', LR_model_newton), ('gc', BC_model)]
estimators_8 = [('ac', AdC_model_final), ('lr', LR_model_newton)]
estimators_9 = [('svc', SVC_model_final), ('gc', BC_model)]
estimators_10 = [('ac', AdC_model_final), ('svc', SVC_model_final)]
estimators_11 = [('ac', AdC_model_final), ('svc', SVC_model_final), ('gc', BC_model)]
estimators_12 = [('ac', AdC_model_final), ('svc', SVC_model_final), ('gc', BC_model), ('lr', LR_model_newton)]
estimators_13 = [('gc', BC_model), ('dt', DT_model_final)]
estimators_14 = [('gc', BC_model), ('ac', AdC_model_final)]
```

Figure D.2 – Estimators used in the Stacking Classifier model

Table D.51 – Accuracy of Ensemble Stacking Classifier best model with the best parameters

	Model	Scale Type	Training	Validation
0	Stacking Classifier – Best Parameters	Robust Scaler	0.846154	0.844444

Table D.52 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Stacking Classifier of Final Ensemble Stacking Classifier model with 9 features

	Time	Training	Validation	Model
0	1.908 +/- 0.362	0.877 +/- 0.018	0.807 +/- 0.144	F1-Score
1	1.908 +/- 0.362	0.877 +/- 0.018	0.783 +/- 0.135	Accuracy
2	1.908 +/- 0.362	0.859 +/- 0.027	0.513 +/- 0.365	Specificity
3	1.908 +/- 0.362	0.892 +/- 0.034	0.790 +/- 0.166	Sensitivity

D.14 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – ENSEMBLE EXTREME GRADIENT BOOSTING CLASSIFIER (XGBC w/ GS)

Table D.53 – Accuracy of Ensemble Extreme Gradient Boosting Classifier with Grid Search with different number of features

	Model	Scale Type	Training	Validation
0	XGBC w/ GS – All Features	Robust Scaler	1.000000	0.844444
1	XGBC w/ GS – 3 Features	Robust Scaler	0.995192	0.733333
2	XGBC w/ GS – 3 F – no Size	Robust Scaler	0.995192	0.733333
3	XGBC w/ GS – 4 Features	Robust Scaler	1.000000	0.800000
4	XGBC w/ GS – 5 Features	Robust Scaler	1.000000	0.800000
5	XGBC w/ GS – 9 Features	Robust Scaler	1.000000	0.811111

Table D.54 – *Parameter Space*

Parameter	Values
<i>max_depth</i>	[6, 0, 3, 10, None, 17]
<i>learning_rate</i>	[0.3, 0.01, 0.1, 0.5, 0.75, 1.0]
<i>n_estimators</i>	[100, 3, 5, 50, 75, 200, 750, 1000]
<i>subsample</i>	[1, 0.1, 0.25, 0.5, 0.8]
<i>colsample_bytree</i>	[1, 0.2, 0.5, 0.8]

Table D.55 – Best Parameters found from *Parameter Space* with *Grid Search*

Parameter	Best Value
<i>max_depth</i>	6
<i>learning_rate</i>	0.3
<i>n_estimators</i>	100
<i>subsample</i>	1
<i>colsample_bytree</i>	1

Table D.56 – Accuracy of Ensemble Extreme Gradient Boosting Classifier with Grid Search best model with the best parameters

	Model	Scale Type	Training	Validation
0	XGBC w/ GS – Best Parameters	Robust Scaler	1.000000	0.844444

Table D.57 – F1-Score, Accuracy, Specificity, Sensitivity of Ensemble Extreme Gradient Boosting Classifier with Grid Search of Final Ensemble Extreme Gradient Boosting Classifier with Grid Search model with all features

	Time	Training	Validation	Model
0	0.081 +/- 0.044	1.000 +/- 0.000	0.837 +/- 0.137	F1-Score
1	0.081 +/- 0.044	1.000 +/- 0.000	0.819 +/- 0.132	Accuracy
2	0.081 +/- 0.044	1.000 +/- 0.000	0.551 +/- 0.395	Specificity
3	0.081 +/- 0.044	1.000 +/- 0.000	0.913 +/- 0.192	Sensitivity

D.15 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – TREE-BASED PIPELINE OPTIMIZATION TOOL(TPOT)

Table D.58 – Accuracy of TPOT with different number of features (default parameters)

	Model	Scale Type	Training	Validation
0	TPOT – All Features	Robust Scaler	1.000000	0.911111
1	TPOT – 3 Features	Robust Scaler	0.836538	0.677778
2	TPOT – 3 F – no <i>Size</i>	Robust Scaler	0.812500	0.655556
3	TPOT – 4 Features	Robust Scaler	0.629808	0.688889
4	TPOT – 5 Features	Robust Scaler	0.831731	0.688889
5	TPOT – 9 Features	Robust Scaler	0.966346	0.833333

Table D.59 – Accuracy of TPOT with different *Population Size* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 100	1.000000	0.922222
1	TPOT 9Features – 25	0.961538	0.800000
2	TPOT 9Features – 10	1.000000	0.844444
3	TPOT 9Features – 50	1.000000	0.911111
4	TPOT 9Features – 75	1.000000	0.900000
5	TPOT 9Features – 95	1.000000	0.900000

Table D.60 – Accuracy of TPOT with different *Mutation Rate* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 0.9	0.831731	0.722222
1	TPOT 9Features – 0.1	1.000000	0.855556
2	TPOT 9Features – 0.3	1.000000	0.855556
3	TPOT 9Features – 0.5	1.000000	0.944444
4	TPOT 9Features – 0.7	0.995192	0.877778

Table D.61 – Accuracy of TPOT with different *Crossover Rate* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 0.1	0.980769	0.833333

1	TPOT 9Features – 0.3	0.947115	0.800000
2	TPOT 9Features – 0.4	1.000000	0.844444
3	TPOT 9Features – 0.5	1.000000	0.944444
4	TPOT 9Features – 0.7	0.966346	0.888889

Table D.62 – Accuracy of TPOT with different cv parameter

	Model	Training	Validation
0	TPOT 9Features – Default 5	1.000000	0.911111
1	TPOT 9Features – 3	1.000000	0.844444
2	TPOT 9Features – 7	1.000000	0.922222
3	TPOT 9Features – 10	1.000000	0.888889
4	TPOT 9Features – 25	1.000000	0.911111
5	TPOT 9Features – 50	1.000000	0.944444
6	TPOT 9Features – 75	1.000000	0.900000

Table D.63 – Accuracy of TPOT with different *Random State* parameter

	Model	Training	Validation
0	TPOT 9Features – Default 10	1.000000	0.900000
1	TPOT 9Features – 15	1.000000	0.933333
2	TPOT 9Features – 42	1.000000	0.911111
3	TPOT 9Features – 123	1.000000	0.833333

```
HBox(children=(FloatProgress(value=0.0, description='Optimization Progress', max=600.0, style=ProgressStyle(de...

Generation 1 - Current best internal CV score: 0.8419999999999999
Generation 2 - Current best internal CV score: 0.862
Generation 3 - Current best internal CV score: 0.8740000000000001
Generation 4 - Current best internal CV score: 0.889
Generation 5 - Current best internal CV score: 0.889

Best pipeline: ExtraTreesClassifier(FastICA(input_matrix, tol=0.35000000000000003), bootstrap=False, criterion=entrop
y, max_features=0.9500000000000001, min_samples_leaf=5, min_samples_split=10, n_estimators=100)
```

Figure D.3 – Accuracy of TPOT best model with the best parameters

Table D.64 – F1-Score, Accuracy, Specificity, Sensitivity of TPOT of Final TPOT Classifier model with all features

	Time	Training	Validation	Model
0	6664.3333 +/- 14222.348	0.998 +/- 0.004	0.907 +/- 0.073	F1-Score
1	6664.3333 +/- 14222.348	0.998 +/- 0.004	0.900 +/- 0.074	Accuracy
2	6664.3333 +/- 14222.348	0.998 +/- 0.003	0.624 +/- 0.435	Specificity
3	6664.3333 +/- 14222.348	0.997 +/- 0.008	0.867 +/- 0.176	Sensitivity

D.16 DIAGNOSTIC RESULTS – CLASSIFICATION PROBLEM – FINAL RESULTS

Table D.65 – Accuracy of all models

	Model	Training	Validation
0	Gaussian Naïve Bayes	0.711538	0.700000
1	Logistic Regression	0.668269	0.666667
2	Decision Tree	0.682692	0.700000
3	Random Forest	0.754808	0.744444
4	Neural Network	0.807692	0.766667
5	Support Vector Machine	0.855769	0.833333
6	Multi-Layer Perceptron with Grid Search	0.673077	0.677778
7	K-Nearest Neighbors	0.764423	0.744444
8	Bagging Classifier	0.745192	0.744444
9	Adaboost Classifier	0.706731	0.677778
10	Voting Classifier	0.855769	0.855556
11	Gradient Boosting Classifier	0.740385	0.722222
12	Stacking Classifier	0.846154	0.844444
13	Extreme Gradient Boosting Classifier	1.000000	0.844444
14	TPOT	1.000000	0.981395

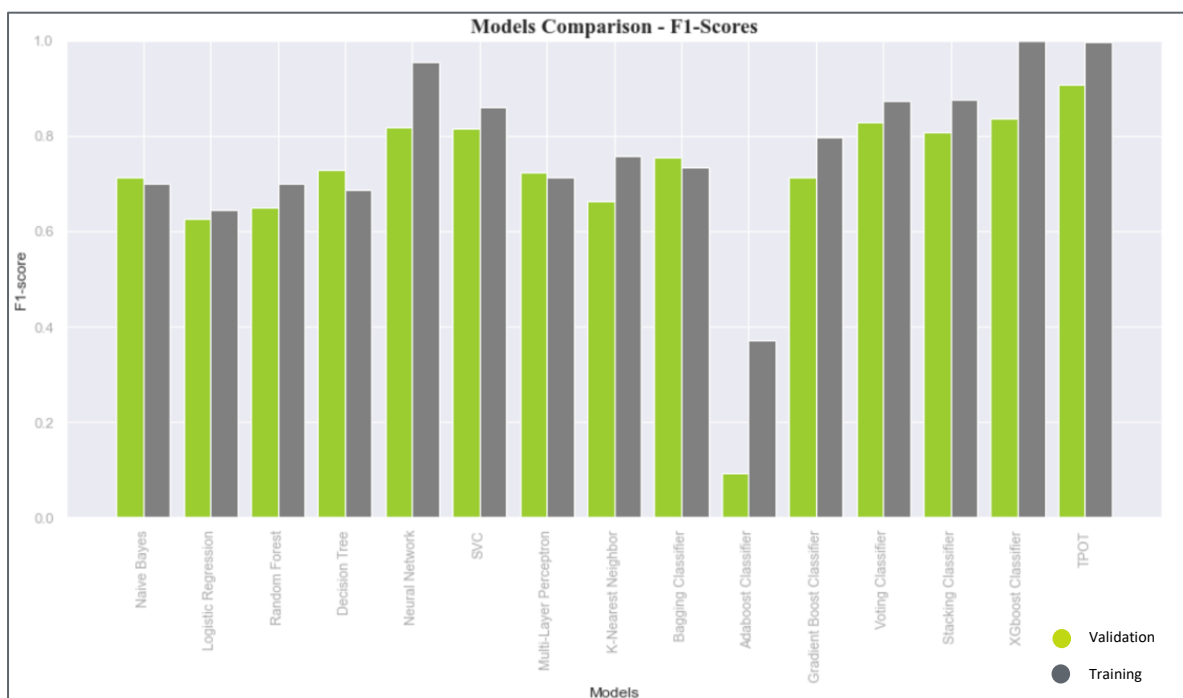


Figure D.4 – F-1 Scores of all model

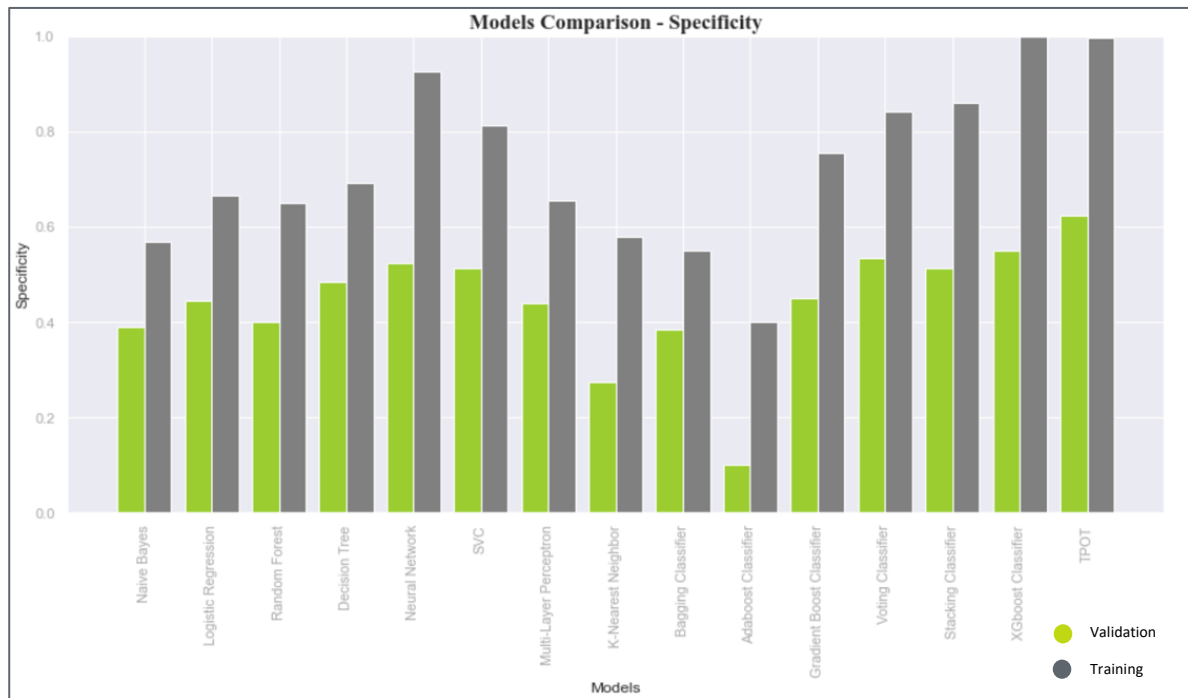


Figure D.5 – Specificity of all models

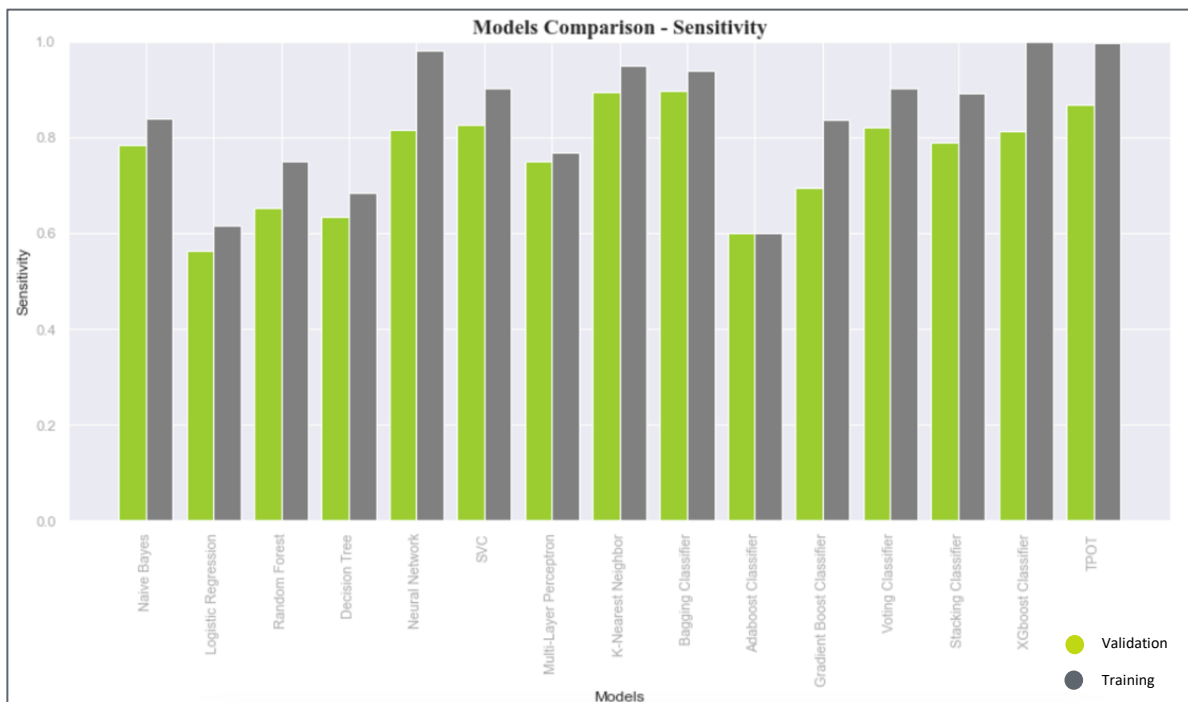


Figure D.6 – Sensitivity of all models

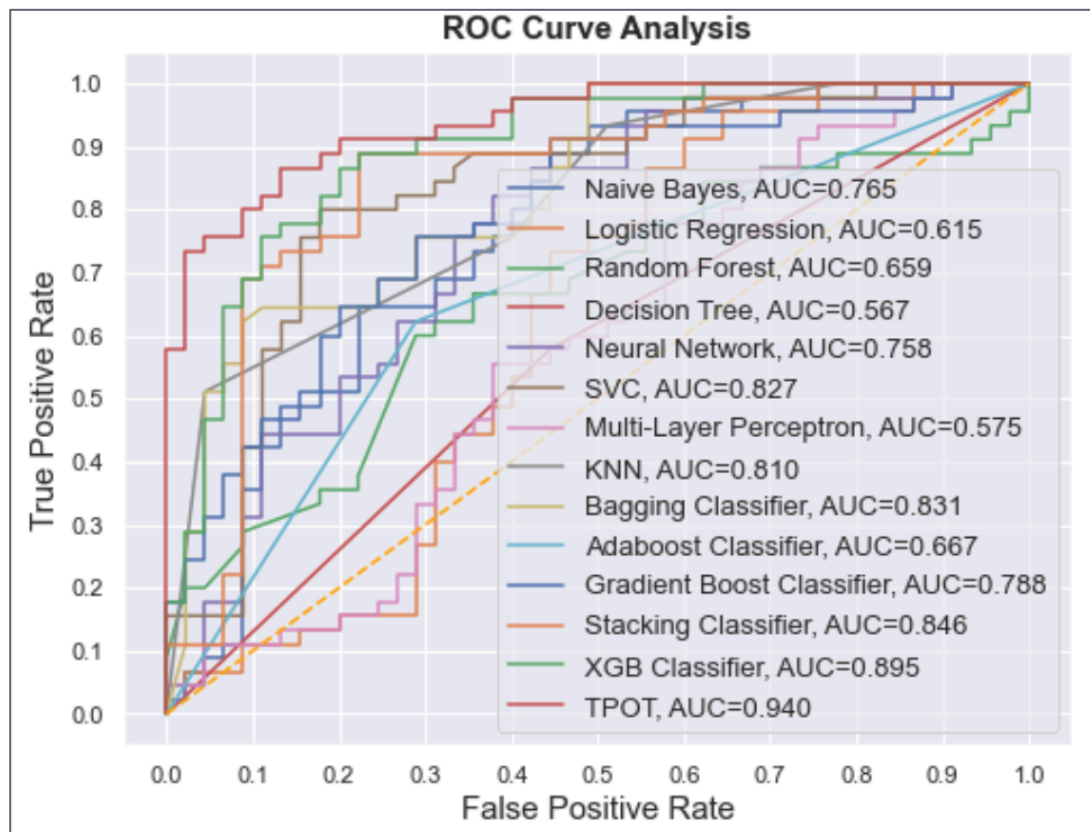


Figure D.7 – ROC Curve

E.1 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – LINEAR REGRESSION (LiR)

Table E.1 – MSE and R² of Linear Regression with different numbers of features

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	LiR – All Features	0.046467	0.262882	0.862416	0.167869
1	LiR – 3 Features	0.270590	0.309802	0.198813	0.019349
2	LiR – 5 Features	0.267768	0.304435	0.207168	0.036338
3	LiR – 9 Features	0.261630	0.308472	0.225343	0.023560

Table E.2 – MAE, MSE, RMSE, R² and MAPE of Linear Regression of Final model with 5 features

	Time	Training	Validation	Model
0	0.005 +/- 0.000	0.433 +/- 0.012	0.455 +/- 0.111	MAE
1	0.005 +/- 0.000	0.272 +/- 0.015	0.302 +/- 0.140	MSE
2	0.005 +/- 0.000	0.521 +/- 0.015	0.537 +/- 0.123	RMSE
3	0.005 +/- 0.000	0.185 +/- 0.022	-5.923 +/- 17.253	R ²
4	0.005 +/- 0.000	161.815 +/- 11.753	174.673 +/- 95.184	MAPE

E.2 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – GAUSSIAN PROCESS REGRESSION (GPR)

Table E.3 – MSE and R^2 of Gaussian Process Regression with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	GPR – All Features	0.000000	0.332633	1.000000	-0.052919
1	GPR – 3 Features	0.000000	0.317580	1.000000	-0.005271
2	GPR – 5 Features	0.000000	0.317580	1.000000	-0.005271
3	GPR – 9 Features	0.000000	0.366165	1.000000	-0.159063

Table E.4 – MAE, MSE, RMSE, R^2 and MAPE of Gaussian Process Regression of Final model with 5 features

	Time	Training	Validation	Model
0	0.012 +/- 0.003	0.000 +/- 0.000	0.489 +/- 0.168	MAE
1	0.012 +/- 0.003	0.000 +/- 0.000	0.346 +/- 0.246	MSE
2	0.012 +/- 0.003	0.000 +/- 0.000	0.557 +/- 0.198	RMSE
3	0.012 +/- 0.003	1.000 +/- 0.000	-4.975 +/- 14.208	R^2
4	0.012 +/- 0.003	349.890 +/- 27.641	100.673 +/- 0.000	MAPE

E.3 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – DECISION TREE (DT)

Table E.5 – MSE and R^2 of Decision Tree with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	DT – All Features	0.000000	0.383327	1.000000	-0.213387
1	DT – 3 Features	0.000000	0.393296	1.000000	-0.244945
2	DT – 5 Features	0.000000	0.323451	1.000000	-0.023854
3	DT – 9 Features	0.000000	0.359287	1.000000	-0.137292

Table E.6 – MSE and R^2 of Decision Tree best model with the best parameters

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	Decision Tree	0.176601	0.201804	0.477105	0.361208

Table E.7 – MAE, MSE, RMSE, R^2 and MAPE of Decision Tree of Final model with 5 features

	Time	Training	Validation	Model
0	0.005 +/- 0.001	0.320 +/- 0.013	0.497 +/- 0.108	MAE
1	0.005 +/- 0.001	0.158 +/- 0.010	0.379 +/- 0.149	MSE
2	0.005 +/- 0.001	0.397 +/- 0.012	0.604 +/- 0.125	RMSE
3	0.005 +/- 0.001	0.526 +/- 0.042	-7.857 +/- 21.675	R^2
4	0.005 +/- 0.001	262.037 +/- 22.135	229.194 +/- 0.603	MAPE

E.4 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – RANDOM FOREST (RF)

Table E.8 – MSE and R^2 of Random Forest with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	RF – All Features	0.047323	0.296149	0.859882	0.062565
1	RF – 3 Features	0.052025	0.250842	0.845960	0.205984
2	RF – 5 Features	0.053317	0.245570	0.842135	0.222670
3	RF – 9 Features	0.044858	0.271387	0.867180	0.140949

Table E.9 – MSE and R^2 of Random Forest best model with the best parameters

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	Random Forest	0.190800	0.240689	0.435061	0.238122

Table E.10 – MAE, MSE, RMSE, R^2 and MAPE of Random Forest of Final model with 5 features

	Time	Training	Validation	Model
0	0.023 +/- 0.001	0.363 +/- 0.010	0.466 +/- 0.109	MAE
1	0.023 +/- 0.001	0.190 +/- 0.009	0.307 +/- 0.127	MSE
2	0.023 +/- 0.001	0.435 +/- 0.010	0.543 +/- 0.117	RMSE
3	0.023 +/- 0.001	0.431 +/- 0.024	-6.338 +/- 18.368	R^2
4	0.023 +/- 0.001	214.177 +/- 21.001	186.322 +/- 77.773	MAPE

E.5 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ELASTIC NET (EN)

Table E.11 – MSE and R^2 of Elastic Net with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	EN – All Features	0.045368	0.296149	0.865671	0.140367
1	EN – 3 Features	0.337736	0.326412	0.000000	-0.033227
2	EN – 5 Features	0.337736	0.326412	0.000000	-0.033227
3	EN – 9 Features	0.337736	0.326412	0.000000	-0.033227

Table E.12 – MSE and R^2 of Elastic Net best model with the best parameters

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	Elastic Net	0.262004	0.289224	0.224234	0.084488

Table E.13 – MAE, MSE, RMSE, R^2 and MAPE of Elastic Net of Final model with all features

	Time	Training	Validation	Model
0	0.005 +/- 0.001	0.432 +/- 0.013	0.475 +/- 0.125	MAE
1	0.005 +/- 0.001	0.265 +/- 0.015	0.319 +/- 0.155	MSE
2	0.005 +/- 0.001	0.514 +/- 0.015	0.550 +/- 0.135	RMSE
3	0.005 +/- 0.001	0.205 +/- 0.026	-6.938 +/- 20.333	R^2
4	0.005 +/- 0.001	183.115 +/- 15.681	166.842 +/- 65.070	MAPE

E.6 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – RIDGE REGRESSION (RR)

Table E.14 – MSE and R² of Ridge Regression with different numbers of features

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	RR – All Features	0.216862	0.317939	0.357897	-0.006409
1	RR – 3 Features	0.270596	0.309080	0.198793	0.021636
2	RR – 5 Features	0.267777	0.303487	0.207142	0.039340
3	RR – 9 Features	0.261645	0.307304	0.225297	0.027255

Table E.15 – MSE and R² of Ridge Regression best model with the best parameters

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	Ridge Regression	0.268428	0.296813	0.205213	0.060464

Table E.16 – MAE, MSE, RMSE, R² and MAPE of Ridge Regression of Final model with 5 features

	Time	Training	Validation	Model
0	0.004 +/- 0.001	0.434 +/- 0.013	0.455 +/- 0.112	MAE
1	0.004 +/- 0.001	0.272 +/- 0.015	0.300 +/- 0.144	MSE
2	0.004 +/- 0.001	0.521 +/- 0.015	0.535 +/- 0.126	RMSE
3	0.004 +/- 0.001	0.184 +/- 0.022	-5.822 +/- 16.971	R ²
4	0.004 +/- 0.001	156.587 +/- 10.944	168.419 +/- 88.928	MAPE

E.7 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – BAYESIAN RIDGE REGRESSION (BRR)

Table E.17 – MSE and R² of Bayesian Ridge Regression with different numbers of features

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	BRR – All Features	0.259369	0.282834	0.232037	0.104714
1	BRR – 3 Features	0.271052	0.304326	0.197444	0.036682
2	BRR – 5 Features	0.269261	0.303487	0.202748	0.070716
3	BRR – 9 Features	0.267309	0.291886	0.208527	0.076061

Table E.18 – MSE and R² of Bayesian Ridge Regression best model with the best parameters

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	Bayesian Ridge Regression	0.253969	0.282834	0.232037	0.104714

Table E.19 – MAE, MSE, RMSE, R² and MAPE of Bayesian Ridge Regression of Final model with all features

	Time	Training	Validation	Model
0	0.007 +/- 0.001	0.428 +/- 0.014	0.469 +/- 0.124	MAE
1	0.007 +/- 0.001	0.262 +/- 0.015	0.312 +/- 0.149	MSE
2	0.007 +/- 0.001	0.511 +/- 0.015	0.544 +/- 0.131	RMSE
3	0.007 +/- 0.001	0.215 +/- 0.030	-7.1232 +/- 21.004	R ²
4	0.007 +/- 0.001	187.165 +/- 17.139	168.911 +/- 66.543	MAPE

E.8 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – NEURAL NETWORK (NN)

Table E.20 – MSE and R^2 of Neural Network with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	NN – All Features	0.039323	0.512381	0.883568	-0.621895
1	NN – 3 Features	0.254840	0.294672	0.245446	0.067243
2	NN – 5 Features	0.236864	0.315033	0.298671	0.002791
3	NN – 9 Features	0.152782	0.385828	0.547630	-0.221302

Table E.21 – MSE and R^2 of Neural Network best model with the best parameters

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	Neural Network	0.232600	0.289241	0.311297	0.084434

Table E.22 – MAE, MSE, RMSE, R^2 and MAPE of Neural Network of Final model with all features

	Time	Training	Validation	Model
0	0.182 +/- 0.016	0.351 +/- 0.023	0.492 +/- 0.157	MAE
1	0.182 +/- 0.016	0.188 +/- 0.021	0.372 +/- 0.218	MSE
2	0.182 +/- 0.016	0.433 +/- 0.024	0.588 +/- 0.172	RMSE
3	0.182 +/- 0.016	0.438 +/- 0.037	-6.961 +/- 19.754	R^2
4	0.182 +/- 0.016	233.853 +/- 20.035	211.832 +/- 89.154	MAPE

E.9 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – MULTI-LAYER PERCEPTRON WITH GRID SEARCH (MLP)

Table E.23 – MSE and R^2 of Multi-Layer Perceptron with Grid Search with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	MLP – All Features	0.044098	0.446176	0.869431	-0.412331
1	MLP – 3 Features	0.254261	0.302128	0.247160	0.043639
2	MLP – 5 Features	0.227812	0.319972	0.325473	-0.012842
3	MLP – 9 Features	0.122116	0.381058	0.638428	-0.206205

Table E.24 – *Parameter Space*

Parameter	Values
<i>hidden_layer_sizes</i>	[(25,25), (50,100), (5,2), (10,10,10), (45,2,11), (25,11,7,5,3,),(15,)]
<i>activation</i>	['tanh', 'relu', 'logistic']
<i>solver</i>	['adam', 'lbfgs', 'sgd']
<i>alpha</i>	[0.0001, 0.001]
<i>learning_rate</i>	['invscaling', 'constant']

Table E.25 – Best Parameters found from *Parameter Space* with *Grid Search*

Parameter	Best Value
<i>hidden_layer_sizes</i>	(45, 2, 11)
<i>activation</i>	<i>tanh</i>
<i>solver</i>	<i>adam</i>
<i>alpha</i>	0.001
<i>learning_rate</i>	<i>constant</i>

Table E.26 – MSE and R² of Multi-Layer Perceptron with Grid Search best model with the best parameters

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	Multi-Layer Perceptron	0.270063	0.314311	0.200372	0.005078

Table E.27 – MAE, MSE, RMSE, R² and MAPE of Multi-Layer Perceptron with Grid Search of Final model with 3 features

	Time	Training	Validation	Model
0	0.063 +/- 0.023	0.436 +/- 0.014	0.456 +/- 0.113	MAE
1	0.063 +/- 0.023	0.275 +/- 0.017	0.295 +/- 0.139	MSE
2	0.063 +/- 0.023	0.525 +/- 0.017	0.530 +/- 0.127	RMSE
3	0.063 +/- 0.023	0.174 +/- 0.030	-5.388 +/- 15.733	R ²
4	0.063 +/- 0.023	199.351 +/- 18.050	190.756 +/- 107.555	MAPE

E.10 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – SUPPORT VECTOR MACHINE (SVM)

Table E.28 – MSE and R^2 of Support Vector Machine with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	SVM – All Features	0.148875	0.306714	0.559196	0.029125
1	SVM – 3 Features	0.250099	0.278532	0.259483	0.118331
2	SVM – 5 Features	0.210074	0.310510	0.377993	0.017108
3	SVM – 9 Features	0.181710	0.347845	0.451976	-0.101073

Table E.29 – MSE and R^2 of Support Vector Machine best model with the best parameters

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	Support Vector Machine	0.237621	0.215772	0.296429	0.316993

Table E.30 – MAE, MSE, RMSE, R^2 and MAPE of Support Vector Machine of Final model with 3 features

	Time	Training	Validation	Model
0	0.011 +/- 0.008	0.377 +/- 0.021	0.497 +/- 0.133	MAE
1	0.011 +/- 0.008	0.220 +/- 0.027	0.380 +/- 0.182	MSE
2	0.011 +/- 0.008	0.469 +/- 0.028	0.600 +/- 0.149	RMSE
3	0.011 +/- 0.008	0.340 +/- 0.062	-7.999 +/- 22.931	R^2
4	0.011 +/- 0.008	246.671 +/- 29.284	214.484 +/- 92.899	MAPE

E.11 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – K-NEAREST NEIGHBOR (KNN)

Table E.31 – MSE and R^2 of K-Nearest Neighbor with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	KNN – All Features	0.261075	0.279295	0.226986	0.115918
1	KNN – 3 Features	0.244102	0.252492	0.277239	0.200758
2	KNN – 5 Features	0.267707	0.248196	0.207350	0.214357
3	KNN – 9 Features	0.238916	0.331113	0.292595	-0.048109

Table E.32 – MSE and R^2 of K-Nearest Neighbor best model with the best parameters

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	K-Nearest Neighbor	0.267707	0.248196	0.207350	0.214357

Table E.33 – MAE, MSE, RMSE, R^2 and MAPE of K-Nearest Neighbor of Final model with 3 features

	Time	Training	Validation	Model
0	0.007 +/- 0.005	0.381 +/- 0.012	0.472 +/- 0.132	MAE
1	0.007 +/- 0.005	0.222 +/- 0.013	0.331 +/- 0.155	MSE
2	0.007 +/- 0.005	0.471 +/- 0.014	0.560 +/- 0.137	RMSE
3	0.007 +/- 0.005	0.334 +/- 0.031	-6.989 +/- 20.329	R^2
4	0.007 +/- 0.005	139.552 +/- 13.142	174.429 +/- 77.339	MAPE

E.12 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE BAGGING REGRESSOR (BR)

Table E.34 – MSE and R² of Ensemble Bagging Regressor with different numbers of features

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	BR – All Features	0.066559	0.296949	0.802925	0.060034
1	BR – 3 Features	0.074220	0.236333	0.780243	0.251909
2	BR – 5 Features	0.065503	0.258720	0.806051	0.181046
3	BR – 9 Features	0.073856	0.342047	0.781321	-0.082719

Table E.35 – MSE and R² of Ensemble Bagging Regressor best model with the best parameter

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	Ensemble Bagging Regressor	0.220742	0.217538	0.346406	0.311403

Table E.36 – MAE, MSE, RMSE, R² and MAPE of Ensemble Bagging Regressor of Final model with 3 features

	Time	Training	Validation	Model
0	0.087 +/- 0.012	0.388 +/- 0.013	0.386 +/- 0.112	MAE
1	0.087 +/- 0.012	0.220 +/- 0.015	0.218 +/- 0.135	MSE
2	0.087 +/- 0.012	0.469 +/- 0.017	0.450 +/- 0.133	RMSE
3	0.087 +/- 0.012	0.340 +/- 0.024	-2.965 +/- 9.295	R ²
4	0.087 +/- 0.012	218.003 +/- 14.886	188.570 +/- 101.422	MAPE

E.13 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE ADABOOST REGRESSOR (AR)

Table A.37 – MSE and R^2 of Ensemble Adaboost Regressor with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	AR – All Features	0.101627	0.298171	0.699092	0.056168
1	AR – 3 Features	0.185495	0.221057	0.450769	0.300265
2	AR – 5 Features	0.179274	0.267482	0.469190	0.153309
3	AR – 9 Features	0.156020	0.247711	0.538942	0.215893

Table E.38 – MSE and R^2 of Ensemble Adaboost Regressor best model with the best parameter

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	Ensemble Adaboost Regressor	0.191283	0.207720	0.433632	0.342481

Table E.39 – MAE, MSE, RMSE, R^2 and MAPE of Ensemble Adaboost Regressor of Final model with 3 features

	Time	Training	Validation	Model
0	0.366 +/- 0.020	0.378 +/- 0.014	0.489 +/- 0.128	MAE
1	0.366 +/- 0.020	0.187 +/- 0.014	0.360 +/- 0.171	MSE
2	0.366 +/- 0.020	0.433 +/- 0.016	0.585 +/- 0.143	RMSE
3	0.366 +/- 0.020	0.438 +/- 0.020	-8.804 +/- 25.496	R^2
4	0.366 +/- 0.020	240.811 +/- 20.884	221.651 +/- 107.493	MAPE

E.14 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE GRADIENT BOOSTING REGRESSOR (GBR)

Table E.40 – MSE and R² of Ensemble Gradient Boosting Regressor with different numbers of features

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	GBR – All Features	0.011301	0.293124	0.966539	0.072142
1	GBR – 3 Features	0.057654	0.278475	0.829294	0.118512
2	GBR – 5 Features	0.045272	0.269392	0.865955	0.147264
3	GBR – 9 Features	0.028119	0.306008	0.916742	0.031360

Table E.41 – MSE and R² of Ensemble Gradient Boosting Regressor best model with the best parameter

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	Ensemble Gradient Boosting Regressor	0.000001	0.224828	0.999996	0.288327

Table E.42 – MAE, MSE, RMSE, R² and MAPE of Ensemble Gradient Boosting Regressor of Final model with 3 features

	Time	Training	Validation	Model
0	0.057 +/- 0.010	0.002 +/- 0.000	0.565 +/- 0.129	MAE
1	0.057 +/- 0.010	0.000 +/- 0.000	0.480 +/- 0.185	MSE
2	0.057 +/- 0.010	0.003 +/- 0.001	0.680 +/- 0.138	RMSE
3	0.057 +/- 0.010	1.000 +/- 0.000	-10.374 +/- 28.226	R ²
4	0.057 +/- 0.010	349.750 +/- 27.622	308.151 +/- 206.297	MAPE

E.15 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE VOTING REGRESSOR (VR)

```
#To define the estimators
estimators = [('dt', DT_model_final), ('svr', SVR_model_final), ('rr', RR_model_final)]

estimators_1 = [('en', EN_model_final), ('lr', lr), ('rf', RF_model_final), ('dt', DT_model_final), ('nn', NN_model_final)]
estimators_2 = [('gbr', GBR_model_final), ('br', BR_model_final), ('svr', SVR_model_final)]
estimators_3 = [('gbr', GBR_model_final), ('dt', DT_model_final), ('svr', SVR_model_final)]
estimators_4 = [('gbr', GBR_model_final), ('ar', AR_model_final), ('br', BR_model_final)]
estimators_5 = [('gbr', GBR_model_final), ('br', BR_model_final)]
estimators_6 = [('ar', AR_model_final), ('br', BR_model_final)]
estimators_7 = [('lr', lr), ('br', BR_model_final), ('nn', NN_model_final)]
estimators_8 = [('ar', AR_model_final), ('en', EN_model_final)]
estimators_9 = [('svr', SVR_model_final), ('br', BR_model_final)]
estimators_10 = [('ar', AR_model_final), ('svr', SVR_model_final)]
estimators_11 = [('ar', AR_model_final), ('svr', SVR_model_final), ('br', BR_model_final)]
estimators_12 = [('ar', AR_model_final), ('svr', SVR_model_final), ('br', BR_model_final), ('gpr', gpr)]
```

Figure E.1 – Estimators used in the Voting Regressor model

Table E.43 – MSE and R² of Ensemble Voting Regressor with different numbers of features

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	VR – All Features	0.106020	0.288126	0.686086	0.087964
1	VR – 3 Features	0.199749	0.209147	0.408566	0.337963
2	VR – 5 Features	0.151001	0.309066	0.552903	0.021679
3	VR – 9 Features	0.097719	0.410530	0.710665	-0.299496

Table E.44 – MSE and R² of Ensemble Voting Regressor best model with the best parameter

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	Ensemble Voting Regressor	0.199508	0.196563	0.409279	0.377797

Table E.45 – MAE, MSE, RMSE, R^2 and MAPE of Ensemble Voting Regressor of Final model with 3 features

	Time	Training	Validation	Model
0	0.334 +/- 0.080	0.368 +/- 0.011	0.490 +/- 0.134	MAE
1	0.334 +/- 0.080	0.190 +/- 0.013	0.360 +/- 0.179	MSE
2	0.334 +/- 0.080	0.435 +/- 0.014	0.583 +/- 0.148	RMSE
3	0.334 +/- 0.080	0.431 +/- 0.024	-7.993 +/- 23.14	R^2
4	0.334 +/- 0.080	235.831 +/- 21.899	208.560 +/- 96.272	MAPE

E.16 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE STACKING REGRESSOR (SR)

```
#To define the estimators
estimators = [('dt', DT_model_final), ('svr',SVR_model_final)]

estimators_1 = [('gpr', gpr), ('lr', lr), ('rf', RF_model_final), ('dt', DT_model_final), ('nn', NN_model_final), ('svr',SVR_model_final)]
estimators_2 = [('gbr', GBR_model_final), ('br',BR_model), ('svr',SVR_model_final)]
estimators_3 = [('gbr', GBR_model_final), ('en', EN_model_final), ('svr',SVR_model_final)]
estimators_4 = [('gbr', GBR_model_final), ('ar', AR_model_final), ('br',BR_model)]
estimators_5 = [('gbr', GBR_model_final), ('br',BR_model)]
estimators_6 = [('ar', AR_model_final), ('br',BR_model)]
estimators_7 = [('rr', RR_model_final), ('br',BR_model)]
estimators_8 = [('AR', AR_model_final), ('gpr',gpr)]
estimators_9 = [('svr', SVR_model_final), ('BR',BR_model)]
estimators_10 = [('ar', AR_model_final), ('svr',SVR_model_final)]
estimators_11 = [('ar', AR_model_final), ('svr',SVR_model_final), ('br', BR_model)]
estimators_12 = [('ar', AR_model_final), ('svr',SVR_model_final), ('br', BR_model), ('EN', EN_model_final)]
estimators_13 = [('br',BR_model), ('nn', NN_model_final)]
estimators_14 = [('gbr',GBR_model), ('aR', AR_model_final)]
```

Figure E.2 – Estimators used in the Stacking Regressor model

Table E.46 – MSE and R^2 of Ensemble Stacking Regressor best model with the best parameter

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	Ensemble Stacking Regressor	0.256475	0.222765	0.240607	0.294857

Table E.47 – MAE, MSE, RMSE, R^2 and MAPE of Ensemble Stacking Regressor of Final model

	Time	Training	Validation	Model
0	1.924 +/- 0.106	0.448 +/- 0.034	0.553 +/- 0.170	MAE
1	1.924 +/- 0.106	0.303 +/- 0.060	0.456 +/- 0.237	MSE
2	1.924 +/- 0.106	0.548 +/- 0.053	0.650 +/- 0.193	RMSE
3	1.924 +/- 0.106	0.089 +/- 0.184	-8.002 +/- 21.987	R^2
4	1.924 +/- 0.106	215.040 +/- 60.169	210.746 +/- 117.926	MAPE

E.17 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – ENSEMBLE EXTREME GRADIENT BOOSTING REGRESSOR (XGBR)

Table E.48 – MSE and R^2 of Ensemble Extreme Gradient Boosting Regressor with Grid Search with different numbers of features

	Model	MSE Training	MSE Validation	Score R^2 Training	Score R^2 Validation
0	XGBR – All Features	0.000000	0.295432	0.999999	0.064838
1	XGBR – 3 Features	0.000002	0.254483	0.999993	0.194458
2	XGBR – 5 Features	0.000002	0.271313	0.999995	0.141185
3	XGBR – 9 Features	0.000001	0.291207	0.999998	0.078211

Table E.49 – *Parameter Space*

Parameter	Values
<i>max_depth</i>	[6, 0, 3, 10, None, 17]
<i>learning_rate</i>	[0.3, 0.01, 0.1, 0.5, 0.75, 1.0]
<i>n_estimators</i>	[100, 3, 5, 50, 75, 200, 750, 1000]
<i>subsample</i>	[1, 0.1, 0.25, 0.5, 0.8]
<i>colsample_bytree</i>	[1, 0.2, 0.5, 0.8]

Table E.50 – Best Parameters found from *Parameter Space* with *Grid Search*

Parameter	Best Value
<i>max_depth</i>	3
<i>learning_rate</i>	0.01
<i>n_estimators</i>	200
<i>subsample</i>	0.5
<i>colsample_bytree</i>	0.2

Table A.51 – MSE and R^2 of Ensemble Extreme Gradient Boosting Regressor with Grid Search best model with the best parameters

Model	MSE	MSE	Score R^2	Score R^2
-------	-----	-----	-------------	-------------

		Training	Validation	Training	Validation
0	Ensemble Extreme Gradient Boosting Classifier	0.223830	0.292294	0.337263	0.074770

Table A.52 – MAE, MSE, RMSE, R^2 and MAPE of Ensemble Extreme Gradient Boosting Regressor with Grid Search of Final model with 3 features

	Time	Training	Validation	Model
0	0.208 +/- 0.124	0.409 +/- 0.010	0.489 +/- 0.112	MAE
1	0.208 +/- 0.124	0.224 +/- 0.012	0.322 +/- 0.138	MSE
2	0.208 +/- 0.124	0.473 +/- 0.013	0.557 +/- 0.115	RMSE
3	0.208 +/- 0.124	0.327 +/- 0.015	-8.463 +/- 24.686	R^2
4	0.208 +/- 0.124	189.275 +/- 17.007	183.636 +/- 99.971	MAPE

E.18 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – TREE-BASED PIPELINE OPTIMIZATION TOOL (TPOT)

Table E.53 – MSE and R² of TPOT with different numbers of features

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	TPOT – All Features	0.251471	0.291909	-0.251471	-0.291909
1	TPOT – 3 Features	0.271757	0.315779	-0.271757	-0.315779
2	TPOT – 5 Features	0.275561	0.318678	-0.275561	-0.318678
3	TPOT – 9 Features	0.255475	0.310234	-0.255475	-0.310234

```

Generation 1 - Current best internal CV score: -0.2965777428461482
Generation 2 - Current best internal CV score: -0.2965777428461482
Generation 3 - Current best internal CV score: -0.2958229574338657
Generation 4 - Current best internal CV score: -0.2958229574338657
Generation 5 - Current best internal CV score: -0.2958229574338657

Best pipeline: RidgeCV(MinMaxScaler(VarianceThreshold(SGDRegressor(input_matrix, alpha=0.0, eta0=1.0, fit_intercept=False, l1_ratio=0.0, learning_rate=invscaling, loss=epsilon_insensitive, penalty=elasticnet, power_t=0.1), threshold=0.0005)))

```

Figure E.3 – Accuracy of TPOT best model with the best parameters

Table E.54 – MAE, MSE, RMSE, R² and MAPE of TPOT of Final model with 3 features

	Time	Training	Validation	Model
0	160.053 +/- 40.293	0.420 +/- 0.024	0.60 +/- 0.151	MAE
1	160.053 +/- 40.293	0.257 +/- 0.032	0.304 +/- 0.196	MSE
2	160.053 +/- 40.293	0.506 +/- 0.032	0.528 +/- 0.170	RMSE
3	160.053 +/- 40.293	0.229 +/- 0.091	-4.466 +/- 13.076	R ²
4	160.053 +/- 40.293	154.471 +/- 14.800	139.222 +/- 51.017	MAPE

E.19 DIAGNOSTIC RESULTS – REGRESSION PROBLEM – FINAL RESULTS

Table E.55 – MSE and R² of all models

	Model	MSE Training	MSE Validation	Score R ² Training	Score R ² Validation
0	Linear Regression	2.677680e-01	0.304435	0.207168	0.036338
1	Gaussian Process Regressor	3.582604e-21	0.317580	1.000000	-0.005271
2	Decision Tree	1.766006e-01	0.201804	0.477105	0.361208
3	Random Forest	1.908004e-01	0.240689	0.435061	0.238122
4	Elastic Net	2.620041e-01	0.289224	0.224234	0.084488
5	Ridge Regression	2.684281e-01	0.296813	0.205213	0.060464
6	Bayesian Ridge Regressor	2.593687e-01	0.282834	0.232037	0.104714
7	Neural Network	2.326000e-01	0.289241	0.311297	0.084434
8	Multi-Layer Perceptron with Grid Search	2.700063e-01	0.314311	0.200372	0.005078
9	Support Vector Machine	2.376212e-01	0.215772	0.296429	0.316993
10	K-Nearest Neighbors	2.677066e-01	0.248196	0.207350	0.214357
11	Bagging Regressor	2.207423e-01	0.217538	0.346406	0.311403
12	Adaboost Regressor	1.912828e-01	0.207720	0.433632	0.342481
13	Voting Regressor	1.995077e-01	0.196563	0.409279	0.377797
14	Gradient Boosting Regressor	1.319312e-06	0.224828	0.999996	0.288327
15	Stacking Regressor	2.564746e-01	0.222765	0.240607	0.294857
16	Extreme Gradient Boosting Regressor	2.238303e-01	0.292294	0.337263	0.074770
17	TPOT	2.699849e-01	0.286383	-0.269985	-0.286383

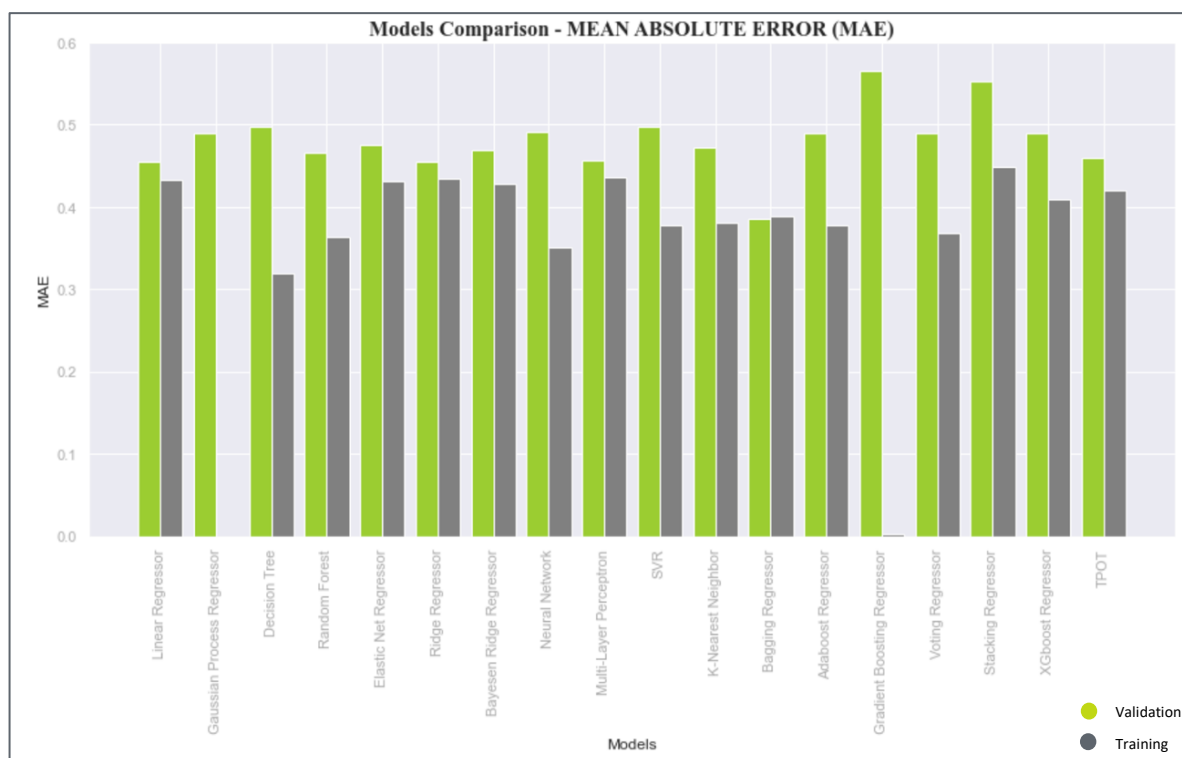


Figure E.4 – MAE of all models

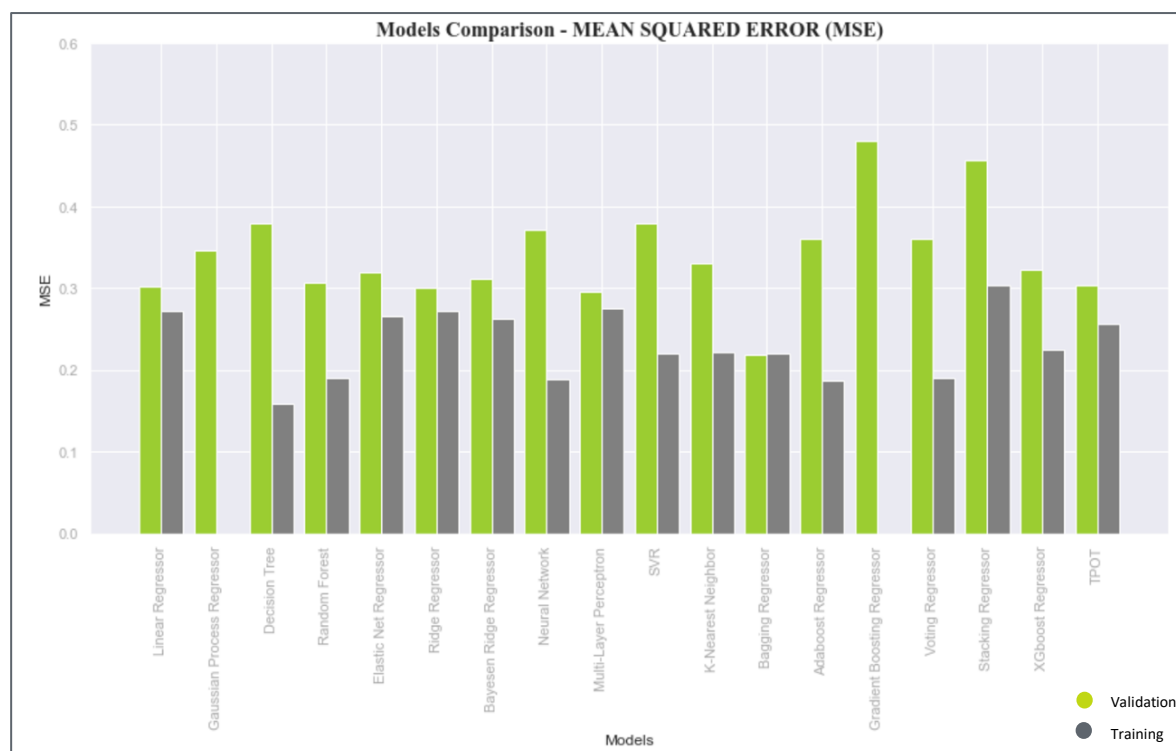


Figure E.5 – MSE of all models

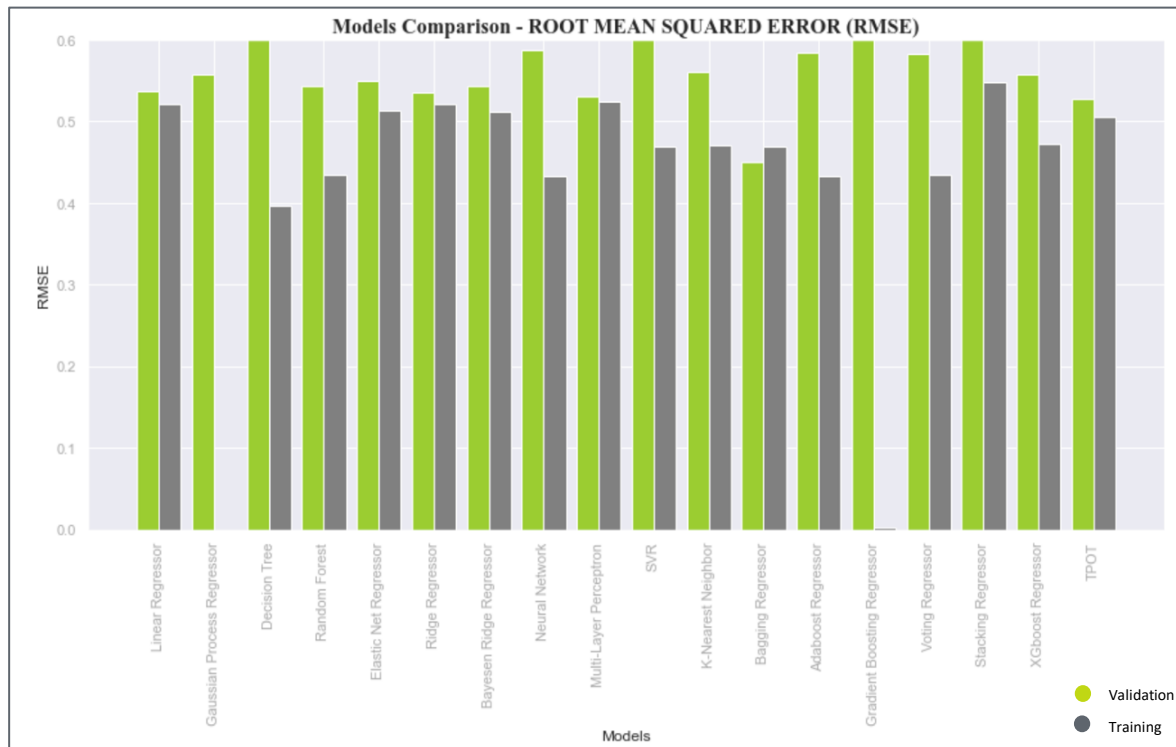


Figure E.6 – RMSE of all models

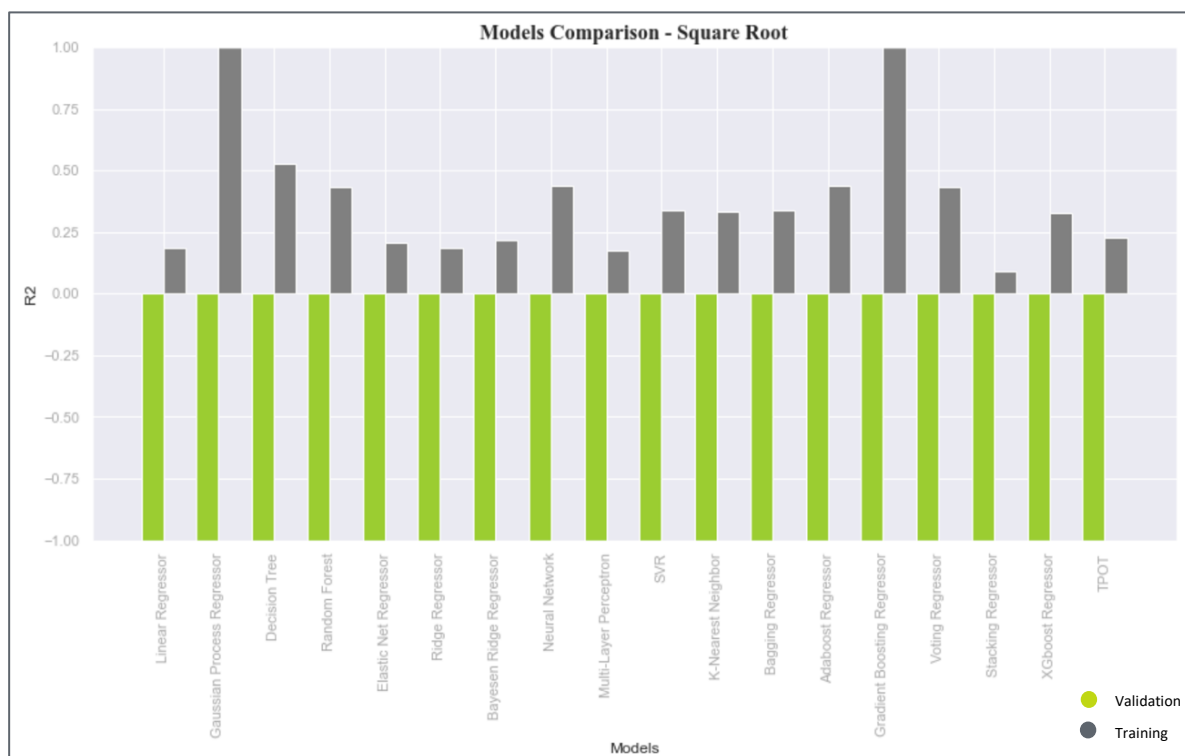


Figure E.7 – R² of all models

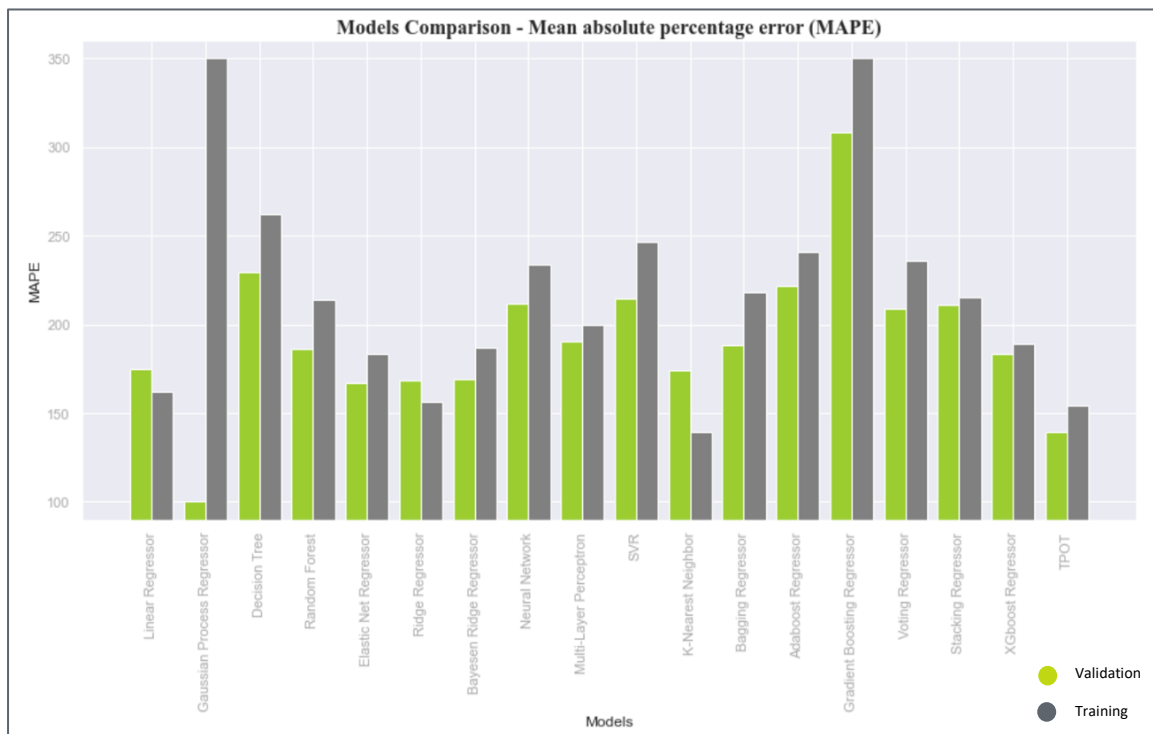
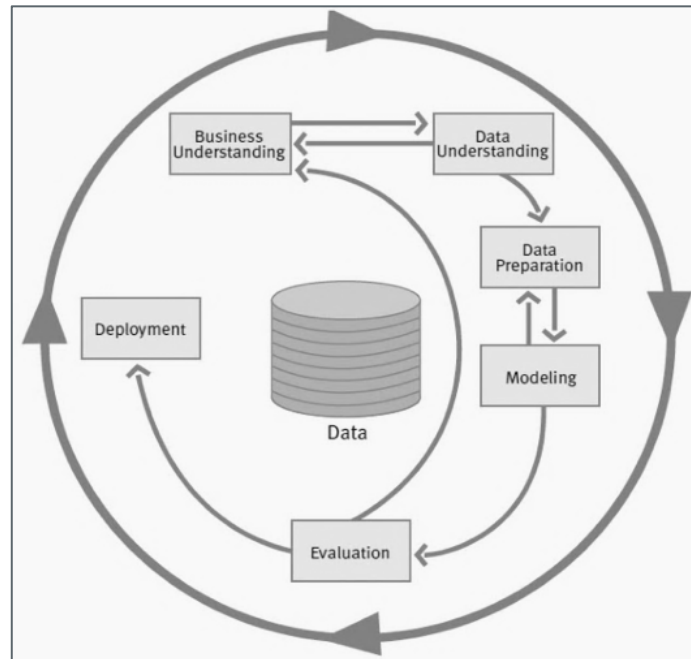


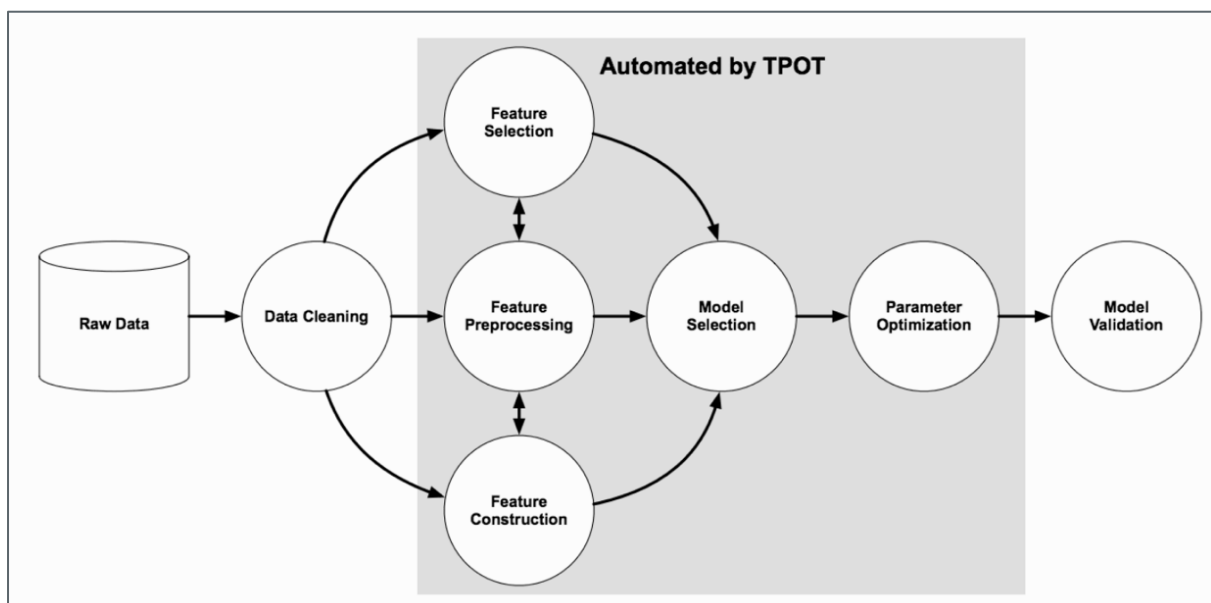
Figure E.8 – MAPE of all models

ANNEXES

ANNEX 1: CRISP-DM



ANNEX 2: MACHINE LEARNING PIPELINE AUTOMATED BY TPOT



ANNEX 3: GANT CHART

