UNIVERSIDADE NOVA DE LISBOA

MScCBBi

MASTER IN
COMPUTATIONAL BIOLOGY
& BIOINFORMATICS

Ana Patrícia Fernandes Brás da Silva
MSc in Integrated Studies of the Oceans

A Study of Machine Learning for Artificial Intelligence-Based Enzyme Classification.

Nov, 2023

NOVA SCHOOL OF SCIENCE & TECHNOLOGY
NOVA MEDICAL SCHOOL
NOVA IMS Information Management School
DESDE 1902 INSTITUTO DE HIGIENE E MEDICINA TROPICAL UNIVERSIDADE NOVA DE LISBOA

Ana Patrícia Fernandes Brás da Silva

Graduated in Integrated Studies of the Oceans

A Study of Machine Learning for Artificial Intelligence-Based Enzyme Classification.

Dissertation to obtain the Master's Degree in Computational Biology and Bioinformatics

Supervisor: Leonardo Vanneschi, Professor, NOVA IMS

Co-Supervisor: Isabel Rocha, Professor, ITQB NOVA

Jury:

President: Diana Lousa, Principal Researcher, ITQB NOVA

Opponent: Nuno Lourenço, Assistant Professor, DEI Coimbra University

Instituto de Tecnologia Química e Biológica António Xavier

November of 2023

Copyright

Acknowledgements

I would like to first and foremost thank both my supervisors Professor Leonardo Vanneschi and Professor Isabel Rocha for the opportunity to do this thesis and for guiding me throughout the length of the project.

I would like to thank Professor Diana and Professor Manuel for helping me at the beginning with the data and code for running in the cluster system. With the end of this project I have now stopped haunting the cluster system with my presence, freeing up several computers. For now at least. I am also grateful to Davide Farinari for taking the time to explain how to run the TPOT algorithm best.

For the people in the SSBio lab, thank you for being such good company and colleagues and showing me the wonder that is Zé Varunca. My weight and wallet have still not recovered.

Several close friends had to suffer through my excessive talks about this dissertation and my appreciation for their patience and support cannot be measured. Andreia R., Andreia P., Querido, Daniela e Leonor, my life would be poorer without knowing you guys. A special shoutout has to be given to a friend I got while doing this masters. Victoria Gil, I am so glad to have met you and your friendship made this masters more enjoyable.

Finally, I wish to thank my family. My grandparents who have always been supportive, if a bit confused on what I'm actually doing. My brother, who let me borrow his gaming computer to run a few experiments that my laptop couldn't handle. My dad for always convincing me to get some fresh air by bribing me with delicious food and restaurants. And finally my mom, who had to deal with her son, daughter and husband discussing algorithms and programming at the dinner table.

Thank you all.

Abstract

Enzyme Commission (EC) numbers prediction models allow for the prediction of an enzyme's function by using only its sequence. They typically have 5 Levels with Level 0 to distinguish if the sequence is an enzyme and the Levels 1 to 4 analogous to each EC number digit. For the most part, these models use machine learning (ML) methods in order to predict the EC numbers, keeping the same structure across the four digits, despite the difference in data composition through the digits. With the hundreds of EC number combinations possible, creating a new ML pipeline for each EC number digit has been impractical as it would need to be optimized for each class which takes time and expertise to reach the best pipeline. Tree-based Pipeline Optimization Tool (TPOT) is an automated-ML algorithm that uses Genetic Programming (GP) in order to create and optimize ML pipelines for the data given. With it, the creation of an optimized ML pipeline becomes easier and faster. TPOT was used to create two EC number prediction models, with a personalized ML pipeline for each possible digit, up until the third EC number digit. These two models, C40 and Swiss, were created from Swiss-Prot data, with the first using a clustered dataset from the latter. These models were then compared between each other and four previously published models (ECPred, DeepEC, HECNet and Benz-WS) using a created validation dataset. The pipelines returned by TPOT differed between both models, despite the only difference in the dataset being the clustering, which was expected due to TPOT's stochastic nature. The created models were comparable to, or surpassed, published models. This concludes that TPOT can be promising to use for the creation of models for biological data.

Keywords

Machine Learning; TPOT; Genetic Programming; EC number; Enzymes

Resumo

Modelos de previsão de números da *Enzyme Commission* (EC) permitem a previsão da função de uma enzima ao usar apenas a sua sequência. Tipicamente têm 5 níveis: nível 0 para distinguir se a sequência é de uma enzima e os níveis de 1 a 4 como análogos a cada dígito do número EC. Em geral, estes modelos utilizam métodos de *Machine Learning* (ML) de forma a prever este número e mantém a arquitetura do modelo constante, apesar da diferença da composição de dados nos diferentes números. Com as centenas de combinações de números EC possíveis, a criação de uma *pipeline* ML para cada dígito do número EC é pouco prático, pois esta teria de ser otimizada para cada classe para chegar à melhor *pipeline*, o que leva tempo e experiência. *Tree-based Pipeline Optimization Tool* (TPOT) é um algoritmo de ML automatizado que utiliza Programação Genética (GP) de forma a criar, e otimizar, as *pipelines* de ML. Com esta ferramenta, a criação de uma *pipeline* otimizada torna-se mais fácil e rápida. TPOT foi utilizada na criação de dois modelos para a previsão de número EC, com uma *pipeline* de ML personalizada para cada dígito até ao nível três. Estes dois modelos, C40 e Swiss, foram criados a partir de dados do Swiss-Prot, com o primeiro modelo a utilizar o mesmo conjunto de dados que o Swiss mas que sofreram *clustering*. Estes modelos foram comparados entre eles e quatro outros modelos (ECPred, DeepEC, HECNet and Benz-WS) a partir de um conjunto de dados de validação criado para o propósito. As *pipelines* obtidas alteraram-se entre os modelos criados, algo esperado devido à natureza estocástica do TPOT, e estes foram comparáveis, ou ultrapassaram, os restantes modelos. Conclui-se que o TPOT pode ser promissor na criação de modelos para dados biológicos.

Palavras-chave

Machine Learning; TPOT; Programação Genética; EC number; Enzimas

Index

List of Figures

List of Tables

List of Abbreviations and Acronyms

AI	Artificial Intelligence
AutoML	Automated Machine Learning
BAC	Balanced Accuracy
CNN	Convolutional Neural Network
DL	Deep Learning
EA	Evolutionary Algorithms
EC	Enzyme Commision
GP	Genetic Programming
HMM	Hidden Markov Model
IUB	International Union of Biochemistry
IUBMB	International Union of Biochemistry and Molecular Biology
kNN	k-nearest neighbor
ML	Machine Learning
NC-IUBMB	Nomenclature Committee of the International Union of Biochemistry and Molecular Biology
NN	Neural Networks
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TPOT	Tree-based Pipeline Optimization Tool

1 Introduction

Enzymes are proteins with catalytic activity involved in accelerating several biological processes essential to life. Their study has yielded several industrial and pharmaceutical products (Meghwanshi et al., 2020; Sharma et al., 2021), though humans have unknowingly used protein enzymes for millennia (McGovern et al., 2004). Unsurprisingly, the discovery, or creation, of new enzymes with a relevant function for the industry and medical fields has garnered interest in the quick and accurate prediction of their function using just basic information such as their sequence.

With the decreasing costs of sequencing technology, an unprecedented amount of raw biological sequencing data has been generated and has become available to researchers, stored in several easy to access databases such as UniProt (Bateman et al., 2023) and BRENDA (Schomburg et al., 2014). UniProt allows researchers to deposit protein sequences that can then be used for studies by other research groups, and include the possibility of manually annotating the protein sequences with the current specialist knowledge. Swiss-Prot (Boeckmann et al., 2003) is the database section belonging to the UniProt database, in which specialists have annotated sequences with current knowledge on them. It is considerably smaller than the non-reviewed section, TrEMBL (Boeckmann et al., 2003), only being less than 1% of the total of Uniprot (as of September 2023). These manual annotations can include a variety of information such as the subcellular location, presence of variants and function.

The fraction of curated protein sequences compared to the universe of known sequences has led to various attempts in using quicker strategies to obtain the possible function of an enzyme when trying to find proteins of interest. A common strategy is to use the similarity of a sequence of an unknown protein function with known protein sequences, with the assumption that similar proteins tend to follow a similar function, though what minimum threshold of similarity should be considered to associate the same function to a sequence is highly variable (Higdon et al., 2010). A popular choice for this strategy is by using the pair-wise alignment algorithm BLAST (Altschul et al., 1990), in which a heuristic search on the database will return similar sequences to the query sequence. A recent methodology, Seq2Enz, is an example of a method which utilizes an adapted BLAST methodology specifically to predict enzyme functions (Pathak & Jayaram, 2022).

Even within similarity searches there is a variety of strategies for annotating a protein's function. Hidden Markov model (HMM) is just one of several approaches used in the Machine Learning (ML) field for protein function prediction. It is a probabilistic approach well established in the bioinformatics field, especially in regards to sequence analysis (Krogh et al., 1994; Sasikumar & Kalpana, 2016). Different models use this technique in their own structure e.g. HMMER, a model specialized in homology search using HMMs (Prakash et al., 2017), has shown to be successful in its objectives. Specialized models for protein functions prediction such as Benz-WS (Baldazzi et al., 2021) and HMMeta (Gbenro et al., 2020) have successfully adapted HMM for this purpose, with Benz-WS reporting higher scores comparing with other ML models for the same objective.

Since at least the nineties (des Jardins et al., 1997), different ML models have been created for uncovering protein/enzyme functions without using similarity approaches as part of the input. With the increasing knowledge of enzyme properties, the translation of said properties into a readable input for ML models have led to a proliferating field of encoding options, from binary encoding strategies to complex protein embeddings (Jing et al., 2020). Along with this, the advances in the ML field have led to a variety of strategies and architectures' combinations as possible options for when creating models. These range from more traditional ML algorithms, such as Random Forests and Support Vector Machines (SVM), to Deep Learning (DL) architectures such as Convolutional Neural Networks (CNN) (Bonetta & Valentino, 2020). With so many possibilities, the choices when constructing the model and the parameters to be used when training can be overwhelming and take time and specialized knowledge to optimize them for the purpose.

A common representation of an enzyme's function is by the EC number, a hierarchical four-digit number which describes a function by the reaction(s) catalyzed by the enzyme in question (McDonald & Tipton, 2023). It is a useful representation that has been used when creating ML models for predicting enzymes' functions (Baldazzi et al., 2021; Dalkiran et al., 2018; des Jardins et al., 1997; Y. Li et al., 2018; Memon et al., 2020; Ryu et al., 2019). Others have chosen Gene Ontology terms instead (e.g. You et al., 2018). For the EC number predictions, models have differed between predicting just the first couple of numbers (e.g. des Jardins et al., 1997) to the complete four digits (e.g. Baldazzi et al., 2021; Dalkiran et al., 2018; Memon et al., 2020), as the latter two digits, and especially the fourth, are harder for prediction as, with the increased specificity as the digits advance, the available sequences for training get scarcer, making it harder to train and neither overfit or underfit. Despite the difference in data amounts and characteristics between the EC numbers, the prediction models tend to use the same ML pipeline approach across all digits (e.g. Memon et al., 2020).

This thesis' purpose is the creation of an EC number predictive model using TPOT (Olson et al., 2016). TPOT should return an optimized ML pipeline for each EC number, resulting in a model with several different algorithm combinations primed for the data available at each digit, constructed using a 'top-down' approach (Silla & Freitas, 2011) where each EC number digit will be trained to distinguish between the next digits. Two models will be created, one with a clustered dataset of Swiss-Prot and the other model with the non-duplicated sequences of the whole of Swiss-Prot. Models construction also includes the choice of protein encodings, of which Protein embeddings originating from Transformer models, were chosen to encode the protein sequences into a numerical input. Each model will have 4 levels: Level 0 will predict whether a sequence is an enzyme or not, Level 1 is associated with the first EC number, Level 2 with the second EC number and Level 3 with the third. When the models are complete, these will be compared to each other and four other EC number prediction models – ECPred, DeepEc, HECNet and Benz-WS – using a validation dataset composed of newly created Swiss-Prot entries from January 1st to May 18th of 2023 and four commonly used scoring metrics (Accuracy, F1, Precision and Recall) and adaptations of the latter three measures specifically for hierarchical models. The created models are expected to either have similar or higher scores, at each Level, as the previously mentioned models.

The thesis is organized in several chapters. Chapter 2 is the theoretical background, which includes the historical and technical background - such as the explanation of how TPOT works - needed for the understanding of this thesis, followed by Chapter 3, which reviews previous prediction models' characteristics such as features representations and different classifiers. Chapter 4 describes the detailed methodology used for this work. Chapter 5 reports the results, from the TPOT pipelines' scores for training and pipeline validation for the different EC numbers, to the comparison between models. Chapter 6 discusses the results and concludes the thesis.

2 Theoretical Background

2.1 EC number: standardization of function depiction

The first time an enzyme was described was in 1833 by Payne and Persoz, who proposed a simple naming rule of adding *–ase* at the end of the name. With such a simple rule to follow, as the number of enzymes discovered and characterized grew. However, the need for unique, descriptive standardized naming conventions became necessary. By the mid 1950's there were already several enzymes with multiple or misleading names, which led to suggestions in how to keep a consistent naming and classification system (Tipton & Boyce, 2000). In an attempt to solve this conundrum, the then IUB created the Commission on Enzymes in 1956, with the ambitious idea of establishing a set of rules and recommendations for enzyme nomenclature. By 1961, a report with the recommendations was published and within that report, enzymes could have a systematic name, trivial name (nowadays changed to accepted name) and a four digit classification system that detailed the function of the enzyme – the EC Number (McDonald & Tipton, 2023; Thompson, 1962; Tipton & Boyce, 2000).

This four digit classification takes into account the reaction catalyzed by the enzyme, with the first digit defining the enzyme class (of which there were only 6 in the original proposal – Table 2.1), and then subdivided further in subclass and sub-subclass (second and third number respectively) in order to detail the type of reaction and finishing with the fourth signifying the substrate, that serves to identify the enzyme within the sub-subclass (Thompson, 1962).

Table 2.1 The seven EC classes and their associated number

EC class number	*EC class*
1	Oxidoreductases
2	Transferases
3	Hydrolases
4	Lyases
5	Isomerases
6	Ligases
7	Translocases

Recognized enzymes and their recommended EC number were gathered in the NC-IUBMB Enzyme list, which then had to undergo several updates in printed editions as more and more enzymes were recognized and identified, up until after the last printed version in 1992 (Figure 2.1) started to prove itself to be unwieldy and it was decided to make the list available online. It would also allow for a faster registration of new enzymes to be made available to the public (McDonald & Tipton, 2023). With the higher amount of known enzymes, the system also had to have been updated so as to support the new knowledge so new subclasses and beyond were created. It was only in 2018 however, that a new enzyme class was created - Translocases (Class 7) – and consequently, several more subclasses (and sub-subclasses), that resulted in the update of several EC numbers (Balaji, 2021).

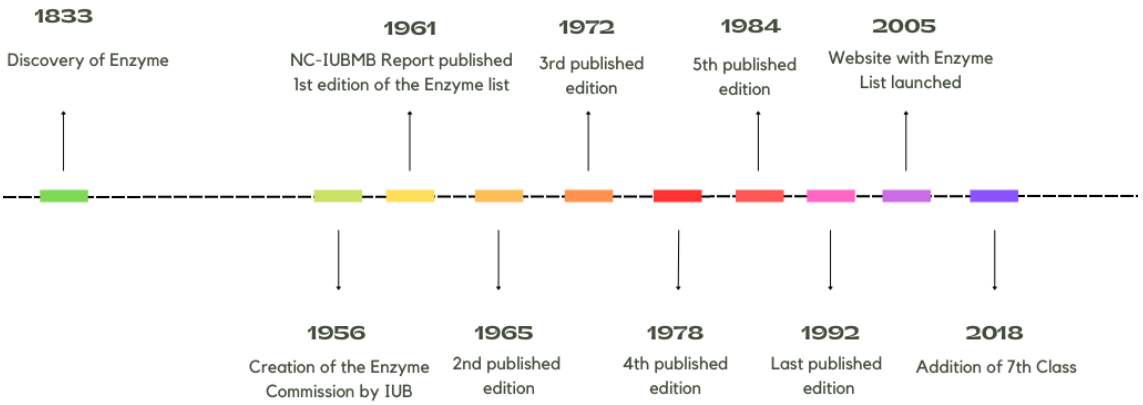


Figure 2.1 Timeline from the discovery of enzymes in a lab, the establishment of the nomenclature timeline, to the last major alteration in the EC number organization.

Though not under the direct responsibility of the IUBMB enzyme committee, there are a variety of databases which use EC data, such as BRENDA (Schomburg et al., 2014) and UniProt (Bateman et al., 2023). This later database has a great variety of protein sequences, with a section of manually reviewed sequences (Boeckmann et al., 2003) but the majority of them un-reviewed. With the use of high-throughput sequencing technologies generating data in droves, the amount of non-reviewed entries grow far faster than these are characterized.

With a wealth of untapped knowledge and interest in the discovery of enzymes for several industries, researchers have created bioinformatics tools in order to automatically annotate these uncharacterized protein sequences and quickly but accurately find enzymes that may be of interest. Of high interest is the discovery of the EC number of unknown enzymes, so it should be unsurprising that researchers have turned to machine learning to create prediction models for the EC number using just its sequence.

2.2 Machine Learning in Biology

A subset of AI, ML are computer programs from which machines can “learn” from data without the need for explicit programming for it, using a measure of performance. In practice, the models learn by arriving at solution with the minimum error possible with that specification. It is a field with an ever growing quantity of techniques and used in a variety of real-life problems. In bioinformatics, ML has been ever more necessary as more data becomes available to allow the study of complex biological problems (Olson et al., 2018). For biomedical research, the hope is high in the use of ML models for disease diagnosis and treatment e.g. (Iqbal et al., 2021).

ML algorithms can fall on one of three categories: supervised learning, unsupervised learning and reinforcement learning. For the first case, the program is trained using labeled inputs so as to learn a function that can explain the data (e.g. Logistic Regression). Unsupervised learning uses unlabeled data for training and discovers by itself the structure of the data (e.g. clustering). Reinforcement learning uses feedback (positive or negative) so as to repeat the process considering said feedback in order to approximate to the solution needed for the problem. Examples of each category can be easily found in

the bioinformatics field e.g. (Dalkiran et al., 2018; Ding et al., 2022; Song et al., 2021), though for the purpose of this thesis, only supervised learning will be used.

Within supervised learning there is also the choice of whether the problem is a Classification problem (the targets can be labelled in distinct categories) or a Regression problem (the targets are continuous values).

With the identification of the category of the problem and the data available, the creation of an ML model then follows several steps:

- Data Preparation: exploratory analyses of the raw data in order to identify and deal with missing or mislabeled data points, either by removing or correcting them. For data that is not naturally numerical (e.g. protein sequences), this may also include encoding the data into a numerical vector.
- Data transformation: choice of the relevant features and/or transforming the features to be able to obtain information (e.g. normalization). This step may also involve the creation of new features if needed.
- Model Selection: choosing between different ML models to find the best fit for the data. This involves comparing the trained models between each other and choosing the one with the best results by, for example, scores.
- Parameter Optimization: tweaking the parameters of the model to improve its performance on the data.
- Model Validation: the chosen, fitted, trained model is tested on data unused for training or testing and has its performance evaluated.

(Olson et al., 2016)

These steps can be very time consuming and monotonous, with some trial-and-error involved to optimize the final model into the best it can be. ML experts narrow the time due to experience but it is still a repetitive process.

In order to minimize human error and simplify the process, several Automated Machine Learning (AutoML) tools have popped up in recent years with the purpose of doing the most repetitive of the steps, returning the best model it could find for the data given by evaluating several pipelines between themselves in an automatic way (Truong et al., 2019).

2.3 Evolutionary Algorithms (EA)

The rising complexity of optimization problems across data-heavy fields have resulted in several different tools to attempt to solve the local optima problem. EA belong to the Evolutionary Computation subfield and contain several optimization algorithms created by taking inspiration on Charles Darwin's Theory of Evolution. These are stochastic methods that generate new solutions within a search space and evaluate them using fitness evaluation (Agoston E. Eiben & Smith, 2015). By their stochastic nature, it is not a guarantee that the algorithm will find the actual best solution but they work well in situations where the computational capability is limited and the available information is lacking (Vikhar, 2017)

These algorithms use concepts of biological evolution such Selection, Mutation and Recombination (or Crossover) (Table 2.2) and adapt them as operators in order to find the best solution by having a population of individuals (possible solutions) evolve across several generations (runs of the algorithm).

Table 2.2 Concepts of Selection, Crossover and Mutation within an Evolutionary Algorithm context.

Selection	The individuals (possible solutions) selected for the following generation tend to be random, with influence from their fitness (score). The higher the fitness, the higher the probability of choosing the individual for the next generation.
Recombination	Individuals have chromosomes (the solution features) and there is a typically high chance of two individuals to cross, producing children (new solutions) that may have a better - or worse - score. Corresponds to sexual reproduction.
Mutation	Individuals' chromosomes have a chance of mutating (changing) creating a new solution. Corresponds to asexual reproduction.

The following is a general explanation of how they work:

- A Population of N individuals is created (usually randomly).
- The Population undergoes a Selection process wherein n individuals (less than N) are selected and transferred to a newly created Population P_1.
- Individuals of P_1 create new individuals through recombination and mutation events up until the Population P_1 reaches the same number of N individuals.
- P_1 individuals are evaluated.
- Process repeats until a previously established termination condition is fulfilled.

Once the termination condition is fulfilled, the best individual of the final population is considered the best solution to the problem (A E Eiben & Smit, 2012).

2.4 Genetic Programming

Genetic Programming (GP) is a technique belonging to the EA field. Where other techniques of the EA field evolve solutions from data to solve problems (e.g. Genetic Algorithms), GP evolves programs with the objective to arrive at the program combination that obtains the best solution for the data inputted (Vikhar, 2017). There are several representation types of GP (e.g. Tree-based GP, Stack-based GP, Linear GP) some more used in certain fields than others.

Tree-based GP is a common representation for the GP algorithm, wherein the nodes of the tree are represented as operators and the leaves as variables, such as numbers, letters etc. In this type of representation, the recombination (also called crossover) event can be visualized as an exchange of

two trees subtrees, creating new combinations (Figure 2.2). The mutation event is the alteration of a node into another e.g. a multiplication operator mutating into a subtraction one.

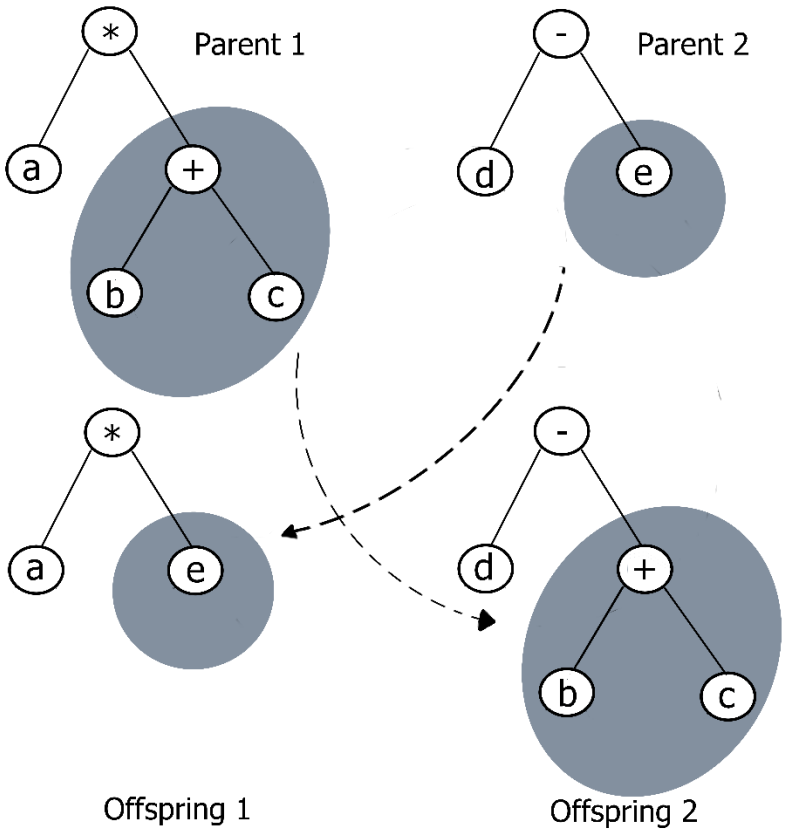


Figure 2.2 Schematic of a crossover event of a tree-based genetic programming population. Two trees in an existing population (parents) randomly exchange a subsection between each other (grey circles), creating two new trees (offsprings).

Despite the different types of GP representation, the structure and operators (selection and genetic operators) are generally the same (Ahvanooey et al., 2019).

2.5 TPOT

TPOT is an AutoML tool where the ML steps from data transformation to the parameter optimization steps are automated, including tasks such as Feature selection or Feature preprocessing (e.g. normalization of the features given). It uses the Tree-based GP technique (implemented through the Python package DEAP) and the python package scikit-learn algorithms (Pedregosa et al., 2011) as operators. The tool is specific to supervised learning, and can handle either Classification or Regression problems (Olson et al., 2016).

In its' default parameter and configuration, TPOT evaluates ten thousand pipelines in a complete run. This thoroughness, consequently, can lead to long run times when using larger datasets (Radzi et al., 2021; Truong et al., 2019) but the tool is made to easily change not just the parameters, such a number of generations, size of population, mutation and selection rates, but also to restrict the created pipeline algorithms to just a chosen few. Altering the configuration from the default in some cases may lead to worse results however. Radzi et al. (2021) studied different TPOT configurations, from the default to configurations of only one classifier, which were compared between each other and with other models,

with a radiomics dataset, concluding that the default TPOT configuration, despite taking longer and being computationally heavier, led to better pipelines that non-default TPOT.

New extensions of TPOT have been created to attempt to solve or improve certain problems. Parmentier et al. (2019) suggested their adaptation TPOT-SH as a solution lowering the amount of time needed for larger datasets runs by adding to the TPOT algorithm an adaption of another technique; TPOT-MDR (Sohn et al., 2017) was developed specifically for genetic analysis studies and was added to the TPOT API, similarly to TPOT-NN (Romano et al., 2021), an extension created to add Neural Networks (NN) to the TPOT pipeline creation process.

For clinical research, TPOT has had promising results on the creation of prediction models (Dou et al., 2021; Orlenko et al., 2018) and another extension was created to better deal with this type of data (Manduchi et al., 2020).

3 Previous Models

EC number prediction models are a Classification problem where the sequence is an input that returns a result, which would either be an EC number or informing the sequence is not of an enzyme.

Since at least the 90's there have been models for predicting the EC number. The first model that did not use just sequence similarity for predicting the function was published in 1997, with an impressive reported accuracy of over 70% for predicting the first digit of the EC number and 68% for the second digit (des Jardins et al., 1997).

As a consequence of the long time these models have been studied and created, there is a wide variety of them, each improving as time and new techniques become available. An example of the different strategies can be as simple as how the classifiers are trained. These models can be divided in hierarchical or non-hierarchical depending on how they were constructed: if the classifier had a model trained for each EC number division this is an hierarchical classifier whereas if it was trained with the complete EC number, it is non-hierarchical (Ferdous et al., 2022). Other differences can be on how the data is represented as input for the model or even what type of classifier, or ensemble of classifiers, are used for the model itself (Table 3.1).

3.1 Features

When creating a model, the choice of the data is the essential step for a successful model. Without good, representative data, the model will not be able to find a solution for the problem given and will only give useless results. For non-numeric data, such as protein sequences, which are represented by letters assigned to each aminoacid present in the sequence, there is a need for transforming the data in numerical vectors.

For protein sequences, there are several representations available with different information present. They may be divided in categories: Binary encoding, Psychochemical Properties encoding, Evolution-Base encoding, Structure-Based encoding and Machine-Learning encoding. Within each category there is a wide variety of choices of encodings, with different levels of complexity and information (Jing et al., 2020).

Different encodings will have their own advantages and disadvantages. One-Hot encoding for example, is a very simple type of sparse encoding. Each aminoacid is identified in a binary vector of length 20 (21 if there are unknown/dummy aminoacids) by the position of the 1, with the rest of the vector filled by zeroes (Spänig & Heider, 2019). It does not have any additional knowledge on the protein besides the aminoacid itself; yet, it is a common encoding to appear alongside others in EC number prediction model (Y. Li et al., 2018; Memon et al., 2020; Ryu et al., 2019) even despite its simplicity and high multi-dimensionality (Jing et al., 2020).

Generally, the strategy for the protein encoding choice in these models is to use several different representations instead of just a single one, selecting which features could have more relevance for the

prediction (Bum et al., 2007). Other information such as Functional and Protein Family Domains can also be added as features in these models (Y. Li et al., 2018; Memon et al., 2020).

Recently, however, unsupervised protein embeddings have been gaining more attention as input features in models. Protein sequences themselves are rich in information as, even though it is just one type of representation of proteins, information about, for example, its structure and function, are determined by the sequence itself and, as such, should be encoded in it (Guo et al., 2022). Researchers are using architectures used in Natural Language Processing along with vast amounts of raw proteins sequences as data in order to try to obtain a representation that contains that inherent information.

These models' purpose is to extract meaning from written texts (or voice inputs) to then be used for a variety of different tasks, such as translation of a language (Hirschberg & Manning, 2015). These models differ in their architecture and, consequently, in how they deal with the data. More advanced models such as ELMo (LSTM) and BERT (Transformer) are context-based, able to learn how words apply in a given context and give an embedding considering that context. Context-free strategies such as bag-of-word from Word2Vec associate a word to an embedding with no regard to the context it is used (Ethayarajh, 2019). Both LSTM and Transformer architectures led to big leaps in the Natural Language Processing field, with the latter a newer architecture that has outperformed Recurrent Neural Networks (RNN) in several tasks (Gillioz et al., 2020).

When using these type of architectures for protein embeddings, the models are trained with protein sequences datasets such as UniRef clusters (Suzek et al., 2015). Language Models tend to have the pre-training as unsupervised learning, where the models learn the inherent knowledge within the sequences by discovering the patterns and, for the models that are advanced enough, the context of said patterns (Brandes et al., 2022; Elnaggar et al., 2021). When using the model on a sequence, it returns a vector representation describing the sequence– the embedding – which can then be used as input in models. How much knowledge is extracted from the sequences is dependent on the data available when training and the models chosen, as more advanced models should retrieve more information to include in the embedding e.g. BERT models, with their Transformers architecture, should have retrieved more information in its embedding than ELMo models (Cui et al., 2021). Studies done with embeddings have affirmed that these models have learned subjacent evolutionary information (Marquet et al., 2022) and have shown to be useful in protein function prediction (Oliveira et al., 2023; van den Bent et al., 2021), subcellular location (Stärk et al., 2021) and drug-target interactions (Zheng et al., 2022).

There are several protein Language Models' embeddings available (e.g. Elnaggar et al., 2021; Heinzinger et al., 2019; Rives et al., 2021; Strodthoff et al., 2020) and Python packages such as bio-embeddings (Dallago et al., 2021) allow for easy access to these models for the creation of protein embeddings. Even if one does not wish to spend time embedding a database full of proteins or does not have the computational means for them, UniProt's (Bateman et al., 2023) website has a Download section in which one can download a file with the databas'e ProtT5 model's protein embeddings (Elnaggar et al., 2021), accessible using UniProt's own sequences' accession code.

Future protein models may start to use protein embeddings instead of an ensemble of different encodings as these become more and more complex and widespread. UDSMProt, in fact, successfully used its own protein embedding as its only feature representation for its EC number prediction model (Strodthoff et al., 2020) and DAttProt used an adapted Transformer encoder for its own encoding mechanism (Lin et al., 2022). TEMPROT used a Transformer model's embeddings from ProtTrans (Elnaggar et al., 2021) instead of their own embedding model like the previous mentioned models, for protein function annotation prediction (Oliveira et al., 2023).

3.2 Classifiers

Similarly to the features, the range of possible classifiers for use in the EC number predictive models is ever evolving, with complexity increasing as more possibilities are added.

Support Vector Machine (SVM) and K nearest neighbors (kNN), are popular choices of classifiers (Cai, 2003; Dalkiran et al., 2018; S. Kumar et al., 2015; Nath & Mitchell, 2012; Resende et al., 2012; Shen & Chou, 2007; Y.-C. Wang et al., 2011; Watanabe et al., 2020), followed by others such as Naïve Bayes (Borro et al., 2006) and ensemble methods such as random forests (C. Kumar & Choudhary, 2012). Bonetta & Valentino (2020) presents a more detailed review of the ML algorithms used in protein function prediction models.

With the interest in Deep Learning (DL) at an all-time high due to the high accuracy results in several fields (e.g. Transformer's architectures in protein embeddings), it is unsurprising that more recent models have started to use more modern DL architectures in their own classifiers. Table 3.1 presents a column with classifiers of recent EC number prediction models, in which half of the models present DL architectures such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) for prediction.

Though models with DL classifiers have gained popularity and have shown to have good results, other models, such as Benz-WS, boast similar, if not better scores with their different strategy (Baldazzi et al., 2021). ECPred, a model with an ensemble of classifiers, is a representative model others use to compare their own performance even today (Baldazzi et al., 2021; Lin et al., 2022; Memon et al., 2020; Ryu et al., 2019; Strodthoff et al., 2020)

Table 3.1 Recent examples of EC number prediction models with details of their training data, input, classifier and up to what EC digit it can predict, with level 0 predicting whether the sequence is of an enzyme or not and levels 1 to 4 associated each with the EC number. CNN – Convolutional Neural Networks; LSTM – Long-Short Term Memory; PSSM – Position specific scoring matrix; SVM – Support Vector Machines; RNN – Recurrent Neural Network.

Model	Data	Input	Classifier	Levels of predictions	Reference	Comments
DAttProt	Swiss-Prot	-Associated Transformers embedding	CNN; Position-wise shared Linear Inference	Levels 0 to 2	(Lin et al., 2022)	Installable model, Python based
Seq2Enz (mask BLAST methodology)	Swiss-Prot	-Structural, chemical side-chain properties (Jayaram, 2008) -Sequence	No classifier, Homology based (BLAST adaption)	Levels 0 to 4	(Pathak & Jayaram, 2022)	Webserver available, job never finished and gave prediction.
Benz-WS	UniProt TrEMBL	-Sequence	Hidden Markov Model (Reference sequence) Pfam model (architecture)	Levels 0 to 4	(Baldazzi et al., 2021)	Webserver available.
HECNet ¤ mlHECNet (2021)	Swiss-Prot (Max. 40% similarity)	-PSSM -One-Hot Encoding -Functional Domains* Raptorx-Property tool (S. Wang, Li, et al., 2016) for: -Disordered Regions -Secondary Structure -Solvent Accessibility	Siamese Triplet Network (CNN)	Levels 0 to 4	(Memon et al., 2020) (Khan et al., 2021)	Webserver available.
UDSMProt	Swiss-Prot (Custom EC40 and EC50)	-Associated embedding (RNN)	Finetuned RNN	Levels 0 to 2	(Strodthoff et al., 2020)	Installable model, Python based.

DeepEC	Swiss-Prot TremBL	-One-Hot Encoding (embedding layer transforms previous encoding)	CNN (x3) If CNN contradicts - Homology	Levels 0 to 3	(Ryu et al., 2019)	Installable model, Python based.
ECPred	Swiss-Prot (UNIREF50)	- SPMap - BLAST-kNN - Pepstats-SVM	SPMap BLAST-kNN Pepstats-SVM	Levels 0 to 4	(Dalkiran et al., 2018)	Installable model, JavaScript based.
DEEPre ¤mlDEEPre	Swiss-Prot (Max. 40% similarity)	-One-Hot Encoding -PSSM DeepCNF tool (S. Wang, Peng, et al., 2016) for: -Solvent accessibility - Secondary Structures	CNN and RNN (LSTM)	Levels 0 to 3	(Y. Li et al., 2018) (Zou et al., 2019)	Webserver accessible but does not give any prediction, error in the prediction step.
Enzynet	PDB	-Shape representation (algorithm in paper)	3D CNN VoxNet architecture	Levels 0 to 4	(Amidi et al., 2018)	Installable model, Python based.
¤ server.malab.cn/MEC/	Swiss-Prot (Max. 65% similarity)	-PSSM	kNN	Level 1	(Che et al., 2016)	Webserver no longer accessible.
SVM Kumar et al	UniProt (Selected enzymes & non-enzymes)	PROFEAT tool (Z. R. Li et al., 2006) for: -Amino acid composition -Dipeptide composition -Correlation feature -Composition -Transition	SVM	Level 0 to 2	(S. Kumar et al., 2015)	Not available.

		-Distribution -Pseudo amino acid composition (SVM-RFE algorithm to perform feature selection)				
EzyPred	ENZYME (Max. 40% similarity)	-Functional Domains* -Pse-PSSM	OET-kNN	Level 0 to 2	(Shen & Chou, 2007)	Webserver in paper no longer available.

* Functional Domains are by Pfam database (Mistry et al., 2021).

4 Methodology

Data for training, testing and validation were obtained from the UniProt Database (Bateman et al., 2023), specifically the reviewed section Swiss-Prot (Boeckmann et al., 2003). For training and testing, the downloaded section was of all sequences as of October 2022, with a total of 568 002 protein sequences. This amount was reduced by removing homologous sequences (only keeping the first instance of the sequence – removal of 88072 sequences) and, only sequences with more than 50 and less than 5000 aminoacids were considered in the construction of the datasets, leading to the removal of 10112 more sequences. For the actual EC number prediction pipelines, 15441 sequences were removed due to being enzymes with more than 1 EC number associated.

The validation dataset was composed of sequences added to Swiss-Prot from January 1st to May 18th, to a total of 786 sequences, with the only limitation being the length of sequences.

The Bio.SeqIO module from the Biopython package (Cock et al., 2009) was used in order to parse the fasta files obtained from UniProt. These sequences had to be then encoded into numerical vectors in order to be used.

Due to the memory requirements, the following steps were done in ITQB's server, with use of SLURM as a resource management system (Yoo et al., 2003).

Two protein embeddings models were considered for this thesis, both Transformer based: ProtBert and ProtT5 (Elnaggar et al., 2021). These embeddings are arrays of variable length with more than 1 dimension. As the TPOT algorithm can only accept numerical vectors, the raw embeddings were not possible to be used directly so instead, the average of the embeddings was used, an option available in the bio-embeddings package, which currently includes a wide variety of embedding models such ProtBert and ProtT5 (Dallago et al., 2021). This average, that the package names per-protein embeddings, creates a single vector of 1024 length though a global average pooling of the combined embedding. The embedding process was done with the bio-embeddings package, in an Anaconda virtual environment with python version 3.8.10 though only of the ProtBert model was run through. The ProtT5 model embeddings, were available for download at the UniProt website and were used.

The environment used for the embedding process was saved in a YAML file environment.yml for installation ease. Of note is that the bio-embeddings was only possible to be downloaded in a 24 Gb RAM computer, as attempts on 8 Gb and 16 Gb computers resulted in the installation process being killed due to memory.

Both models protein-levels embeddings up until October 2022 were kept in HDF5 format (Koranne, 2011), needing just the UniProt code for accessing and creating the training and validation datasets.

Both models can be accessed at the github repository https://github.com/Ananas-bio/Tpot_ec_prediction, however, due to the size limits imposed in these repositories, only the pipelines

from Level 1 forward were possible to be uploaded, as Level 0 pipelines surpassed this size limitation. This assumes any sequence given will be an enzyme.

4.1 Models

Two models were constructed, one trained with the full amount of sequences available with the limitations previously established while the other model had the dataset more than halved by the clustering of sequences with more than 40% similarity, using the clustering program CD-HIT in the command line (Fu et al., 2012). The latter model, henceforth called the C40 model, uses only one sequence as a representative for each cluster, meaning the longest sequences of the cluster created by the defined similarity threshold. The other model, called Swiss model from here on, has the complete dataset available. Exception is made for the first enzyme/non-enzyme (level 0) model, for which a subsample of 200 thousand proteins (half enzymes half non-enzymes) was used for both memory and time reasons. Due to being less than half the whole dataset, this step was repeated with a different set of proteins to verify if the random choice influenced the training results and the resulting pipelines. A detailed account of the number of sequences present at each pipeline in both models are present at Appendix A 1 and Appendix A 2- When training and testing, the datasets used 70% of the data, leaving 30% for validation of the given pipeline (pipeline validation). The model's creation schematic can be observed in Figure 4.1.

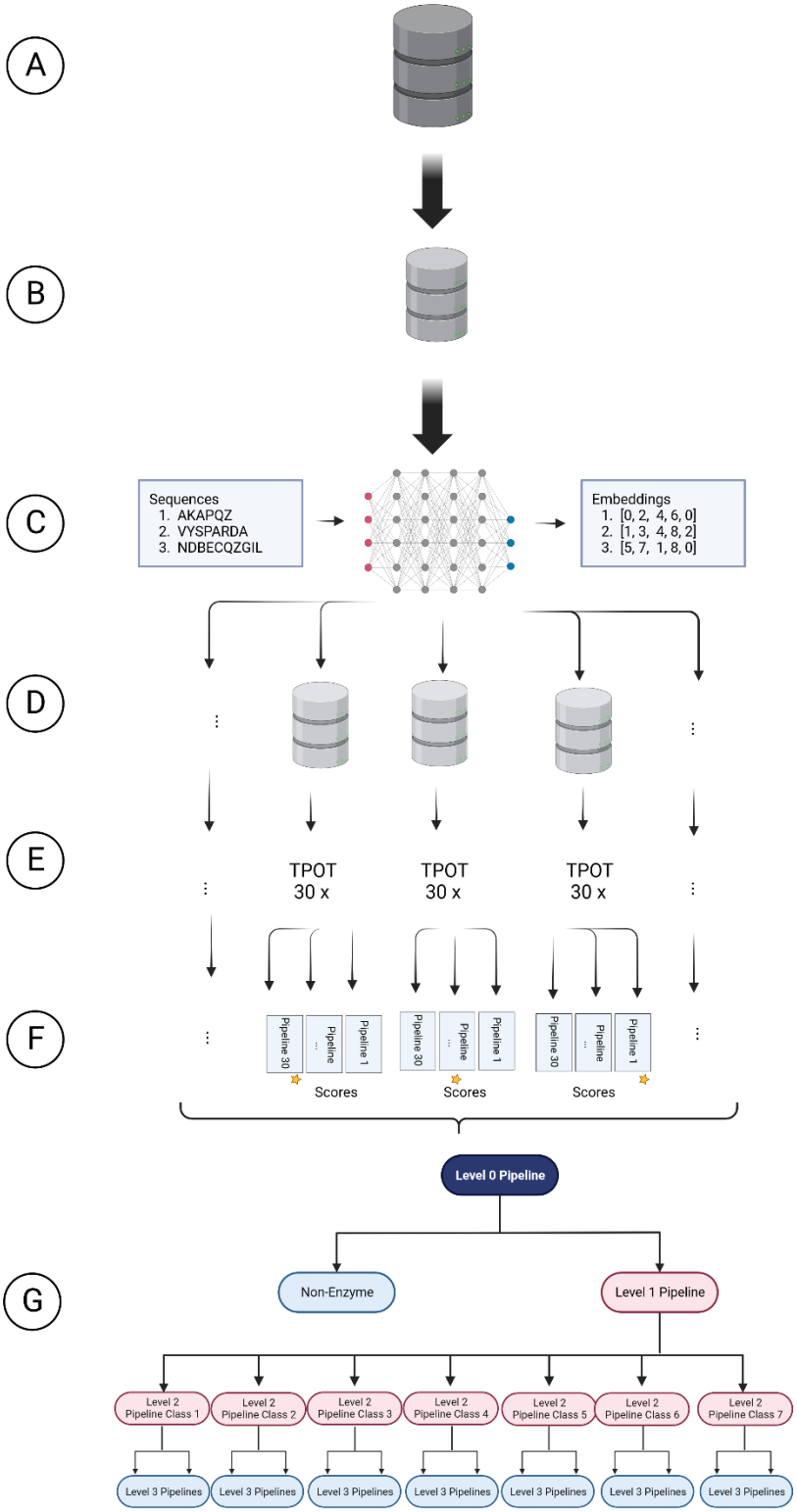


Figure 4.1 Schematic of the models creation. A: database download; B: Cleaning of sequences; C: Transformation of the sequences into embeddings; D: Creation of datasets for training/testing and pipeline validation per level and EC number in latter levels; E: Running of training/testing datasets through TPOT, 30 times; F: Comparison between the training scores of the pipelines given by the TPOT runs, choosing the best scoring one for including in the model; G: Organization of the model, from the Level 0 prediction pipelines to the latter Level 3 pipelines (shortened). Created with BioRender.com

In both, the models have been made for 4 levels. Level 0 is a binary classification problem of whether the sequence given is, or is not, an enzyme. Subsequent levels from 1 to 3 are related to the

EC number classification system: level 1 is the first digit of the EC number, the Class, level 2 the second digit of the EC number, the subclass, and level 3 the sub-subclasses. These models were created by having an optimized pipeline at each level and for each EC number in order predict the following one. These pipelines were created by passing the different datasets through the TPOT algorithm (Olson et al., 2016).

With the exception of the Swiss level 0, the TPOT configuration was consistent throughout the levels, with the following changes to the default TPOT Classifier API configuration: 50 generations, 25 of population, Mutation rate 0.5 and a limit of 4200 minutes for the whole pipeline to finish (in case it reached that time limit, the process would finish regardless of the number of generations it had gone through). For the full Swiss model, the Cross-Validation was lowered from the default 5 folds to 3 folds due to time limitations. TPOT's internal scoring between pipelines was done through balanced-accuracy (BAC) (Pedregosa et al., 2011) due to imbalanced datasets. This BAC metric follows Guyon's et al. (2015) definition.

The TPOT was always run 30 times for each class/subclass. Exceptions were datasets for sub-subclasses prediction in which no sub-subclass reached 50 sequences, the minimum threshold considered for training. These subclasses were removed from the models.

When constructing the models, the chosen pipelines for each level (and EC number) were organized hierarchically, starting with the pipeline for Level 0, in which the choice could be an enzyme or a non-enzyme. In case it was predicted as a non-enzyme, the model would them return non-enzyme sequence. If the sequence was predicted as an enzyme, it would then go through the pipeline for Level 1. Depending on which class was predicted, the pipeline for the predicted class at Level 2 would then be done. Level 3 predictions follow the same logic except for, as mentioned, cases where the subclass did not have enough data for training, which would result in a dash ('-'), signifying unknown sub-subclass.

Another exception case that can give a dash for unknown sub-subclass instead of a number happens in subclasses in which there is a sub-subclass with a high quantity of sequences and it either has no other known sub-subclasses to predict against or the other sub-subclasses present have low quantities of sequences (e.g. a sub-subclass could have 1000 sequences and the rest of the sub-subclasses barely a dozen in total). To deal with these extremely unbalanced cases, these low quantity sub-subclasses were joined together as a negative class and used to train a pipeline of whether the sequence belonged to the high amount of sequences' sub-subclass or not. These negative sub-subclasses could be supplemented (or created in cases where there were no other sub-subclasses present) with sequences of the same subclass in order to reach a similar quantity of sequence as the positive class (the sub-subclass with a high amount of sequences). If the pipeline in these cases returns the prediction as part of the negative class, it will return a dash ('-') instead of a number.

4.2 Scoring

In order to choose between the pipelines returned by TPOT, different scoring metrics were considered. The chosen metrics were obtained by using the Scikit-learn metrics options (Pedregosa et al., 2011).

When comparing between the 30 TPOT originated pipelines, in the binary classifier cases (e.g. the enzyme/non-enzyme level) these metrics were the Accuracy, Precision, Recall and F1-score. When considering the multiclass cases these were Accuracy, Precision Macro, Recall Macro and F1-score Macro, which are the sum of the scores divided by the number of classes present. In all cases the comparisons were done using the validation score of the pipeline. For this thesis the F1-score (either default or Macro) was the score used for choosing the optimized pipeline, as F1.

4.3 Model Certainty Probability

For evaluation of the EC number prediction by the user, both created models also offer the associated probability of certainty for each digit of the predicted EC number. While most of the Classification algorithms included in SciKit-learn have a function already available to access that probability (predict_proba) some, like LinearSVC and SGD, do not. When the best of the 30 models given by the TPOT included either of these classifiers, the pipeline was wrapped in the calibration function CalibratedClassifierCV in order to calibrate the probability, allowing for access to it using the same function.

4.4 Validation

For validating and comparing with other models, the previously described validation dataset had the sequences predicted by the created models and four others: the ECPred model, DeepEC model, HECNet and Benz-WS. Other models, including those for EC number prediction, tend to use flat metrics for each level (e.g. Dalkiran et al., 2018) so the same Accuracy, F1, Precision and Recall scores previously described were calculated for each level and compared between each model, in order to see the performance per level on the same dataset of proteins that none of the models were trained on.

There are reservations on the use of flat metrics for hierarchical models performance as these do not take into account the hierarchy (Silla & Freitas, 2011). There are several possible hierarchical methods that could be considered though none seems to have been widely adapted (Kosmopoulos et al., 2015). The chosen metrics were adaptations of the Precision, Recall and F1 scores for hierarchical models, as proposed by Kiritchenko et al. (2006), calculated with the help of the Python library HiClass (Miranda et al., 2023).

5 Results

This section will be divided into two subsections: the results on models development and the validation of said models. The first subsection will describe the results achieved using TPOT, including the best and worst training runs for each EC number with enough data for training and testing, and the best pipeline validation scores and classifiers. The second subsection will compare, using the validation dataset, the scores of the created models with other published EC number prediction models: ECPred (Dalkiran et al., 2018), DeepEC (Ryu et al., 2019), HECNet (Memon et al., 2020) and BENZ-WS (Baldazzi et al., 2021).

5.1 Model Development

The model follows a hierarchical structure, starting with identifying whether the given sequence is an enzyme and following up with predicting the EC number, with the subsequent classes dependent on the previous prediction. Whereas in other models, whether they follow the hierarchical structure or not, the classifier is kept the same through the classes (Dalkiran et al., 2018; Li et al., 2018; Memon et al., 2020; Ryu et al., 2019, etc.), the purpose of running TPOT was to return the best pipeline for each individual class. Consequently, each model has a variety of different pipelines and classifiers even at the same level, changing according to what is best for the data given.

5.1.1 Results per level

5.1.1.1 Enzyme commission level 0

Level 0 is a binary classification problem with only one of two results: either the sequence given to the model is an enzyme or it is not. The datasets used for training and testing were three: one with the fully clustered data associated with the C40 model and the other two datasets were subsampled from the complete Swiss database (Swiss_1 and Swiss_2). The fitness score (score of the best pipeline in a population in each generation) for both Swiss datasets was higher compared to the C40 dataset, even in the lowest scored runs (Figure 5.1). Table 5.1 presents the scores and Classifiers of the best scoring pipelines for each of the datasets.

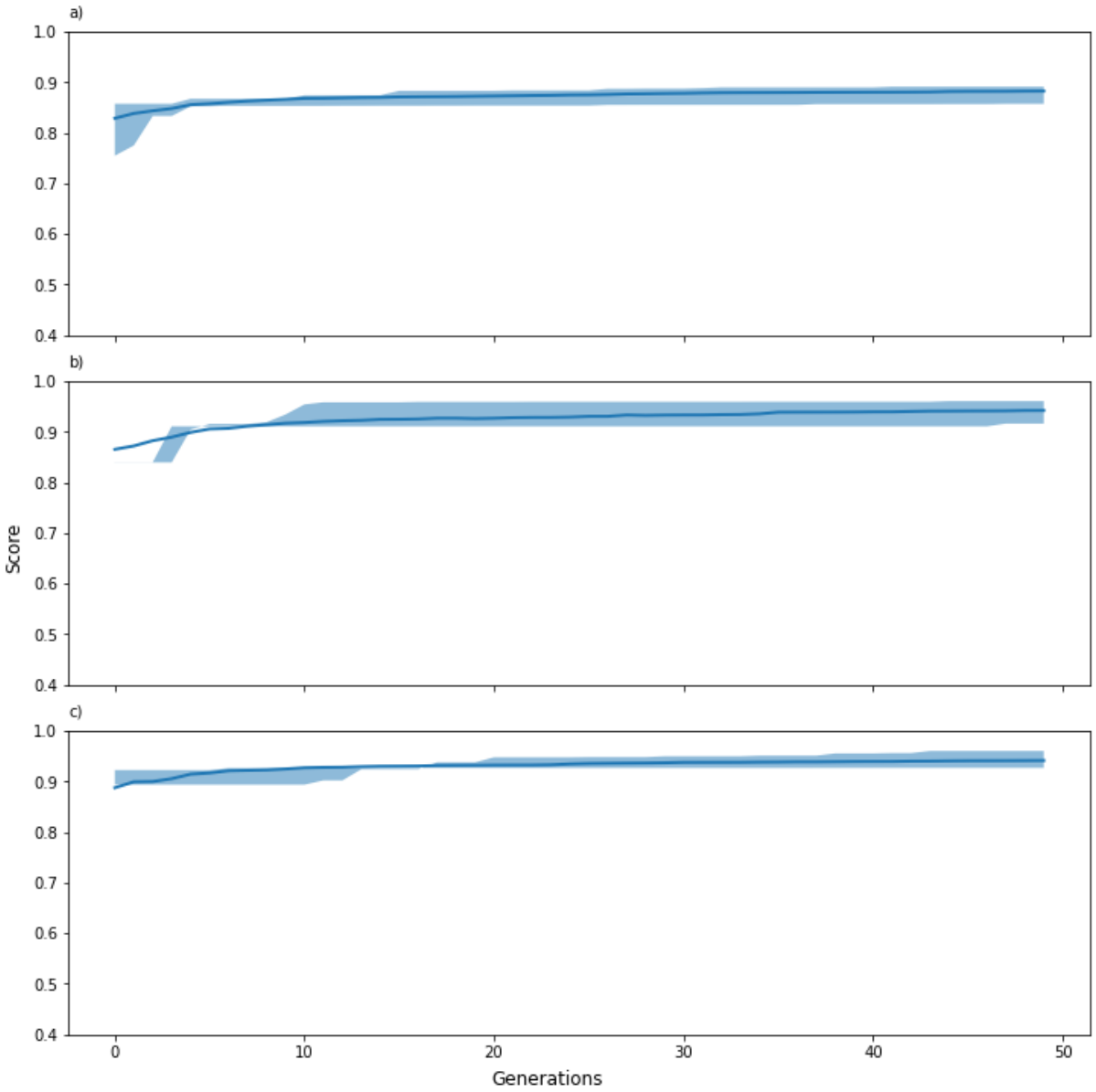


Figure 5.1 Level 0 training scores across 50 generations, the best and worst runs of the 30 runs for each dataset are indicated by the limit of the shaded area. The darker line indicates the average scores of the total runs. a) is the C40, the clustered database; b) and c) are different subsamples from the Swiss-Prot database with no clustering done.

Table 5.1 : Level 0 TPOT best training scores and classifiers with the different databases – C40 with the clustered database and Swiss 1 and Swiss 2 with random training subsamples of the complete Swiss-Prot database.

	ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	0.8948	0.8417	0.8292	0.8546	KNeighborsClassifier
SWISS 1	0.9666	0.9636	0.9567	0.9707	KNeighborsClassifier
SWISS 2	0.9643	0.9613	0.9541	0.9686	KNeighborsClassifier

Since both Swiss 1 and Swiss 2 best pipelines had very similar scores and configurations, the best overall pipeline according to the F1 score was chosen for the non-clustered EC prediction model.

5.1.1.2 Level 1 ï Class prediction

As the first Level related directly with the EC number, this level's pipelines had to solve a multiclass problem, with 7 possible classes the given sequences could be from. Whereas in the previous Level the best and worse pipelines from the 30 runs were distinct from the two models', in this Level the pipelines of the C40 model are in between the best and worst of the Swiss model (Figure 5.2). The final chosen pipelines had validation scores near 1 for the Swiss dataset, with none of the scores in the C40 dataset surpassing 0.95. (Table 5.2). Both have complex ensembles, with different Classifiers, contrasting with the previous case where the classifier was the same and the pipelines were similar. Average wise, the Swiss TPOT runs struggled to reach scores over 0.75.

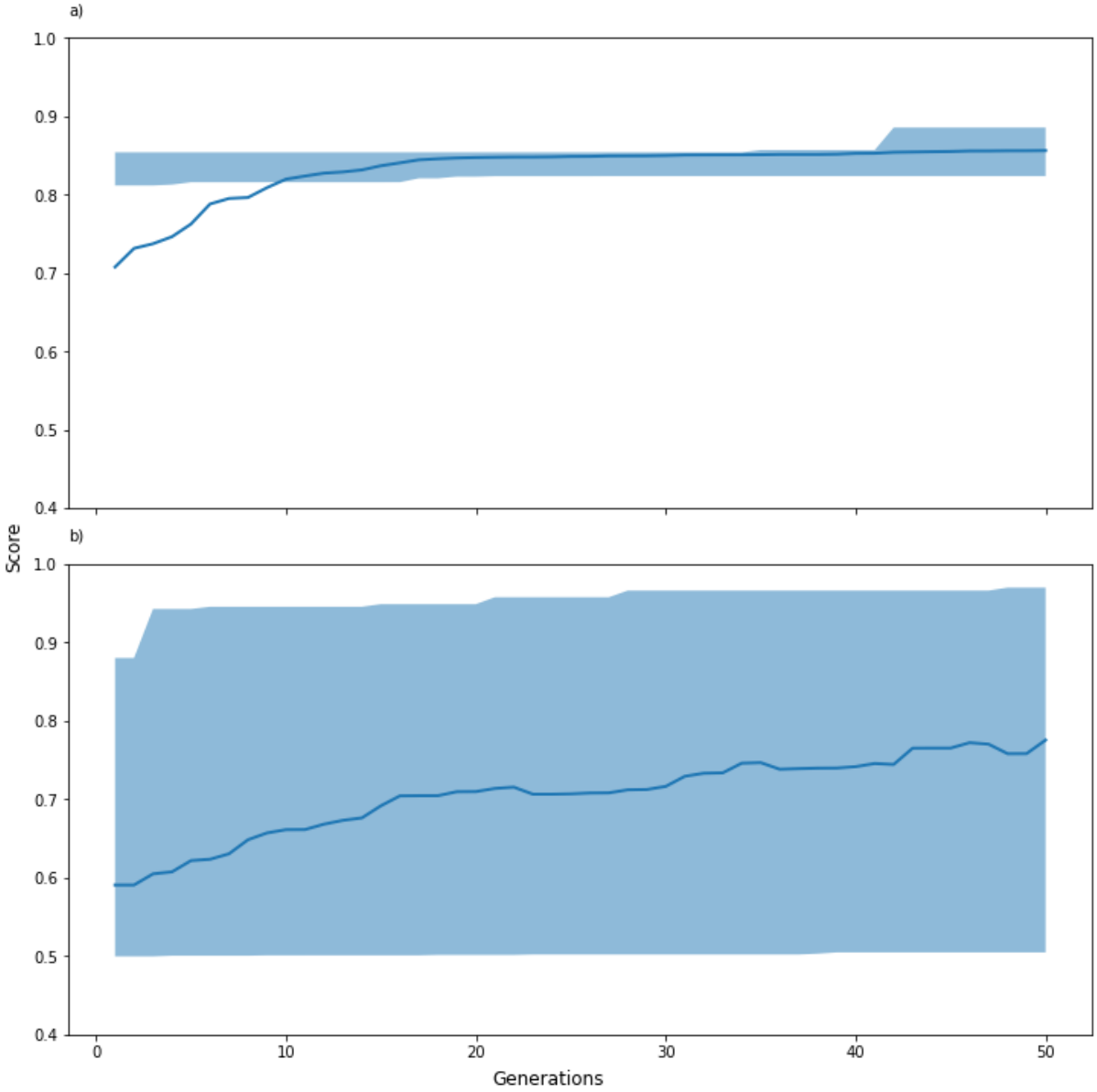


Figure 5.2 Level 1 training scores across 50 generations, the best and worst runs out of the 30 runs done for each dataset are indicated by the upper and lower limit of the shaded area. The darker line is the average of the total runs. a) C40 model using the clustered dataset and b) Swiss model with the non-clustered sequences Swiss-Prot dataset. Note that not all runs of the Swiss arrived at the established 50 generations in 3 days.

Table 5.2 Level 1 best scores and associated classifier for both models.

	ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	0.9482	0.9147	0.9226	0.9082	MLPClassifier
SWISS	0.9856	0.9769	0.9784	0.9754	KNeighborsClassifier

When observing the per class scores, these differ slightly, with the Swiss model pipeline obtaining higher scores than the C40 model's pipeline. Of note for the C40 pipeline is Class 4 and 5 lower F1 scores (Table 5.3). It should also be considered that the validation sets used for this scoring are composed of randomly chosen sequences, with each class having different amount of sequences.

Table 5.3 Level 1 Classes specific scoring from the test datasets of the respective models: C40 the clustered dataset and Swiss the non-clustered dataset.

		ACCURACY	**F1**	**PRECISION**	**RECALL**
C40	Class 1	0.98	0.93	0.94	0.92
	Class 2	0.96	0.95	0.94	0.95
	Class 3	0.97	0.95	0.94	0.95
	Class 4	0.98	0.8	0.8	0.8
	Class 5	0.98	0.84	0.89	0.8
	Class 6	0.99	0.92	0.95	0.95
	Class 7	0.99	0.93	0.91	0.95
SWISS	Class 1	0.98	0.97	0.97	0.97
	Class 2	0.98	0.97	0.97	0.98
	Class 3	0.98	0.97	0.98	0.96
	Class 4	0.99	0.96	0.97	0.96
	Class 5	0.99	0.96	0.97	0.95
	Class 6	0.99	0.97	0.96	0.99
	Class 7	0.99	0.99	0.99	0.98

5.1.1.3 Level 2 ï Subclass prediction

For this level, each class had to have its own pipeline in order to predict the associated subclass. As such, each model has 7 pipelines to consider at this level, and which pipeline should be used is decided by the prediction of the Level 1 pipeline. Once again, the training scores of the Swiss model surpass, in general, the C40 model (Figure 5.3 and Figure 5.4). In both cases, Class 3 has the lowest scores in the training runs and most of the validation scores, especially noticeable on the C40 model, where test scores for all were lower or equal to 80% only (Table 5.4). Removing all subclasses with less than a total value of 50 elements for Class 3 of the C40 model did increase the test and training scores to higher than 90%, suggesting the subclasses removed had very few elements to train on making the TPOT algorithm struggle to recognize their presence while training.

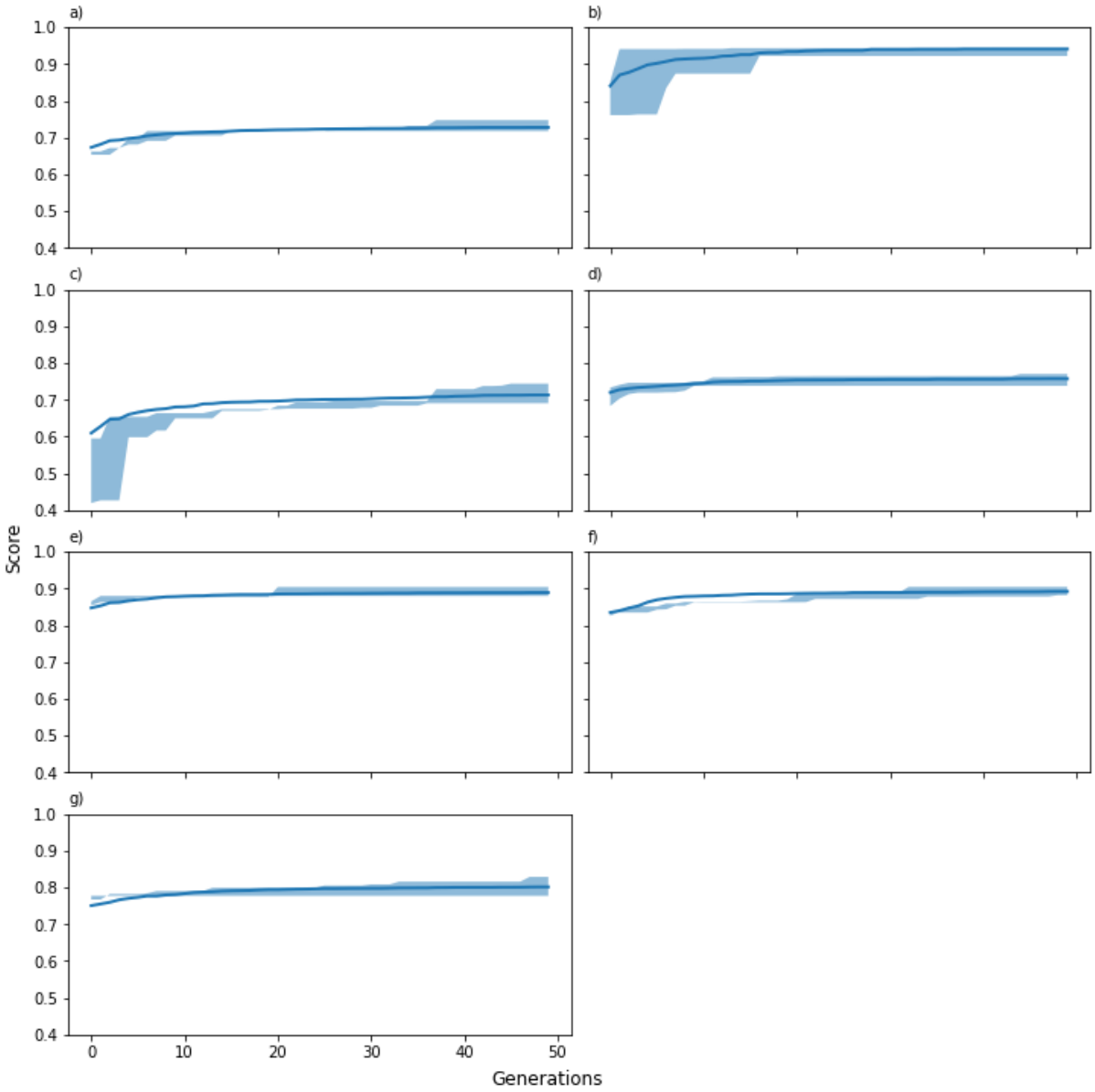


Figure 5.3 Level 2 C40 dataset training scores across 50 generations, the best and worst runs out of the 30 runs done for each dataset are indicated by the upper and lower limit of the shaded area. The darker line is the average of the total runs. a) Class 1, b) Class 2, c) Class 3, d) Class 4, e) Class 5, f) Class 6, g) Class 7.

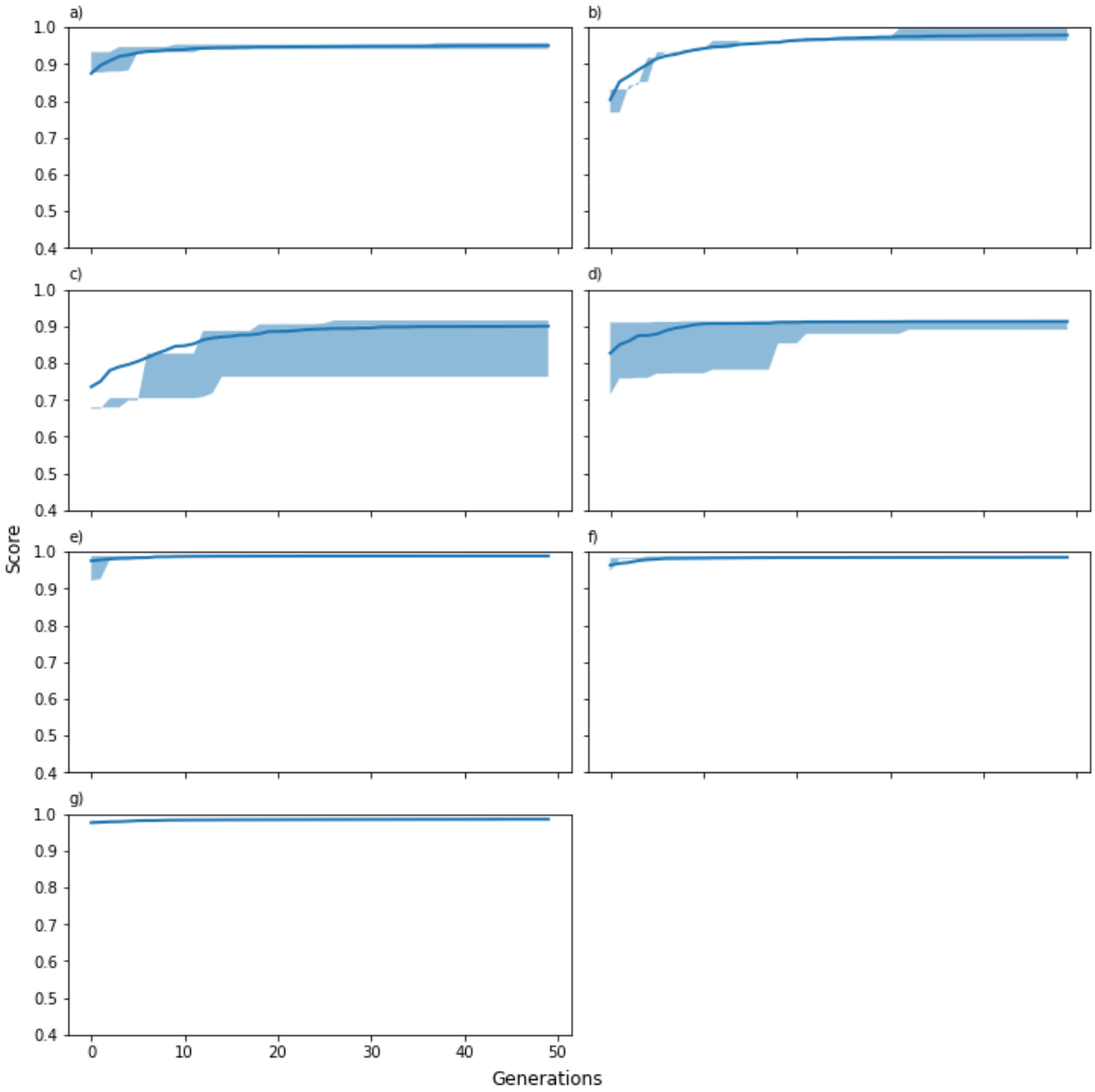


Figure 5.4 Level 2 Swiss dataset training scores across 50 generations, the best and worst runs out of the 30 runs done for each dataset are indicated by the upper and lower limit of the shaded area. The darker line is the average of the total runs. a) Class 1, b) Class 2, c) Class 3, d) Class 4, e) Class 5, f) Class 6, g) Class 7.

Table 5.4 Level 2 Best scores and associated classifier of both C40 and Swiss models.

		ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	Class 1	0.8772	0.7434	0.7613	0.7629	LinearSVC
	Class 2	0.9805	0.9706	0.9765	0.9651	LinearSVC
	Class 3	0.8029	0.6577	0.7432	0.6140	MLPClassifier
	Class 4	0.8665	0.7587	0.7706	0.7551	LinearSVC
	Class 5	0.9802	0.9670	0.9679	0.9665	MLPClassifier
	Class 6	0.9364	0.9050	0.9740	0.8793	LinearSVC
	Class 7	0.9343	0.8812	0.8857	0.8833	LinearSVC
	Class 1	0.9777	0.9570	0.9682	0.9682	LogisticRegression

SWISS	Class 2	0.9981	0.9973	0.9979	0.9967	LinearSVC
	Class 3	0.9826	0.9673	0.9682	0.9669	LogisticRegression
	Class 4	0.9881	0.9823	0.9872	0.9779	MLPClassifier
	Class 5	0.9958	0.9939	0.9954	0.9924	LinearSVC
	Class 6	0.9899	0.9890	0.9990	0.9801	LinearSVC
	Class 7	0.9959	0.9934	0.9941	0.9928	LinearSVC

5.1.1.4 Level 3 ï Sub-subclasses prediction

The Swiss database had a total of 74 subclasses and 263 sub-subclasses. Removal of the subclasses and sub-subclasses with not enough training data reduced them to 61 and 240 respectively. In the C40 database, due to clustering, there were less sub-subclasses present in the data, with a total of 74 subclasses and 258 sub-subclasses. However, 31 subclasses and a total of 93 sub-subclasses were removed due to insufficient data, leading to the final number of 43 subclasses trained and 165 sub-subclasses possible to be predicted.

The validation scores difference within the subclasses for each model vary between classes but, with few exceptions, they tend to be 90% and higher. The Classifiers between the same subclasses in both models do not tend to be the same at this level (e.g. Class 6 Table 5.5, other classes scores from Appendix B 1 to Appendix B 6).

Table 5.5 Class 6 trained subclasses for sub-subclasses predictions validation scores for both the C40 model and the Swiss model.

	SUBCLASSES	ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	Sub 1	1.0	1.0	1.0	1.0	SGDClassifier
	Sub 2	0.9741	0.9508	0.9062	1.0	DecisionTree Classifier
	Sub 3	0.9460	0.9281	0.9556	0.9057	LinearSVC
	Sub 5	0.9903	0.9830	0.9666	1.0	LinearSVC
SWISS	Sub 1	1.0	1.0	1.0	1.0	LinearSVC
	Sub 2	0.9963	0.9973	0.9983	0.9963	MLPClassifier
	Sub 3	0.9987	0.9985	0.9992	0.9978	LinearSVC
	Sub 4	1.0	1.0	1.0	1.0	GradientBoosting Classifier
	Sub 5	1.0	1.0	1.0	1.0	SGDClassifier

The majority of subclasses trained did not have a wide difference between the best and worst runs (Figure 5.5 and Figure 5.6).

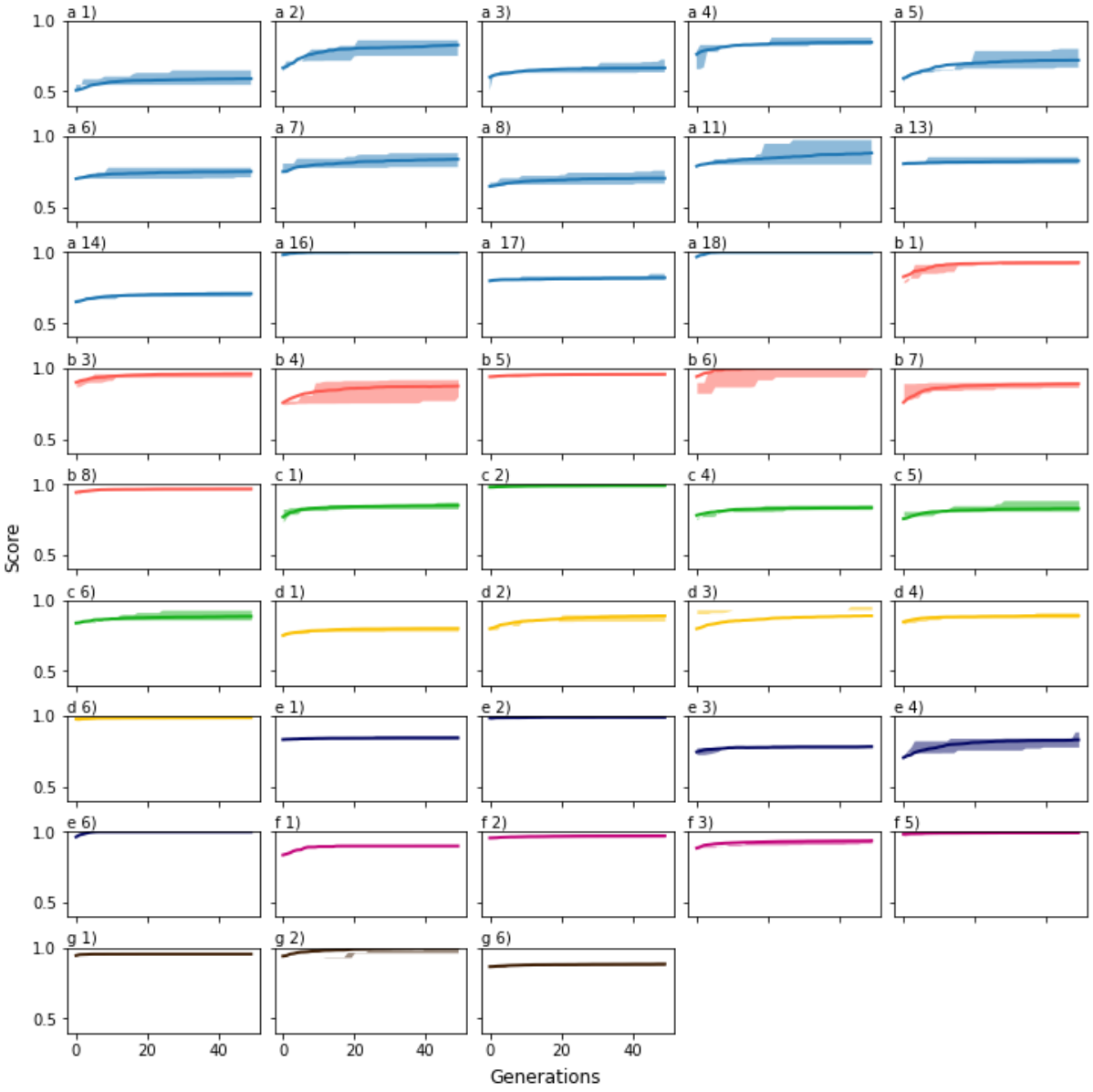
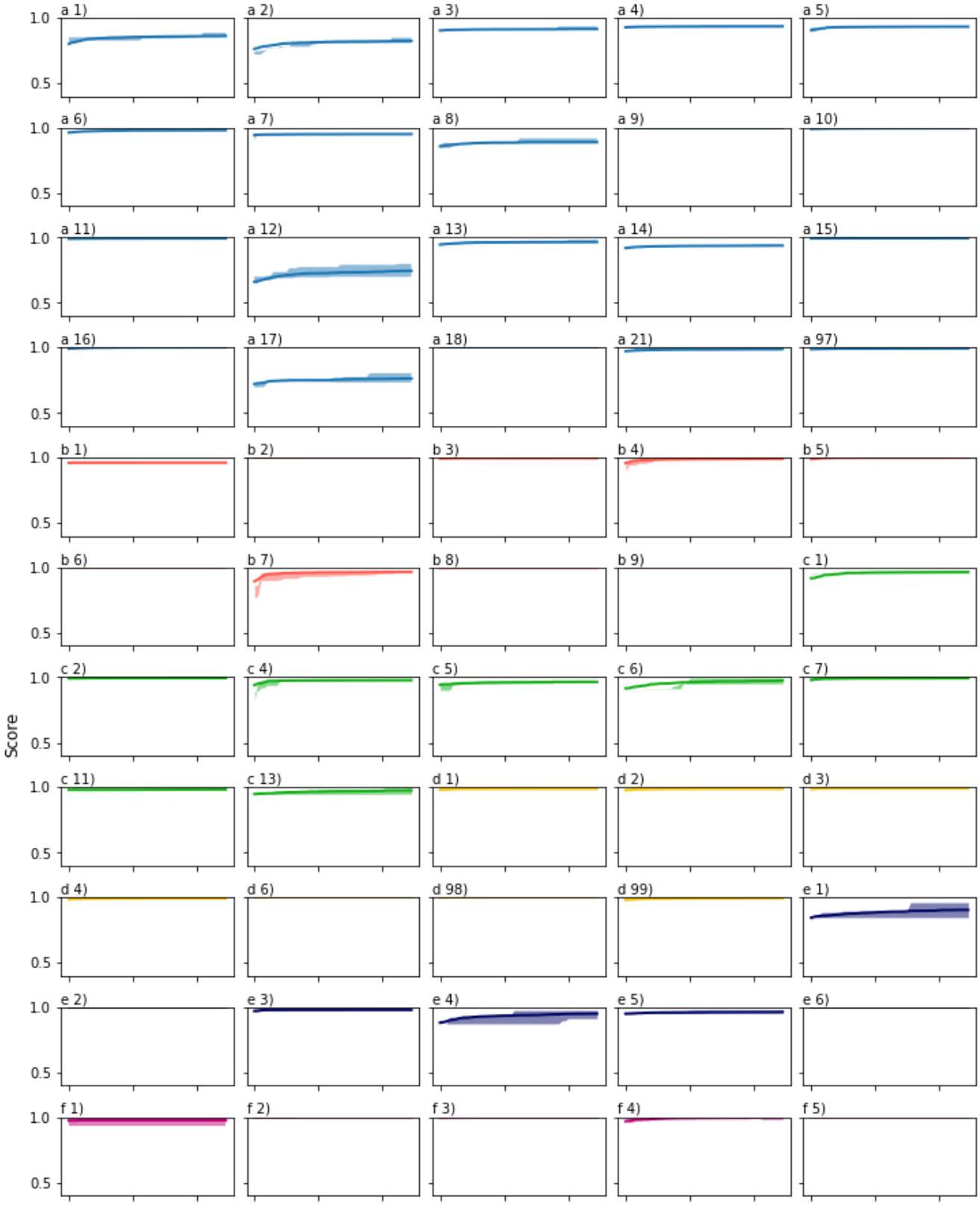


Figure 5.5 Level 3 of C40 dataset, the best and worst runs out of the 30 runs done for each dataset are indicated by the upper and lower limit of the shaded area. The darker line is the average of the total runs. The lowercase letters indicate the class (as well as color) and the number associated is the subclass the subplot belongs to. a – Class 1; b – Class 2 c – Class 3 d – Class 4; e – Class 5; f – Class 6 and g – Class 7.


Score
1.0
0.5
a 1)
a 2)
a 3)
a 4)
a 5)
a 6)
a 7)
a 8)
a 9)
a 10)
a 11)
a 12)
a 13)
a 14)
a 15)
a 16)
a 17)
a 18)
a 21)
a 97)
b 1)
b 2)
b 3)
b 4)
b 5)
b 6)
b 7)
b 8)
b 9)
c 1)
c 2)
c 4)
c 5)
c 6)
c 7)
c 11)
c 13)
d 1)
d 2)
d 3)
d 4)
d 6)
d 98)
d 99)
e 1)
e 2)
e 3)
e 4)
e 5)
e 6)
f 1)
f 2)
f 3)
f 4)
f 5)

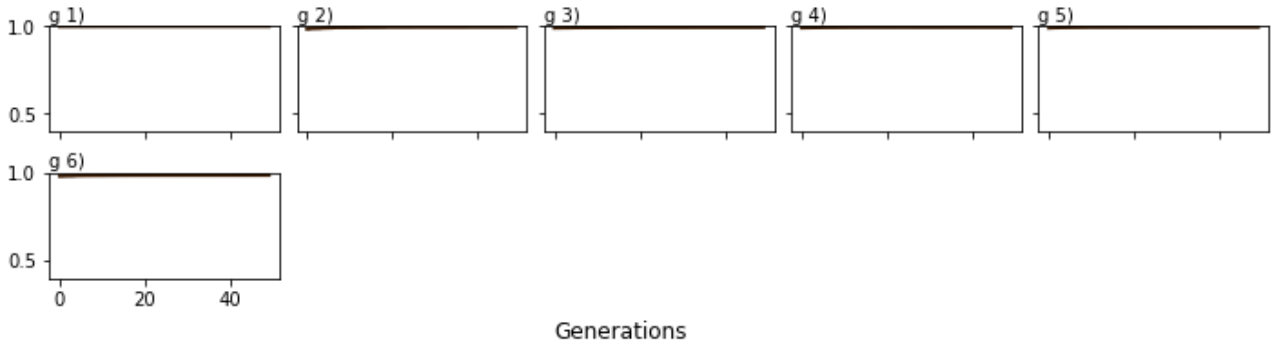


Figure 5.6 Level 3 of Swiss dataset, the best and worst runs out of the 30 runs done for each dataset are indicated by the upper and lower limit of the shaded area. The darker line is the average of the total runs. The lowercase letters indicate the class (as well as color) and the number associated is the subclass the subplot belongs to. a – Class 1; b – Class 2 c – Class 3 d – Class 4; e – Class 5; f – Class 6 and g – Class 7.

5.2 Validation scores

TPOT successfully arrived at pipelines with a high scoring in both training and pipeline validation for both models. With the individual scores of the different levels and classes, the models created should have a performance comparable to published results. For validating this claim however, the validation dataset was used and the models compared according to the results given.

Using only sequences from the Validation dataset, which only has sequences created from January 1st to May 18th of 2023 so never used in any of the models, the flat scores for the models created by TPOT and ECPred, DeepEC, HECNet and Benz-WS were calculated in order to compare among these. Table 5.6 to Table 5.9 demonstrate each level's scoring for each of the models in question. Benz-WS is the model with more consistent high scores, with the exception of Level 0, in which HECNet surpasses it in Accuracy and F1. The C40 model scored close to Benz-WS across the levels (assuming F1 scoring), with the exception of the Level 3 where Benz-Ws scored far above the rest. The Swiss model did have slightly better scores than C40 at level 3. DeepEC, ECPred and HECNet had a drastic fall in the scores of Level 2 and Level 3.

Table 5.6 Level 0 scores of the 2 created models and 4 other models with the Validation dataset.

	ACCURACY	F1	PRECISION	RECALL
C40	0.84	0.75	0.91	0.63
SWISS	0.84	0.76	0.88	0.67
ECPRED	0.83	0.80	0.71	0.91
DEEPEC	0.72	0.67	0.58	0.79
BENZ-WS	0.86	0.78	0.89	0.69
HECNET	0.89	0.85	0.91	0.80

Table 5.7 Level 1 scores of the created models and the 4 published models with the Validation dataset..

	ACCURACY	F1	PRECISION	RECALL
C40	0.88	0.85	0.82	0.91
SWISS	0.81	0.77	0.76	0.81

ECPRED	0.77	0.76	0.81	0.80
DEEPEC	0.71	0.62	0.59	0.67
BENZ-WS	0.94	0.80	0.78	0.83
HECNET	0.91	0.78	0.79	0.79

Table 5.8 Level 2 scores of the created models and the 4 published models with the Validation dataset.

	ACCURACY	**F1**	**PRECISION**	**RECALL**
C40	0.84	0.80	0.80	0.80
SWISS	0.80	0.76	0.79	0.77
ECPRED	0.60	0.54	0.61	0.58
DEEPEC	0.65	0.55	0.55	0.58
BENZ-WS	0.89	0.82	0.81	0.83
HECNET	0.71	0.41	0.42	0.46

Table 5.9 Level 3 scores of the created models and the 4 published models with the Validation dataset..

	ACCURACY	**F1**	**PRECISION**	**RECALL**
C40	0.76	0.53	0.55	0.54
SWISS	0.79	0.56	0.55	0.60
ECPRED	0.57	0.48	0.58	0.49
DEEPEC	0.66	0.45	0.47	0.45
BENZ-WS	0.91	0.71	0.68	0.77
HECNET	0.49	0.22	0.31	0.24

Hierarchical metrics are relevant for this situation, even though two models present were not built hierarchically (DeepEC and Benz-WS), the EC number prediction is naturally hierarchical in nature. Table 5.10 indicates the scores with the chosen hierarchical measures when considering different levels of prediction (only the first two levels, the first three or all) and offers a good overall look on the performance. From this table we can gather that Benz-WS keeps the highest score for all levels prediction but it is followed closely by C40, Swiss and surprisingly HECNet. HECNet benefits with this scoring method as the higher scores of the Levels 0 and 1 help keep the overall hierarchical score higher than with the flat metrics.

Table 5.10 Hierarchical scores of the different models with different levels considered: 0-1 only considers the first two levels; 0-1-2 considers the first three levels; 0-1-2-3 considers all levels in the models (that are being considered in this study).

	F1			**PRECISION**			**RECALL**		
LEVELS	0-1	0-1-2	0-1-2-3	0-1	0-1-2	0-1-2-3	0-1	0-1-2	0-1-2-3
C40	0.83	0.81	0.8	0.83	0.82	0.81	0.84	0.8	0.79
SWISS	0.82	0.81	0.79	0.82	0.81	0.81	0.83	0.8	0.78
ECPRED	0.77	0.73	0.72	0.77	0.72	0.71	0.77	0.75	0.74

DEEPEC	0.68	0.62	0.6	0.6	0.54	0.53	0.78	0.73	0.71
BENZ-WS	0.85	0.83	0.83	0.84	0.84	0.84	0.85	0.82	0.81
HECNET	0.85	0.82	0.78	0.84	0.83	0.79	0.87	0.82	0.78

6 Discussion

The construction of a machine learning model involves several steps, from the data to be used, treatment of said data for the relevant features, the choice of classifiers and how said model scores, especially compared to existent models. The middle steps are time consuming, typically including either trial and error to work out or needing expertise in machine learning. TPOT solves these steps by returning optimized pipelines, which treat the data and choose the best classifier for the job, using Genetic Programming in order to arrive at an optimized pipeline (Olson et al., 2016).

This thesis' purpose was to evaluate whether TPOT could be used to assist in the construction of an EC number prediction model that can compare to previously published models. Two datasets were considered in order to verify whether the clustering (as recommended to reduce redundancy and any possible 'data leakage' (Hou et al., 2022)) would affect the TPOT's optimal pipelines as compared to the total dataset.

It was observed that the computational requirements and time needed increased drastically with the dataset size. A subsample of the Swiss database had to be used, while the Cross-validation times for the Level 0 had to be lowered so the runs could be finished within the stipulated 3 days' maximum. The computational requirements also increased for multiclass problems and this is especially observed in Level 1's training scores of the Swiss model, as despite the lower number of sequences than Level 0, TPOT struggled to reach better scoring pipelines, with the average of the 30 runs stopping at a score of less than 0.8.

Of the two created models, training and pipeline validation scores generally favored the Swiss model over the C40 model, something that was not observed with the validation dataset level by level scores considered for this thesis. This may be due to the redundancy bias provoked by the appearance of homologues as, despite taking out those that were identical sequences, the similarity between sequences present within the Swiss training and validation sets possibly 'tricked' the models into better scores than in truly had.

The pipelines from both models varied between each other. Level 0's optimized pipelines have similarities to each other, with the use of the KNeighborsClassifier and a data pre-processing normalization algorithm (Appendix C 1) however C40's pipeline has the added complexity of using another classifier (Logistic Regression) for an initial classification which is then fed into the normalizer algorithm. This higher complexity of the C40 pipeline is mirrored in Level 1, with the pipeline including several more algorithms for data preparation than the Swiss pipeline (Appendix C 2). As for the Level 2 and Level 3, pipelines vary in complexity between the Classes pipelines and within them, with Subclasses pipelines ranging from those with a single Classifier to complex pipelines that use several Classifiers and data processing algorithms. This range is possibly due to the different amount of sequences available to train at each subclass and the amount of sub-subclasses. Some of the Level 3 TPOT runs are indistinguishable from the average or are completely flat, with barely any score growth

over the runs. For the most part, this happened when the score was already high (over 90%) but there are others where TPOT still struggled for an optimized pipeline keeping a low score.

The Level 1 pipelines for both models discern between the different classes with high pipeline validation scores. A closer look of C40's pipeline validation scores for each individual class shows that that discernment is lower in two classes, Class 4 and Class 5, with pipeline validation scores that do not reach 0.9. Though protein embeddings present a variety of possible biological information (evolutionary, similarity etc.), the per-protein embeddings (average) used for this project is limited to full protein information, with no individual properties from the residues it is composed of (Dallago et al., 2021). For Class 5 (Isomerases), the lower score could be due to the fact that the enzymes belonging to this class have a lower sequence similarity between each other than in other classes, despite having the same function (Higdon et al., 2010). Along with this, they are also mechanistically diverse in their reactions (Nath & Mitchell, 2012). Lyases, as the enzymes from Class 4 are labeled, could have biological reasons that resulted in this lower scoring. A reason that could be found at the writing of this dissertation was in Nath & Mitchell (2012), where these enzymes were described as harder to predict when considering their chemical signature. This, however, does not account for this score drop as transferases (Class 2) suffer through the same difficulty.

The best pipeline for the Swiss model at Level 0 would have been different, and perhaps scored better in the Validation dataset, had the data been subsampled differently, with a higher amount of sequences in the dataset than the established 200 thousand. The Swiss dataset created was closer in sequence quantity to the C40 dataset than its original amount and, considering the validation scores and the pipelines, the subsampled Swiss dataset does not seem to add more knowledge than the C40 dataset, despite what the training and pipeline validation scores seem to indicate. The change of the Cross-Validation parameter in TPOT to the default 5 could also have been helpful, however the probable bias in the training and pipeline validation scores would probably be kept. This increase of sequences (or Cross-Validation folds) would forcefully also increase the computational effort and the time needed for a complete run which is already on the verge of surpassing the 3 days. Another strategy that could have been used for this level and would remove some of the 'data leakage' problem would be by creating a dataset with a different level of maximum similarity (instead of the 40% maximum similarity using 50% or 60% or perhaps 90%). This would increase the information available for training by adding more relevant sequences, and if it would be subsampled, these sampled sequences would probably not be similar to each other, allowing the training to be more unbiased than what happened to the created Swiss pipeline. This could also be beneficial to the C40 model as, for this level, this clustering level is too restrictive, removing information that TPOT could benefit in order to arrive at a higher scoring pipeline.

When considering the published models to compare with the created models in terms of performance, four published models were chosen, which were also used for recent quality comparison with other models and/or cited in reviews (e.g. Ferdous et al., 2022). While several other models exist with the same conditions, these were older versions of the ones used for comparison (e.g. EzyPred (Shen & Chou, 2007)). In the case of DeePre (Y. Li et al., 2018) we found problems in using the

webserver, although a recently new published model used it for score comparison (Lin et al., 2022), and technical support did not answer back in time for the writing of this thesis. As such, the four chosen models were two downloadable models, ECPred and DeepEC (Ryu et al., 2019), and two web-server based, HECNet (Memon et al., 2020) and Benz-WS (Baldazzi et al., 2021). Of note is that Benz-WS utilizes Hidden Markov Models (HMM) and Pfam models, and uses similarity to reference sequences from HMM clusters for prediction (Baldazzi et al., 2021). Of the other three models, ECPred (Dalkiran et al., 2018) also uses sequence similarity (blast-kNN) in its structure, using a weighted average in order to choose the prediction. The other two, DeepEC and HECNet, much like the created models for this thesis, do not use any direct similarity algorithms in the prediction of the EC number. Instead, HECNet uses an ensemble of encodings, including Functional Domains information, and DeepEC uses an embedding created by a layer within its architecture that receives as input the result from a one-hot encoding method.

Overall, between all the models, using the flat F1 score as the comparison metric, the Benz-WS was more consistent with its high scores, with C40 model a close second, something also reflected in the hierarchical scores. It should be emphasized however that Benz-WS and DeepEC were not constructed in a hierarchical matter, which this scoring method assumes. As such, only considering the hierarchical models for the hierarchical measures, both the C40 and Swiss models scored best. This shows that using TPOT for a ML model creation with biological data can be advantageous over the manual choice of machine learning methods, as both created models present pipelines which return scores comparable with other models, not needing as much expertise or time consuming testing to arrive at optimized pipelines.

Changing of TPOT parameters such as mutation rate and population size according to the class and data characteristics, as well as a different data clustering might help with improving the models. This is especially true for Level 0's pipelines. A bigger Validation dataset, especially including more full EC number sequences, would have also been better when validating the scores. This is particularly true when considering HECNet's 1000 aminoacids limitation, compared with the 5000 limitation of the rest of the models, which may be part of the reason for this model's unexpected low scores on the flat metrics, as these do not consider the high scores on the previous levels.

7 Conclusion and Future remarks

With the wide variety of Machine Learning algorithms freely available nowadays, the choice of how to combine them into a working, and optimized, pipeline for the task can be daunting for researchers. AutoML tools like TPOT are useful in obtaining these optimized pipelines, removing steps that typically need either an expert or time for experimenting different combinations.

In this thesis, TPOT was used to create the pipelines needed for the construction of two EC number prediction models (C40 and Swiss), each with four levels (Level 0 to 3), that were then evaluated against four previously published models (ECPred, DeepEC, HECNet and Benz-WS.) The created model datasets differed only in that the C40's dataset was a clustered version of Swiss' dataset. The sequences in the datasets were represented as the average (per-protein) embeddings from the ProtBert and ProtT5 models, with no added features beyond those.

The resulting models' pipelines differed from one model to another, the TPOT adapting to the data available, even if it was only a clustered dataset of another. The Swiss dataset's first 3 Levels pipelines boasted higher training and pipeline validation scores than the C40 pipelines but when considering the Validation dataset scores', these scores drop, possibly the effect of data leakage.

The multiclass pipelines were not always able to distinguish between the different classes with the same success. In the Level 1 pipeline of the C40 dataset, it was revealed that TPOT struggled to get a pipeline that could predict the same in all classes, with Class 4 and 5's F1 scores not reaching 0.9, where the rest of the models surpasses it. There may be biological reasons for this difficulty that the embeddings have encoded.

The created models' scores with the validation datasets matched or outperformed the ECPred, DeepEC and HECNet from Level 1 onwards when considering the flat metrics typically used. Benz-WS, which uses homology based methods, did have a noteworthy higher score on Level 3, though the scores from the previous two levels weren't much different from C40's own. When considering hierarchical specific metrics, Benz-WS continued to dominate but C40 and Swiss scores for the full 4 levels were, respectively, second and third best. These scores when comparing with the rest of the models show that TPOT can be a useful tool in the creation of biological models' pipelines.

Future works on protein based predictions could benefit from using TPOT, especially if there is not a high volume of data, as the resulting pipeline would not only be adapted specifically for the data available but also explicative in how the data is handled in training the model. Also, a TPOT adaptation specific for protein studies as with the Genome Wide Studies specialized configuration TPOT-MDR could be beneficial for further studies with protein sequences.

8 References

Ahvanooey, M. T., Li, Q., Wu, M., & Wang, S. (2019). A survey of genetic programming and its applications. *KSII Transactions on Internet and Information Systems*, *13*(4), 1765–1794. https://doi.org/10.3837/tiis.2019.04.002

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, *215*(3), 403–410. https://doi.org/10.1016/S0022-2836(05)80360-2

Amidi, A., Amidi, S., Vlachakis, D., Megalooikonomou, V., Paragios, N., & Zacharaki, E. I. (2018). EnzyNet: Enzyme classification using 3D convolutional neural networks on spatial representation. *PeerJ*, *2018*(5), 1–18. https://doi.org/10.7717/peerj.4750

Balaji, S. (2021). The transferred translocases: An old wine in a new bottle. *Biotechnology and Applied Biochemistry*, *69*(4), 1587–1610. https://doi.org/10.1002/bab.2230

Baldazzi, D., Savojardo, C., Martelli, P. L., & Casadio, R. (2021). BENZ WS: The Bologna ENZyme Web Server for four-level EC number annotation. *Nucleic Acids Research*, *49*(Web Server Issue), W60–W66. https://doi.org/10.1093/nar/gkab328

Bateman, A., Martin, M.-J., Orchard, S., Magrane, M., Ahmad, S., Alpi, E., Bowler-Barnett, E. H., Britto, R., Bye-A-Jee, H., Cukura, A., Denny, P., Dogan, T., Ebenezer, T., Fan, J., Garmiri, P., da Costa Gonzales, L. J., Hatton-Ellis, E., Hussein, A., Ignatchenko, A., … Zhang, J. (2023). UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, *51*(D1), D523–D531. https://doi.org/10.1093/nar/gkac1052

Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.-C., Estreicher, A., Gasteiger, E., Martin, M. J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S., & Schneider, M. (2003). The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, *31*(1), 365–370. https://doi.org/10.1093/nar/gkg095

Bonetta, R., & Valentino, G. (2020). Machine learning techniques for protein function prediction. *Proteins: Structure, Function and Bioinformatics*, *88*(3), 397–413. https://doi.org/10.1002/prot.25832

Borro, L. C., Oliveira, S. R. M., Yamagishi, M. E. B., Mancini, A. L., Jardine, J. G., Mazoni, I., Santos, E. H. dos, Higa, R. H., Kuser, P. R., & Neshich, G. (2006). Predicting enzyme class from protein structure using Bayesian classification. *Genetics and Molecular Research : GMR*, *5*(1), 193–202. http://www.ncbi.nlm.nih.gov/pubmed/16755510

Brandes, N., Ofer, D., Peleg, Y., Rappoport, N., & Linial, M. (2022). ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics*, *38*(8), 2102–2110. https://doi.org/10.1093/bioinformatics/btac020

Bum, J. L., Jong, Y. L., Heon, G. L., & Keun, H. R. (2007). Classification of enzyme function from protein sequence based on feature representation. *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering, BIBE*, 741–747. https://doi.org/10.1109/BIBE.2007.4375643

Cai, C. Z. (2003). SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic Acids Research*, *31*(13), 3692–3697. https://doi.org/10.1093/nar/gkg600

Che, Y., Ju, Y., Xuan, P., Long, R., & Xing, F. (2016). Identification of multi-functional Enzyme with multi-label classifier. *PLoS ONE*, *11*(4), 1–13. https://doi.org/10.1371/journal.pone.0153503

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & De Hoon, M. J. L. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

Cui, F., Zhang, Z., & Zou, Q. (2021). Sequence representation approaches for sequence-based protein prediction tasks that use deep learning. *Briefings in Functional Genomics*, *20*(1), 61–73. https://doi.org/10.1093/bfgp/elaa030

Dalkiran, A., Rifaioglu, A. S., Martin, M. J., Cetin-Atalay, R., Atalay, V., & Doğan, T. (2018). ECPred: A tool for the prediction of the enzymatic functions of protein sequences based on the EC nomenclature. *BMC Bioinformatics*, *19*(1), 1–13. https://doi.org/10.1186/s12859-018-2368-y

Dallago, C., Schütze, K., Heinzinger, M., Olenyi, T., Littmann, M., Lu, A. X., Yang, K. K., Min, S., Yoon, S., Morton, J. T., & Rost, B. (2021). Learned Embeddings from Deep Learning to Visualize and Predict Protein Sets. *Current Protocols*, *1*, 26. https://doi.org/10.1002/cpz1.113

des Jardins, M., Karp, P. D., Krummenacker, M., Lee, T. J., & Ouzounis, C. A. (1997). Prediction of enzyme classification from protein sequence without the use of sequence similarity. *Proceedings. International Conference on Intelligent Systems for Molecular Biology*, *5*, 92–99. http://www.ncbi.nlm.nih.gov/pubmed/9322021

Ding, W., Nakai, K., & Gong, H. (2022). Protein design via deep learning. *Briefings in Bioinformatics*, *23*(3), 1–16. https://doi.org/10.1093/bib/bbac102

Dou, W., Liu, Y., Liu, Z., Yerezhepov, D., Kozhamkulov, U., Akilzhanova, A., Dib, O., & Chan, C.-K. (2021). An AutoML Approach for Predicting Risk of Progression to Active Tuberculosis based on Its Association with Host Genetic Variations. *2021 10th International Conference on Bioinformatics and Biomedical Science*, 82–88. https://doi.org/10.1145/3498731.3498743

Eiben, A E, & Smit, S. K. (2012). Evolutionary Algorithm Parameters and Methods to Tune Them. In *Autonomous search* (pp. 15–36). Springer.

Eiben, Agoston E., & Smith, J. (2015). From evolutionary computation to the evolution of things. *Nature*, *521*(7553), 476–482. https://doi.org/10.1038/nature14544

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., & Rost, B. (2021). ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(8), 29. https://doi.org/10.1109/TPAMI.2021.3095381

Ethayarajh, K. (2019). How contextual are contextualized word representations? Comparing the geometry of BERT, ELMO, and GPT-2 embeddings. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 55–65. https://doi.org/10.18653/v1/d19-1006

Ferdous, S., Shihab, I. F., & Reuel, N. F. (2022). Effects of sequence features on machine-learned enzyme classification fidelity. *Biochemical Engineering Journal*, *187*, 108612. https://doi.org/10.1016/j.bej.2022.108612

Fu, L., Niu, B., Zhu, Z., Wu, S., & Li, W. (2012). CD-HIT: Accelerated for clustering the next-generation sequencing data. *Bioinformatics*, *28*(23), 3150–3152. https://doi.org/10.1093/bioinformatics/bts565

Gbenro, S., Hippe, K., & Cao, R. (2020). *HMMeta*. 1–6. https://doi.org/10.1145/3388440.3414702

Gillioz, A., Casas, J., Mugellini, E., & Khaled, O. A. (2020). Overview of the Transformer-based Models for NLP Tasks. *Proceedings of the Federated Conference on Computer Science and Information Systems*, 179–183. https://doi.org/10.15439/2020F20

Guo, H.-B., Perminov, A., Bekele, S., Kedziora, G., Farajollahi, S., Varaljay, V., Hinkle, K., Molinero, V., Meister, K., Hung, C., Dennis, P., Kelley-Loughnane, N., & Berry, R. (2022). AlphaFold2 models indicate that protein sequence determines both structure and dynamics. *Scientific Reports*, *12*(1), 10696. https://doi.org/10.1038/s41598-022-14382-9

Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Tin Kam Ho, Macia, N., Ray, B., Saeed, M., Statnikov, A., & Viegas, E. (2015). Design of the 2015 ChaLearn AutoML challenge. *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

https://doi.org/10.1109/IJCNN.2015.7280767

Heinzinger, M., Elnaggar, A., Wang, Y., Dallago, C., Nechaev, D., Matthes, F., & Rost, B. (2019). Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics*, *20*(1), 723. https://doi.org/10.1186/s12859-019-3220-8

Higdon, R., Louie, B., & Kolker, E. (2010). Modeling sequence and function similarity between proteins for protein functional annotation. *HPDC 2010 - Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 499–502. https://doi.org/10.1145/1851476.1851548

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261–266. https://doi.org/10.1126/science.aaa8685

Hou, Q., Waury, K., Gogishvili, D., & Feenstra, K. A. (2022). Ten quick tips for sequence-based prediction of protein properties using machine learning. *PLoS Computational Biology*, *18*(12), 4–9. https://doi.org/10.1371/journal.pcbi.1010669

Iqbal, M. J., Javed, Z., Sadia, H., Qureshi, I. A., Irshad, A., Ahmed, R., Malik, K., Raza, S., Abbas, A., Pezzani, R., & Sharifi-Rad, J. (2021). Clinical applications of artificial intelligence and machine learning in cancer diagnosis: looking into the future. *Cancer Cell International*, *21*(1), 1–11. https://doi.org/10.1186/s12935-021-01981-1

Jayaram, B. (2008). Decoding the Design Principles of Amino Acids and the Chemical Logic of Protein Sequences. *Nature Precedings*. https://doi.org/10.1038/npre.2008.2135.1

Jing, X., Dong, Q., Hong, D., & Lu, R. (2020). Amino Acid Encoding Methods for Protein Sequences: A Comprehensive Review and Assessment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *17*(6), 1918–1931. https://doi.org/10.1109/TCBB.2019.2911677

Khan, K. A., Memon, S. A., & Naveed, H. (2021). A hierarchical deep learning based approach for multi-functional enzyme classification. *Protein Science*, *30*(9), 1935–1945. https://doi.org/10.1002/pro.4146

Kiritchenko, S., Matwin, S., Nock, R., & Famili, A. F. (2006). Learning and eval- uation in the presence of class hierarchies: application to text categorization. In *Proceed- ings of the Conference of the Canadian Society for Computational Studies of Intelligence* (pp. 395–406). Springer.

Koranne, S. (2011). Hierarchical data format 5: HDF5. In *Handbook of Open Source Tools* (pp. 191–200). Springer.

Kosmopoulos, A., Partalas, I., Gaussier, E., Paliouras, G., & Androutsopoulos, I. (2015). Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery*, *29*(3), 820–865. https://doi.org/10.1007/s10618-014-0382-x

Krogh, A., Brown, M., Mian, I. S., Sjölander, K., & Haussler, D. (1994). Hidden Markov Models in computational biology applications to protein modeling. *Journal of Molecular Biology*, *235*(5), 1501–1531. https://doi.org/10.1006/jmbi.1994.1104

Kumar, C., & Choudhary, A. (2012). A top-down approach to classify enzyme functional classes and sub-classes using random forest. *EURASIP Journal on Bioinformatics and Systems Biology 2012*, 1–4.

Kumar, S., Bhola, A., & Tiwari, A. K. (2015). Classification of enzyme functional classes and subclasses using support vector machine. *2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, ABLAZE 2015*, 411–417. https://doi.org/10.1109/ABLAZE.2015.7155031

Li, Y., Wang, S., Umarov, R., Xie, B., Fan, M., Li, L., & Gao, X. (2018). DEEPre: Sequence-based enzyme EC number prediction by deep learning. *Bioinformatics*, *34*(5), 760–769. https://doi.org/10.1093/bioinformatics/btx680

Li, Z. R., Lin, H. H., Han, L. Y., Jiang, L., Chen, X., & Chen, Y. Z. (2006). PROFEAT: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. *Nucleic Acids Research*, *34*(Web Server), W32–W37.

https://doi.org/10.1093/nar/gkl305

Lin, K., Quan, X., Jin, C., Shi, Z., & Yang, J. (2022). An Interpretable Double-Scale Attention Model for Enzyme Protein Class Prediction Based on Transformer Encoders and Multi-Scale Convolutions. *Frontiers in Genetics*, *13*(April), 1–12. https://doi.org/10.3389/fgene.2022.885627

Manduchi, E., Fu, W., Romano, J. D., Ruberto, S., & Moore, J. H. (2020). Embedding covariate adjustments in tree-based automated machine learning for biomedical big data analyses. *BMC Bioinformatics*, *21*(1), 430. https://doi.org/10.1186/s12859-020-03755-4

Marquet, C., Heinzinger, M., Olenyi, T., Dallago, C., Erckert, K., Bernhofer, M., Nechaev, D., & Rost, B. (2022). Embeddings from protein language models predict conservation and variant effects. *Human Genetics*, *141*(10), 1629–1647. https://doi.org/10.1007/s00439-021-02411-y

McDonald, A. G., & Tipton, K. F. (2023). Enzyme nomenclature and classification: the state of the art. *FEBS Journal*, *290*(9), 2214–2231. https://doi.org/10.1111/febs.16274

McGovern, P. E., Zhang, J., Tang, J., Zhang, Z., Hall, G. R., Moreau, R. A., Nuñez, A., Butrym, E. D., Richards, M. P., Wang, C. S., Cheng, G., Zhao, Z., & Wang, C. (2004). Fermented beverages of pre- and proto-historic China. *Proceedings of the National Academy of Sciences of the United States of America*, *101*(51), 17593–17598. https://doi.org/10.1073/pnas.0407921102

Meghwanshi, G. K., Kaur, N., Verma, S., Dabi, N. K., Vashishtha, A., Charan, P. D., Purohit, P., Bhandari, H. S., Bhojak, N., & Kumar, R. (2020). Enzymes for pharmaceutical and therapeutic applications. *Biotechnology and Applied Biochemistry*, *67*(4), 586–601. https://doi.org/10.1002/bab.1919

Memon, S. A., Khan, K. A., & Naveed, H. (2020). HECNet: A hierarchical approach to enzyme function classification using a siamese triplet network. *Bioinformatics*, *36*(17), 4583–4589. https://doi.org/10.1093/bioinformatics/btaa536

Miranda, F. M., Köehnecke, N., & Renard, B. Y. (2023). HiClass: a Python Library for Local Hierarchical Classification Compatible with Scikit-learn. *Journal of Machine Learning Research*, *24*(29), 1–17. https://jmlr.org/papers/v24/21-1518.html

Mistry, J., Chuguransky, S., Williams, L., Qureshi, M., Salazar, G. A., Sonnhammer, E. L. L., Tosatto, S. C. E., Paladin, L., Raj, S., Richardson, L. J., Finn, R. D., & Bateman, A. (2021). Pfam: The protein families database in 2021. *Nucleic Acids Research*, *49*(D1), D412–D419. https://doi.org/10.1093/nar/gkaa913

Nath, N., & Mitchell, J. B. O. (2012). Is EC class predictable from reaction mechanism? *BMC Bioinformatics*, *13*(1), 60. https://doi.org/10.1186/1471-2105-13-60

Oliveira, G. B., Pedrini, H., & Dias, Z. (2023). TEMPROT: protein function annotation using transformers embeddings and homology search. *BMC Bioinformatics*, *24*(1), 1–16. https://doi.org/10.1186/s12859-023-05375-0

Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (2016). Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 485–492. https://doi.org/10.1145/2908812.2908918

Olson, R. S., La Cava, W., Mustahsan, Z., Varik, A., & Moore, J. H. (2018). Data-driven advice for applying machine learning to bioinformatics problems. *Pacific Symposium on Biocomputing*, *0*, 192–203. https://doi.org/10.1142/9789813235533_0018

Orlenko, A., Moore, J. H., Orzechowski, P., Olson, R. S., & Wanglieweimayoedu, E. (2018). Considerations for automated machine learning in clinical metabolic profiling : Altered homocysteine plasma concentration associated with metformin exposure Metabolomics , the study of organic chemical signatures within a specimen , has been increasingly. *Pacific Symposium on Biocomputing*, 460–471.

Parmentier, L., Nicol, O., Jourdan, L., & Kessaci, M. E. (2019). TPOT-SH: A faster optimization algorithm to solve the AutoML problem on large datasets. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, *2019-Novem*, 471–478. https://doi.org/10.1109/ICTAI.2019.00072

Pathak, A., & Jayaram, B. (2022). Seq2Enz: An application of mask BLAST methodology with a new chemical logic of amino acids for improved enzyme function prediction. *Biochimica et Biophysica Acta - Proteins and Proteomics*, *1870*(1), 140721. https://doi.org/10.1016/j.bbapap.2021.140721

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Prakash, A., Jeffryes, M., Bateman, A., & Finn, R. D. (2017). The HMMER Web Server for Protein Sequence Similarity Search. *Current Protocols in Bioinformatics*, *60*(1). https://doi.org/10.1002/cpbi.40

Radzi, S. F. M., Karim, M. K. A., Saripan, M. I., Rahman, M. A. A., Isa, I. N. C., & Ibahim, M. J. (2021). Hyperparameter tuning and pipeline optimization via grid search method and tree-based autoML in breast cancer prediction. *Journal of Personalized Medicine*, *11*(10). https://doi.org/10.3390/jpm11100978

Resende, W. K., Nascimento, R. A., Xavier, C. R., Lopes, I. F., & Nobre, C. N. (2012). The use of support vector machine and genetic algorithms to predict protein function. *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 1773–1778. https://doi.org/10.1109/ICSMC.2012.6377994

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., & Fergus, R. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, *118*(15). https://doi.org/10.1073/pnas.2016239118

Romano, J. D., Le, T. T., Fu, W., & Moore, J. H. (2021). TPOT-NN: augmenting tree-based automated machine learning with neural network estimators. *Genetic Programming and Evolvable Machines*, *22*(2), 207–227. https://doi.org/10.1007/s10710-021-09401-z

Ryu, J. Y., Kim, H. U., & Lee, S. Y. (2019). Deep learning enables high-quality and high-throughput prediction of enzyme commission numbers. *Proceedings of the National Academy of Sciences of the United States of America*, *116*(28), 13996–14001. https://doi.org/10.1073/pnas.1821905116

Sasikumar, R., & Kalpana, V. (2016). Hidden Markov Model in Biological Sequence Analysis– A Systematic Review. *International Journal of Scientific and Innovative Mathematical Research*, *4*(3), 1–7. https://doi.org/10.20431/2347-3142.0403001

Schomburg, I., Chang, A., & Schomburg, D. (2014). Standardization in enzymology—Data integration in the world's enzyme information system BRENDA. *Perspectives in Science*, *1*(1–6), 15–23. https://doi.org/10.1016/j.pisc.2014.02.002

Sharma, A., Gupta, G., Ahmad, T., Mansoor, S., & Kaur, B. (2021). Enzyme Engineering: Current Trends and Future Perspectives. *Food Reviews International*, *37*(2), 121–154. https://doi.org/10.1080/87559129.2019.1695835

Shen, H.-B., & Chou, K.-C. (2007). EzyPred: A top–down approach for predicting enzyme functional classes and subclasses. *Biochemical and Biophysical Research Communications*, *364*(1), 53–59. https://doi.org/10.1016/j.bbrc.2007.09.098

Silla, C. N., & Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, *22*(1–2), 31–72. https://doi.org/10.1007/s10618-010-0175-9

Sohn, A., Olson, R. S., & Moore, J. H. (2017). Toward the automated analysis of complex diseases in genome-wide association studies using genetic programming. *GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference*, 489–496. https://doi.org/10.1145/3071178.3071212

Song, B., Li, Z., Lin, X., Wang, J., Wang, T., & Fu, X. (2021). Pretraining model for biological sequence data. *Briefings in Functional Genomics*, *20*(3), 181–195. https://doi.org/10.1093/bfgp/elab025

Spänig, S., & Heider, D. (2019). Encodings and models for antimicrobial peptide classification for multi-

resistant pathogens. *BioData Mining*, *12*(1), 1–29. https://doi.org/10.1186/s13040-019-0196-x

Stärk, H., Dallago, C., Heinzinger, M., & Rost, B. (2021). Light attention predicts protein location from the language of life. *Bioinformatics Advances*, *1*(1), 1–8. https://doi.org/10.1093/bioadv/vbab035

Strodthoff, N., Wagner, P., Wenzel, M., & Samek, W. (2020). UDSMProt: Universal deep sequence models for protein classification. *Bioinformatics*, *36*(8), 2401–2409. https://doi.org/10.1093/bioinformatics/btaa003

Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B., & Wu, C. H. (2015). UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, *31*(6), 926–932. https://doi.org/10.1093/bioinformatics/btu739

Thompson, R. H. S. (1962). Classification and Nomenclature of Enzymes. *American Association for the Advancement of Science*, *137*(3528), 405–408. https://www.jstor.org/stable/1708395

Tipton, K., & Boyce, S. (2000). History of the enzyme nomenclature system. *Bioinformatics*, *16*(1), 34–40. https://doi.org/10.1093/bioinformatics/16.1.34

Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C. B., & Farivar, R. (2019). Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, *2019-Novem*(2017), 1471–1479. https://doi.org/10.1109/ICTAI.2019.00209

van den Bent, I., Makrodimitris, S., & Reinders, M. (2021). The Power of Universal Contextualized Protein Embeddings in Cross-species Protein Function Prediction. *Evolutionary Bioinformatics Online*, *17*. https://doi.org/10.1177/11769343211062608

Vikhar, P. A. (2017). Evolutionary algorithms: A critical review and its future prospects. *Proceedings - International Conference on Global Trends in Signal Processing, Information Computing and Communication, ICGTSPICC 2016*, 261–265. https://doi.org/10.1109/ICGTSPICC.2016.7955308

Wang, S., Li, W., Liu, S., & Xu, J. (2016). RaptorX-Property: a web server for protein structure property prediction. *Nucleic Acids Research*, *44*(W1), W430–W435. https://doi.org/10.1093/nar/gkw306

Wang, S., Peng, J., Ma, J., & Xu, J. (2016). Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields. *Scientific Reports*, *6*(1), 18962. https://doi.org/10.1038/srep18962

Wang, Y.-C., Wang, Y., Yang, Z.-X., & Deng, N.-Y. (2011). Support vector machine prediction of enzyme function with conjoint triad feature and hierarchical context. *BMC Systems Biology*, *5*(S1), S6. https://doi.org/10.1186/1752-0509-5-S1-S6

Watanabe, N., Murata, M., Ogawa, T., Vavricka, C. J., Kondo, A., Ogino, C., & Araki, M. (2020). Exploration and Evaluation of Machine Learning-Based Models for Predicting Enzymatic Reactions. *Journal of Chemical Information and Modeling*, *60*(3), 1833–1843. https://doi.org/10.1021/acs.jcim.9b00877

Yoo, A. B., Jette, M. A., & Grondona, M. (2003). SLURM: Simple Linux Utility for Resource Management. In *Lecture Notes in Computer Science* (pp. 44–60). https://doi.org/10.1007/10968987_3

You, R., Zhang, Z., Xiong, Y., Sun, F., Mamitsuka, H., & Zhu, S. (2018). GOLabeler: Improving sequence-based large-scale protein function prediction by learning to rank. *Bioinformatics*, *34*(14), 2465–2473. https://doi.org/10.1093/bioinformatics/bty130

Zheng, J., Xiao, X., & Qiu, W. R. (2022). DTI-BERT: Identifying Drug-Target Interactions in Cellular Networking Based on BERT and Deep Learning Method. *Frontiers in Genetics*, *13*(June), 1–12. https://doi.org/10.3389/fgene.2022.859188

Zou, Z., Tian, S., Gao, X., & Li, Y. (2019). mlDEEPre: Multi-functional enzyme function prediction with hierarchical multi-label deep learning. *Frontiers in Genetics*, *10*(JAN), 1–10. https://doi.org/10.3389/fgene.2018.00714

Appendix A: Pipeline's number of sequences

Appendix A 1 Number of sequences per pipeline of Level 0 to Level 2 for each model.

	Lvl 0	*Lvl 1*	*Lvl 2 Class 1*	*Lvl 2 Class 2*	*Lvl 2 Class 3*	*Lvl 2 Class 4*	*Lvl 2 Class 5*	*Lvl 2 Class 6*	*Lvl 2 Class 7*
c40	90685	27568	3379	10116	7895	1522	1208	1725	717
Swiss	200000	213244	25574	74617	46518	19163	12395	24313	11573

Appendix A 2 All subclasses present at each class with the number of sequences for each available subclass at each model, and whether it was trained or removed from the model. Cases with a Trained mean the training dataset was altered in such a way it kept only the sub-subclass with the most sequences and joined the rest (and/or complemented with others from the others subclasses within the same class) into a class serving as a negative, turning the problem from a multiclass one to a binary one.*

		Swiss		**C40**	
Classes	Subclasses	Nº Sequences	Status	Nº Sequences	Status
Class 1	Subclass 1	6425	Trained	763	Trained*
	Subclass 2	2804	Trained	247	Trained
	Subclass 3	2501	Trained	316	Trained
	Subclass 4	1330	Trained	118	Trained
	Subclass 5	498	Trained	155	Trained*
	Subclass 6	495	Trained	64	Trained
	Subclass 7	822	Trained	59	Trained
	Subclass 8	1290	Trained	160	Trained
	Subclass 9	106	Trained	6	Removed
	Subclass 10	498	Trained*	35	Removed
	Subclass 11	1137	Trained*	108	Trained*
	Subclass 12	139	Trained	44	Removed
	Subclass 13	564	Trained	123	Trained*
	Subclass 14	3577	Trained	814	Trained
	Subclass 15	408	Trained*	28	Removed
	Subclass 16	231	Trained	58	Trained
	Subclass 17	1616	Trained	153	Trained
	Subclass 18	530	Trained	63	Trained*
	Subclass 20	53	Removed	9	Removed
	Subclass 21	143	Trained	30	Removed
	Subclass 23	13	Removed	1	Removed
	Subclass 97	394	Trained*	25	Removed
Class 2	Subclass 1	14521	Trained	1803	Trained
	Subclass 2	1237	Trained*	70	Removed
	Subclass 3	10074	Trained	1935	Trained
	Subclass 4	8501	Trained	1197	Trained
	Subclass 5	6715	Trained*	587	Trained*
	Subclass 6	1832	Trained	187	Trained*
	Subclass 7	26918	Trained	4072	Trained
	Subclass 8	4138	Trained	249	Trained

	Subclass 9	202	Trained*	9	Removed
	Subclass 10	14	Removed	7	Removed
Class 3	Subclass 1	12165	Trained	2385	Trained
	Subclass 2	4204	Trained	899	Trained
	Subclass 3	42	Removed	18	Removed
	Subclass 4	8963	Trained	1609	Trained
	Subclass 5	6641	Trained	703	Trained
	Subclass 6	9555	Trained	1200	Trained
	Subclass 7	175	Trained*	30	Removed
	Subclass 8	44	Removed	15	Removed
	Subclass 9	21	Removed	3	Removed
	Subclass 10	1	Removed	1	Removed
	Subclass 11	59	Trained*	4	Removed
	Subclass 12	2	Removed	2	Removed
	Subclass 13	236	Trained*	13	Removed
Class 4	Subclass 1	6243	Trained	494	Trained
	Subclass 2	8522	Trained	616	Trained
	Subclass 3	2302	Trained	139	Trained
	Subclass 4	390	Trained*	57	Trained*
	Subclass 5	5	Removed	3	Removed
	Subclass 6	1260	Trained*	157	Trained*
	Subclass 7	2	Removed	1	Removed
	Subclass 8	2	Removed	8	Removed
	Subclass 98	290	Trained*	19	Removed
	Subclass 99	115	Trained*	22	Removed
Class 5	Subclass 1	1736	Trained	222	Trained
	Subclass 2	1292	Trained*	222	Trained*
	Subclass 3	3812	Trained	262	Trained
	Subclass 4	3830	Trained	328	Trained
	Subclass 5	182	Trained*	38	Removed
	Subclass 6	1461	Trained	111	Trained*
	Subclass 99	3	Removed	1	Removed
Class 6	Subclass 1	10994	Trained*	625	Trained*
	Subclass 2	1096	Trained*	129	Trained*
	Subclass 3	10982	Trained	819	Trained
	Subclass 4	76	Trained*	18	Removed
	Subclass 5	1098	Trained*	112	Trained*
	Subclass 6	49	Removed	11	Removed
Class 7	Subclass 1	7652	Trained	391	Trained*
	Subclass 2	869	Trained	105	Trained*
	Subclass 3	586	Trained*	34	Removed
	Subclass 4	1032	Trained*	49	Removed
	Subclass 5	345	Trained*	25	Removed
	Subclass 6	1089	Trained*	113	Trained*

Appendix B: Subclasses score tables

Appendix B 1 Level 3 scores of the better scored pipelines all trained subclasses in Class 1 on both the C40 (clustered data) and Swiss models.

		ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	Sub 1	0.8735	0.7849	0.8051	0.7695	MLPClassifier
	Sub 2	0.8203	0.6442	0.6243	0.6693	SGDClassifier
	Sub 3	0.9220	0.8741	0.9270	0.8616	LinearSVC
	Sub 4	0.8682	0.7727	0.7938	0.7638	LogisticRegression
	Sub 5	0.7714	0.5373	0.5132	0.5714	LinearSVC
	Sub 8	0.8110	0.6335	0.6542	0.6428	LinearSVC
	Sub 11	0.98	0.9795	1.0	0.96	BernoulliNB
	Sub 13	0.9814	0.8905	0.8333	0.9814	GaussianNB
	Sub 14	0.8510	0.7278	0.7393	0.7213	SGDClassifier
	Sub 18	0.8333	0.9629	0.9285	1.0	LinearSVC
SWISS	Sub 1	0.8633	0.7259	0.7332	0.7327	MLPClassifier
	Sub 2	0.9198	0.8616	0.8935	0.8481	MLPClassifier
	Sub 3	0.9218	0.8490	0.8543	0.8455	MLPClassifier
	Sub 4	0.8896	0.8040	0.8510	0.7808	GaussianNB
	Sub 5	0.9777	0.9771	0.9959	0.9629	LinearSVC
	Sub 6	0.9817	0.9803	0.9894	0.9725	MLPClassifier
	Sub 7	1.0	1.0	1.0	1.0	MLPClassifier
	Sub 8	0.9294	0.9051	0.9899	0.8620	MLPClassifier
	Sub 9	1.0	1.0	1.0	1.0	MLPClassifier
	Sub 10	0.9962	0.9961	0.9967	0.9955	MLPClassifier
	Sub 11	0.9860	0.9844	0.9827	0.9861	MLPClassifier
	Sub 12	0.8474	0.7447	0.7502	0.7461	LinearSVC
	Sub 13	0.9974	0.9764	0.9583	0.9974	SGDClassifier
	Sub 14	0.9654	0.9410	0.9520	0.9363	LinearSVC
	Sub 15	1.0	1.0	1.0	1.0	LinearSVC
	Sub 16	1.0	1.0	1.0	1.0	SGDClassifier
	Sub 17	0.9257	0.8523	0.8514	0.8533	SGDClassifier
	Sub 18	0.9941	0.9918	0.9895	0.9941	ExtraTreesClassifier
	Sub 21	1.0	1.0	1.0	1.0	LogisticRegression
	Sub 97	1.0	1.0	1.0	1.0	LinearSVC

Appendix B 2 Level 3 scores of the better scored pipelines al trained subclasses in Class 2 on both the C40 (clustered data) and Swiss models.

		ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	Sub 1	0.9824	0.9881	0.9992	0.9777	LinearSVC
	Sub 3	0.9598	0.9579	0.9874	0.9334	MLPClassifier
	Sub 4	0.8409	0.7699	0.8602	0.7222	MLPClassifier
	Sub 5	0.9559	0.9342	0.9310	0.9375	MLPClassifier
	Sub 6	1.0	1.0	1.0	1.0	LinearSVC
	Sub 7	0.9615	0.9364	0.9535	0.9269	LinearSVC
	Sub 8	1.0	1.0	1.0	1.0	GradientBoostingClassifier
SWISS	Sub 1	0.9992	0.9993	0.9998	0.9989	MLPClassifier
	Sub 2	0.9984	0.9984	1.0	0.9969	MLPClassifier
	Sub 3	0.9969	0.9970	0.9974	0.9967	LinearSVC
	Sub 4	0.9891	0.9887	0.9972	0.9807	LogisticRegression
	Sub 5	0.9981	0.9959	0.9947	0.9970	LinearSVC
	Sub 6	1.0	1.0	1.0	1.0	ExtraTreesClassifier
	Sub 7	0.9918	0.9846	0.9851	0.9842	MLPClassifier
	Sub 8	0.9977	0.9983	0.9991	0.9974	SGDClassifier
	Sub 9	1.0	1.0	1.0	1.0	ExtraTreesClassifier

Appendix B 3 Level 3 scores of the better scored pipelines all trained subclasses in Class 3 on both the C40 (clustered data) and Swiss models.

		ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	Sub 1	0.9045	0.8284	0.8523	0.8127	LinearSVC
	Sub 2	0.9734	0.9778	0.9823	0.9734	MLPClassifier
	Sub 4	0.9043	0.8225	0.8497	0.8186	LinearSVC
	Sub 5	0.8899	0.8579	0.9545	0.8019	LinearSVC
	Sub 6	0.9848	0.9833	0.9900	0.9771	LinearSVC
SWISS	Sub 1	0.9503	0.9173	0.9393	0.9014	SGDClassifier
	Sub 2	0.9903	0.9928	0.9953	0.9903	MLPClassifier
	Sub 4	0.9928	0.9849	0.9829	0.9871	LinearSVC
	Sub 5	0.9960	0.9953	0.9974	0.9933	SGDClassifier
	Sub 6	0.9991	0.9988	0.9986	0.9989	LinearSVC
	Sub 7	0.9803	0.9315	0.8717	1.0	LinearSVC
	Sub 11	0.9615	0.96	1.0	0.9230	LogisticRegression
	Sub 13	1.0	1.0	1.0	1.0	DecisionTreeClassifier

Appendix B 4 Level 3 scores of the better scored pipelines all trained subclasses in Class 4 on both the C40 (clustered data) and Swiss models.

		ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	Sub 1	0.8733	0.7971	0.8017	0.8082	LinearSVC
	Sub 2	0.9136	0.8309	0.8219	0.8489	SGDClassifier
	Sub 3	0.9869	0.9680	0.9583	0.9821	LinearSVC
	Sub 4	0.8885	0.8780	0.9	0.8571	SGDClassifier
	Sub 6	1.0	1.0	1.0	1.0	LinearSVC
SWISS	Sub 1	0.9910	0.9889	0.9927	0.9851	MLPClassifier

Sub 2	0.9956	0.9958	0.9989	0.9928	LinearSVC
Sub 3	1.0	1.0	1.0	1.0	LinearSVC
Sub 4	0.9984	0.9897	0.9797	1.0	LinearSVC
Sub 98	1.0	1.0	1.0	1.0	LinearSVC
Sub 99	1.0	1.0	1.0	1.0	SGDClassifier

Appendix B 5 5 Level 3 scores of the better scored pipelines all trained subclasses in Class 5 on both the C40 (clustered data) and Swiss models.

		ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	Sub 1	0.8701	0.7459	0.7419	0.75	LinearSVC
	Sub 2	0.9738	0.9670	0.9777	0.9565	GradientBoostingClassifier
	Sub 3	0.9956	0.9810	0.9687	0.9954	LinearSVC
	Sub 4	0.9812	0.9861	0.9962	0.9772	LinearSVC
	Sub 6	1.0	1.0	1.0	1.0	MLPClassifier
SWISS	Sub 1	1.0	1.0	1.0	1.0	LinearSVC
	Sub 2	0.9992	0.9988	0.9984	0.9992	LinearSVC
	Sub 3	0.9831	0.9658	0.9661	0.9696	LinearSVC
	Sub 4	1.0	1.0	1.0	1.0	SGDClassifier
	Sub 5	1.0	1.0	1.0	1.0	LinearSVC
	Sub 6	1.0	1.0	1.0	1.0	MLPClassifier

Appendix B 6 Level 3 scores of the better scored pipelines all trained subclasses in Class 7 on both the C40 (clustered data) and Swiss models.

		ACCURACY	F1	PRECISION	RECALL	CLASSIFIER
C40	Sub 1	1.0	1.0	1.0	1.0	LinearSVC
	Sub 2	0.9851	0.9442	0.9166	0.9841	MLPClassifier
	Sub 6	0.9402	0.9392	0.9392	0.9392	MLPClassifier
SWISS	Sub 1	0.9985	0.9990	0.9995	0.9985	LinearSVC
	Sub 2	1.0	1.0	1.0	1.0	RandomForestClassifier
	Sub 3	0.9903	0.9901	0.9916	0.9888	SGDClassifier
	Sub 4	1.0	1.0	1.0	1.0	LinearSVC
	Sub 5	1.0	1.0	1.0	1.0	MLPClassifier
	Sub 6	0.9882	0.9881	0.9881	0.9882	DecisionTreeClassifier

Appendix C: Complete Pipelines.

Appendix C 1 Level 0 Complete optimized Pipelines. The Swiss are subsamples of the complete Swiss Dataset.

	BEST_PIPELINE
C40	KNeighborsClassifier(MinMaxScaler(LogisticRegression(input_matrix, C=25.0, dual=False, penalty=l2)), n_neighbors=8, p=2, weights=distance)
SWISS 1	KNeighborsClassifier(Binarizer(input_matrix, threshold=0.0), n_neighbors=2, p=2, weights=distance)
SWISS 2	KNeighborsClassifier(MaxAbsScaler(VarianceThreshold(input_matrix, threshold=0.001)), n_neighbors=2, p=2, weights=distance)

Appendix C 2 Level 1 Complete optimized Pipelines.

	BEST_PIPELINE
C40	MLPClassifier(RobustScaler(ZeroCount(RobustScaler(VarianceThreshold(OneHotEncoder (RobustScaler(VarianceThreshold(input_matrix, threshold=0.0005)), minimum_fraction=0.05, sparse=False, threshold=10), threshold=0.0005)))), alpha=0.1, learning_rate_init=0.001)
SWISS	KNeighborsClassifier(StandardScaler(PCA(RBFSampler(input_matrix, gamma=0.05), iterated_power=10, svd_solver=randomized)), n_neighbors=6, p=1, weights=distance)

Appendix C 3 Level 2 Complete optimized pipelines, per class of the two different models, C40 and Swiss.

	CLASSE	BEST_PIPELINE
C40	Class 1	LinearSVC(RFE(input_matrix, criterion=gini, max_features=0.25, n_estimators=100, step=0.25), C=10.0, dual=True, loss=hinge, penalty=l2, tol=0.1)
	Class 2	LinearSVC(PCA(input_matrix, iterated_power=4, svd_solver=randomized), C=10.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.001)
	Class 3	MLPClassifier(PCA(VarianceThreshold(input_matrix, threshold=0.0005), iterated_power=8, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.01)
	Class 4	LinearSVC(StandardScaler(DecisionTreeClassifier(input_matrix, criterion=entropy, max_depth=1, min_samples_leaf=9, min_samples_split=9)), C=5.0, dual=True, loss=hinge, penalty=l2, tol=0.01)
	Class 5	MLPClassifier(PCA(SelectFwe(MaxAbsScaler(input_matrix), alpha=0.005), iterated_power=5, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.01)
	Class 6	LinearSVC(SelectPercentile(input_matrix, percentile=65), C=15.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.01)
	Class 7	LinearSVC(ExtraTreesClassifier(StandardScaler(input_matrix), bootstrap=False, criterion=gini, max_features=0.25, min_samples_leaf=15, min_samples_split=9, n_estimators=100), C=5.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.0001)
SWISS	Class 1	LogisticRegression(StandardScaler(input_matrix), C=15.0, dual=False, penalty=l2)

Class 2	LinearSVC(MaxAbsScaler(Normalizer(MaxAbsScaler(Normalizer(MaxAbsScaler(input_matrix), norm=l2)), norm=l2)), C=0.1, dual=True, loss=squared_hinge, penalty=l2, tol=0.1)
Class 3	LogisticRegression(RobustScaler(input_matrix), C=10.0, dual=False, penalty=l2)
Class 4	MLPClassifier(PCA(StandardScaler(RobustScaler(input_matrix)), iterated_power=1, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.01)
Class 5	LinearSVC(CombineDFs(SelectPercentile(input_matrix, percentile=71), input_matrix), C=25.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.01)
Class 6	LinearSVC(SelectPercentile(input_matrix, percentile=63), C=15.0, dual=True, loss=hinge, penalty=l2, tol=0.01)
Class 7	LinearSVC(StandardScaler(input_matrix), C=0.1, dual=True, loss=squared_hinge, penalty=l2, tol=0.1)

Appendix C 4 Level 3 Complete optimized Pipelines of C40 subclasses.

CLASSE	SUBCLASSE	CLASSIFIER
CLASS 1	Sub 1	MLPClassifier(PCA(CombineDFs(input_matrix, input_matrix), iterated_power=10, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.01)
	Sub 2	SGDClassifier(CombineDFs(CombineDFs(input_matrix, input_matrix), StandardScaler(MinMaxScaler(MaxAbsScaler(input_matrix)))), alpha=0.0, eta0=0.01, fit_intercept=True, l1_ratio=0.75, learning_rate=constant, loss=squared_hinge, penalty=elasticnet, power_t=100.0)
	Sub 3	LinearSVC(RobustScaler(StandardScaler(VarianceThreshold(MLPClassifier(input_matrix, alpha=0.001, learning_rate_init=0.5), threshold=0.001))), C=0.1, dual=True, loss=squared_hinge, penalty=l2, tol=0.01)
	Sub 4	LogisticRegression(MLPClassifier(StandardScaler(DecisionTreeClassifier(input_matrix, criterion=gini, max_depth=3, min_samples_leaf=6, min_samples_split=10)), alpha=0.0001, learning_rate_init=0.5), C=25.0, dual=False, penalty=l2)
	Sub 5	LinearSVC(MaxAbsScaler(input_matrix), C=10.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.0001)
	Sub 6	LinearSVC(ZeroCount(ZeroCount(ExtraTreesClassifier(MinMaxScaler(input_matrix), bootstrap=False, criterion=entropy, max_features=0.2, min_samples_leaf=14, min_samples_split=11, n_estimators=100))), C=0.5, dual=False, loss=squared_hinge, penalty=l2, tol=1e-05)
	Sub 7	MLPClassifier(DecisionTreeClassifier(input_matrix, criterion=entropy, max_depth=2, min_samples_leaf=5, min_samples_split=17), alpha=0.01, learning_rate_init=0.01)
	Sub 8	LinearSVC(StandardScaler(RandomForestClassifier(GaussianNB(input_matrix), bootstrap=False, criterion=entropy, max_features=0.9000000000000001, min_samples_leaf=12, min_samples_split=3, n_estimators=100)), C=5.0, dual=False, loss=squared_hinge, penalty=l2, tol=1e-05)

	Sub 11	BernoulliNB(RBFSampler(input_matrix, gamma=0.05), alpha=0.01, fit_prior=False)
	Sub 13	GaussianNB(SGDClassifier(input_matrix, alpha=0.001, eta0=0.01, fit_intercept=True, l1_ratio=0.0, learning_rate=constant, loss=perceptron, penalty=elasticnet, power_t=0.5))
	Sub 14	SGDClassifier(RFE(input_matrix, criterion=gini, max_features=0.5, n_estimators=100, step=0.8), alpha=0.0, eta0=0.1, fit_intercept=False, l1_ratio=0.25, learning_rate=constant, loss=modified_huber, penalty=elasticnet, power_t=1.0)
	Sub 16	RandomForestClassifier(input_matrix, bootstrap=True, criterion=entropy, max_features=0.6500000000000001, min_samples_leaf=5, min_samples_split=9, n_estimators=100)
	Sub 17	SGDClassifier(input_matrix, alpha=0.001, eta0=1.0, fit_intercept=False, l1_ratio=0.75, learning_rate=constant, loss=squared_hinge, penalty=elasticnet, power_t=1.0)
	Sub 18	LinearSVC(StandardScaler(input_matrix), C=25.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.01)
CLASS 2	Sub 1	LinearSVC(ExtraTreesClassifier(StandardScaler(input_matrix), bootstrap=False, criterion=entropy, max_features=0.7000000000000001, min_samples_leaf=4, min_samples_split=10, n_estimators=100), C=10.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.0001)
	Sub 3	MLPClassifier(PCA(input_matrix, iterated_power=7, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.01)
	Sub 4	MLPClassifier(CombineDFs(input_matrix, RobustScaler(input_matrix)), alpha=0.01, learning_rate_init=0.01)
	Sub 5	MLPClassifier(PCA(MaxAbsScaler(input_matrix), iterated_power=4, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.001)
	Sub 6	LinearSVC(input_matrix, C=10.0, dual=True, loss=hinge, penalty=l2, tol=0.1)
	Sub 7	LinearSVC(CombineDFs(StandardScaler(input_matrix), input_matrix), C=20.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.1)
	Sub 8	GradientBoostingClassifier(LinearSVC(input_matrix, C=20.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.01), learning_rate=0.5, max_depth=1, max_features=0.7000000000000001, min_samples_leaf=20, min_samples_split=3, n_estimators=100, subsample=0.55)
CLASS 3	Sub 1	LinearSVC(StandardScaler(Normalizer(input_matrix, norm=l2)), C=0.5, dual=True, loss=hinge, penalty=l2, tol=0.01)
	Sub 2	MLPClassifier(StandardScaler(input_matrix), alpha=0.001, learning_rate_init=0.001)
	Sub 4	LinearSVC(RobustScaler(SelectPercentile(RobustScaler(input_matrix), percentile=75)), C=0.1, dual=False, loss=squared_hinge, penalty=l2, tol=0.01)
	Sub 5	LinearSVC(VarianceThreshold(RFE(input_matrix, criterion=entropy, max_features=0.1, n_estimators=100, step=0.2), threshold=0.0001), C=5.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.0001)

	Sub 6	LinearSVC(MaxAbsScaler(LinearSVC(input_matrix, C=25.0, dual=True, loss=hinge, penalty=l2, tol=0.001)), C=15.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.0001)
CLASS 4	Sub 1	LinearSVC(PCA(RobustScaler(ZeroCount(input_matrix)), iterated_power=10, svd_solver=randomized), C=0.01, dual=True, loss=squared_hinge, penalty=l2, tol=0.1)
	Sub 2	SGDClassifier(StandardScaler(SelectPercentile(input_matrix, percentile=55)), alpha=0.001, eta0=0.1, fit_intercept=False, l1_ratio=1.0, learning_rate=invscaling, loss=squared_hinge, penalty=elasticnet, power_t=0.5)
	Sub 3	LinearSVC(Normalizer(input_matrix, norm=max), C=25.0, dual=True, loss=squared_hinge, penalty=l2, tol=1e-05)
	Sub 4	SGDClassifier(StandardScaler(input_matrix), alpha=0.001, eta0=0.01, fit_intercept=False, l1_ratio=0.75, learning_rate=constant, loss=log, penalty=elasticnet, power_t=1.0)
	Sub 6	LinearSVC(input_matrix, C=20.0, dual=True, loss=hinge, penalty=l2, tol=1e-05)
CLASS 5	Sub 1	LinearSVC(input_matrix, C=15.0, dual=True, loss=hinge, penalty=l2, tol=1e-05)
	Sub 2	GradientBoostingClassifier(input_matrix, learning_rate=0.5, max_depth=7, max_features=0.7500000000000001, min_samples_leaf=17, min_samples_split=16, n_estimators=100, subsample=0.7000000000000001)
	Sub 3	LinearSVC(RFE(input_matrix, criterion=entropy, max_features=0.8500000000000001, n_estimators=100, step=0.2), C=10.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.01)
	Sub 4	LinearSVC(Normalizer(StandardScaler(GaussianNB(Normalizer(input_matrix, norm=l1))), norm=max), C=20.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.01)
	Sub 6	MLPClassifier(input_matrix, alpha=0.0001, learning_rate_init=0.01)
CLASS 6	Sub 1	SGDClassifier(MLPClassifier(input_matrix, alpha=0.001, learning_rate_init=0.5), alpha=0.01, eta0=0.01, fit_intercept=False, l1_ratio=0.0, learning_rate=constant, loss=perceptron, penalty=elasticnet, power_t=0.0)
	Sub 2	DecisionTreeClassifier(SGDClassifier(RobustScaler(input_matrix), alpha=0.0, eta0=1.0, fit_intercept=False, l1_ratio=1.0, learning_rate=constant, loss=hinge, penalty=elasticnet, power_t=0.1), criterion=gini, max_depth=1, min_samples_leaf=6, min_samples_split=13)
	Sub 3	LinearSVC(RFE(RobustScaler(input_matrix), criterion=entropy, max_features=0.35000000000000003, n_estimators=100, step=0.15000000000000002), C=20.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.1)
	Sub 5	LinearSVC(input_matrix, C=25.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.001)
CLASS 7	Sub 1	LinearSVC(input_matrix, C=25.0, dual=False, loss=squared_hinge, penalty=l2, tol=1e-05)

	Sub 2	MLPClassifier(MLPClassifier(input_matrix, alpha=0.0001, learning_rate_init=0.1), alpha=0.001, learning_rate_init=0.1)
	Sub 6	MLPClassifier(ExtraTreesClassifier(MaxAbsScaler(input_matrix), bootstrap=True, criterion=gini, max_features=0.9500000000000001, min_samples_leaf=20, min_samples_split=19, n_estimators=100), alpha=0.0001, learning_rate_init=0.01)

Appendix C 5 Level 3 Complete optimized Pipelines of Swiss subclasses.

CLASS	SUBCLASSE	BEST_PIPELINE
CLASS 1	Sub 1	MLPClassifier(CombineDFs(input_matrix, SelectPercentile(input_matrix, percentile=35)), alpha=0.001, learning_rate_init=0.01)
	Sub 2	MLPClassifier(LinearSVC(StandardScaler(input_matrix), C=0.5, dual=True, loss=hinge, penalty=l2, tol=1e-05), alpha=0.001, learning_rate_init=0.01)
	Sub 3	MLPClassifier(LinearSVC(LinearSVC(input_matrix, C=5.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.01), C=20.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.1), alpha=0.001, learning_rate_init=0.01)
	Sub 4	GaussianNB(LinearSVC(OneHotEncoder(MaxAbsScaler(input_matrix), minimum_fraction=0.25, sparse=False, threshold=10), C=0.5, dual=False, loss=squared_hinge, penalty=l1, tol=0.1))
	Sub 5	LinearSVC(ExtraTreesClassifier(input_matrix, bootstrap=True, criterion=entropy, max_features=0.6000000000000001, min_samples_leaf=19, min_samples_split=11, n_estimators=100), C=15.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.0001)
	Sub 6	MLPClassifier(SelectFwe(input_matrix, alpha=0.02), alpha=0.0001, learning_rate_init=0.1)
	Sub 7	MLPClassifier(RFE(input_matrix, criterion=entropy, max_features=0.8, n_estimators=100, step=0.7000000000000001), alpha=0.0001, learning_rate_init=0.01)
	Sub 8	MLPClassifier(GradientBoostingClassifier(DecisionTreeClassifier(MaxAbsScaler(input_matrix), criterion=gini, max_depth=2, min_samples_leaf=4, min_samples_split=2), learning_rate=0.01, max_depth=1, max_features=0.8500000000000001, min_samples_leaf=3, min_samples_split=10, n_estimators=100, subsample=0.15000000000000002), alpha=0.0001, learning_rate_init=0.01)
	Sub 9	MLPClassifier(input_matrix, alpha=0.01, learning_rate_init=0.001)
	Sub 10	MLPClassifier(RobustScaler(SelectPercentile(GradientBoostingClassifier(MLPClassifier(input_matrix, alpha=0.1, learning_rate_init=0.5), learning_rate=0.001, max_depth=2, max_features=0.45, min_samples_leaf=13, min_samples_split=19, n_estimators=100, subsample=0.3), percentile=73)), alpha=0.001, learning_rate_init=0.001)

	Sub 11	MLPClassifier(RFE(input_matrix, criterion=entropy, max_features=0.55, n_estimators=100, step=0.9500000000000001), alpha=0.1, learning_rate_init=0.001)
	Sub 12	LinearSVC(SelectFwe(SGDClassifier(StandardScaler(input_matrix), alpha=0.001, eta0=0.1, fit_intercept=True, l1_ratio=1.0, learning_rate=constant, loss=perceptron, penalty=elasticnet, power_t=50.0), alpha=0.048), C=0.5, dual=True, loss=hinge, penalty=l2, tol=0.001)
	Sub 13	SGDClassifier(MaxAbsScaler(CombineDFs(SGDClassifier(input_matrix, alpha=0.0, eta0=1.0, fit_intercept=True, l1_ratio=1.0, learning_rate=invscaling, loss=log, penalty=elasticnet, power_t=100.0), RFE(input_matrix, criterion=gini, max_features=0.15000000000000002, n_estimators=100, step=0.9500000000000001))), alpha=0.0, eta0=1.0, fit_intercept=False, l1_ratio=0.0, learning_rate=constant, loss=perceptron, penalty=elasticnet, power_t=0.1)
	Sub 14	LinearSVC(Normalizer(Normalizer(input_matrix, norm=max), norm=max), C=10.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.1)
	Sub 15	LinearSVC(input_matrix, C=20.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.01)
	Sub 16	SGDClassifier(input_matrix, alpha=0.001, eta0=0.1, fit_intercept=True, l1_ratio=1.0, learning_rate=constant, loss=hinge, penalty=elasticnet, power_t=50.0)
	Sub 17	SGDClassifier(SelectFwe(input_matrix, alpha=0.02), alpha=0.0, eta0=0.1, fit_intercept=False, l1_ratio=0.25, learning_rate=constant, loss=modified_huber, penalty=elasticnet, power_t=100.0)
	Sub 18	ExtraTreesClassifier(input_matrix, bootstrap=True, criterion=gini, max_features=0.8, min_samples_leaf=20, min_samples_split=3, n_estimators=100)
	Sub 20	MLPClassifier(input_matrix, alpha=0.0001, learning_rate_init=0.01)
	Sub 21	LogisticRegression(SGDClassifier(MaxAbsScaler(input_matrix), alpha=0.0, eta0=1.0, fit_intercept=False, l1_ratio=0.25, learning_rate=constant, loss=perceptron, penalty=elasticnet, power_t=0.1), C=5.0, dual=False, penalty=l2)
	Sub 97	LinearSVC(MLPClassifier(MinMaxScaler(input_matrix), alpha=0.1, learning_rate_init=0.01), C=0.1, dual=True, loss=squared_hinge, penalty=l2, tol=0.001)
CLASS 2	Sub 1	MLPClassifier(PCA(input_matrix, iterated_power=9, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.01)

	Sub 2	MLPClassifier(RFE(input_matrix, criterion=entropy, max_features=0.7000000000000001, n_estimators=100, step=0.8), alpha=0.0001, learning_rate_init=0.1)
	Sub 3	LinearSVC(PCA(input_matrix, iterated_power=3, svd_solver=randomized), C=10.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.1)
	Sub 4	LogisticRegression(SGDClassifier(MLPClassifier(input_matrix, alpha=0.001, learning_rate_init=0.01), alpha=0.001, eta0=1.0, fit_intercept=False, l1_ratio=0.25, learning_rate=invscaling, loss=perceptron, penalty=elasticnet, power_t=0.0), C=1.0, dual=False, penalty=l2)
	Sub 5	LinearSVC(LinearSVC(input_matrix, C=10.0, dual=True, loss=hinge, penalty=l2, tol=0.0001), C=20.0, dual=True, loss=squared_hinge, penalty=l2, tol=1e-05)
	Sub 6	ExtraTreesClassifier(input_matrix, bootstrap=False, criterion=gini, max_features=0.9500000000000001, min_samples_leaf=2, min_samples_split=11, n_estimators=100)
	Sub 7	MLPClassifier(SelectPercentile(input_matrix, percentile=58), alpha=0.0001, learning_rate_init=0.01)
	Sub 8	SGDClassifier(RobustScaler(input_matrix), alpha=0.001, eta0=0.01, fit_intercept=False, l1_ratio=1.0, learning_rate=constant, loss=modified_huber, penalty=elasticnet, power_t=0.5)
	Sub 9	ExtraTreesClassifier(input_matrix, bootstrap=False, criterion=gini, max_features=0.25, min_samples_leaf=4, min_samples_split=12, n_estimators=100)
CLASS 3	Sub 1	SGDClassifier(input_matrix, alpha=0.0, eta0=0.1, fit_intercept=True, l1_ratio=0.5, learning_rate=constant, loss=modified_huber, penalty=elasticnet, power_t=100.0)
	Sub 2	MLPClassifier(PCA(MaxAbsScaler(input_matrix), iterated_power=1, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.001)
	Sub 4	LinearSVC(RobustScaler(SelectPercentile(input_matrix, percentile=76)), C=5.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.001)

	Sub 5	SGDClassifier(StandardScaler(SGDClassifier(input_matrix, alpha=0.0, eta0=0.1, fit_intercept=True, l1_ratio=0.5, learning_rate=constant, loss=modified_huber, penalty=elasticnet, power_t=1.0)), alpha=0.0, eta0=0.01, fit_intercept=True, l1_ratio=0.75, learning_rate=constant, loss=log, penalty=elasticnet, power_t=1.0)
	Sub 6	LinearSVC(MLPClassifier(input_matrix, alpha=0.0001, learning_rate_init=0.5), C=10.0, dual=True, loss=hinge, penalty=l2, tol=0.1)
	Sub 7	LinearSVC(MaxAbsScaler(input_matrix), C=5.0, dual=True, loss=squared_hinge, penalty=l2, tol=1e-05)
	Sub 11	LogisticRegression(input_matrix, C=25.0, dual=False, penalty=l2)
	Sub 13	DecisionTreeClassifier(ExtraTreesClassifier(input_matrix, bootstrap=False, criterion=gini, max_features=0.35000000000000003, min_samples_leaf=11, min_samples_split=11, n_estimators=100), criterion=gini, max_depth=10, min_samples_leaf=16, min_samples_split=18)
CLASS 4	Sub 1	MLPClassifier(PCA(PCA(input_matrix, iterated_power=8, svd_solver=randomized), iterated_power=8, svd_solver=randomized), alpha=0.0001, learning_rate_init=0.01)
	Sub 2	LinearSVC(SGDClassifier(RobustScaler(input_matrix), alpha=0.0, eta0=1.0, fit_intercept=False, l1_ratio=1.0, learning_rate=constant, loss=hinge, penalty=elasticnet, power_t=0.5), C=0.5, dual=True, loss=squared_hinge, penalty=l2, tol=0.0001)
	Sub 3	LinearSVC(MaxAbsScaler(input_matrix), C=5.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.001)
	Sub 4	LinearSVC(DecisionTreeClassifier(input_matrix, criterion=gini, max_depth=2, min_samples_leaf=12, min_samples_split=3), C=5.0, dual=True, loss=hinge, penalty=l2, tol=1e-05)
	Sub 6	LinearSVC(Normalizer(input_matrix, norm=l2), C=25.0, dual=True, loss=hinge, penalty=l2, tol=0.0001)
	Sub 98	LinearSVC(ExtraTreesClassifier(input_matrix, bootstrap=False, criterion=entropy, max_features=0.1, min_samples_leaf=8, min_samples_split=19, n_estimators=100), C=10.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.001)

	Sub 99	SGDClassifier(LinearSVC(SGDClassifier(input_matrix, alpha=0.01, eta0=0.01, fit_intercept=True, l1_ratio=1.0, learning_rate=invscaling, loss=squared_hinge, penalty=elasticnet, power_t=100.0), C=1.0, dual=True, loss=hinge, penalty=l2, tol=0.0001), alpha=0.0, eta0=0.01, fit_intercept=True, l1_ratio=0.0, learning_rate=constant, loss=squared_hinge, penalty=elasticnet, power_t=10.0)
CLASS 5	Sub 1	LinearSVC(GradientBoostingClassifier(input_matrix, learning_rate=1.0, max_depth=9, max_features=0.9000000000000001, min_samples_leaf=12, min_samples_split=8, n_estimators=100, subsample=0.05), C=5.0, dual=True, loss=hinge, penalty=l2, tol=0.0001)
	Sub 2	LinearSVC(CombineDFs(input_matrix, input_matrix), C=15.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.01)
	Sub 3	LinearSVC(LinearSVC(input_matrix, C=1.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.1), C=0.5, dual=False, loss=squared_hinge, penalty=l2, tol=0.01)
	Sub 4	SGDClassifier(PCA(input_matrix, iterated_power=9, svd_solver=randomized), alpha=0.0, eta0=1.0, fit_intercept=True, l1_ratio=0.0, learning_rate=invscaling, loss=squared_hinge, penalty=elasticnet, power_t=0.0)
	Sub 5	LinearSVC(PCA(MaxAbsScaler(input_matrix), iterated_power=3, svd_solver=randomized), C=25.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.1)
	Sub 6	MLPClassifier(input_matrix, alpha=0.001, learning_rate_init=0.1)
CLASS 6	Sub 1	LinearSVC(input_matrix, C=1.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.01)
	Sub 2	MLPClassifier(GradientBoostingClassifier(LinearSVC(input_matrix, C=10.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.01), learning_rate=1.0, max_depth=8, max_features=0.15000000000000002, min_samples_leaf=3, min_samples_split=2, n_estimators=100, subsample=0.15000000000000002), alpha=0.01, learning_rate_init=0.01)
	Sub 3	LinearSVC(SGDClassifier(input_matrix, alpha=0.0, eta0=0.1, fit_intercept=True, l1_ratio=0.25, learning_rate=invscaling, loss=modified_huber, penalty=elasticnet, power_t=10.0), C=10.0, dual=False, loss=squared_hinge, penalty=l1, tol=0.1)

	Sub 4	GradientBoostingClassifier(input_matrix, learning_rate=0.1, max_depth=8, max_features=0.45, min_samples_leaf=7, min_samples_split=16, n_estimators=100, subsample=0.7500000000000001)
	Sub 5	SGDClassifier(input_matrix, alpha=0.0, eta0=0.1, fit_intercept=True, l1_ratio=0.5, learning_rate=constant, loss=squared_hinge, penalty=elasticnet, power_t=0.1)
CLASS 7	Sub 1	LinearSVC(Normalizer(input_matrix, norm=max), C=10.0, dual=True, loss=squared_hinge, penalty=l2, tol=0.001)
	Sub 2	RandomForestClassifier(SGDClassifier(RobustScaler(input_matrix), alpha=0.0, eta0=0.1, fit_intercept=False, l1_ratio=0.25, learning_rate=invscaling, loss=perceptron, penalty=elasticnet, power_t=0.1), bootstrap=True, criterion=entropy, max_features=0.7000000000000001, min_samples_leaf=6, min_samples_split=5, n_estimators=100)
	Sub 3	SGDClassifier(PCA(input_matrix, iterated_power=1, svd_solver=randomized), alpha=0.0, eta0=0.1, fit_intercept=True, l1_ratio=1.0, learning_rate=constant, loss=modified_huber, penalty=elasticnet, power_t=0.5)
	Sub 4	LinearSVC(input_matrix, C=10.0, dual=False, loss=squared_hinge, penalty=l2, tol=0.01)
	Sub 5	MLPClassifier(PCA(SelectFwe(input_matrix, alpha=0.042), iterated_power=9, svd_solver=randomized), alpha=0.001, learning_rate_init=0.01)
	Sub 6	DecisionTreeClassifier(LinearSVC(RobustScaler(input_matrix), C=0.5, dual=False, loss=squared_hinge, penalty=l2, tol=0.01), criterion=gini, max_depth=9, min_samples_leaf=1, min_samples_split=8)

UNIVERSIDADE NOVA
DE LISBOA

2023

A Study of Machine Learning for Artificial Intelligence-Based Enzyme Classification.

Ana Patrícia Fernandes Brás da Silva
MASTER IN COMPUTATIONAL BIOLOGY & BIOINFORMATICS

UNIVERSIDADE NOVA DE LISBOA