NOVA IMS
Information Management School

MAA

Master Program in Advanced Analytics

Credit Scoring Using Genetic Programming

David Micha Horn

Advisor: Professor Leonardo Vanneschi

Internship report presented as partial requirement for obtaining the Master's degree in Advanced Analytics

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

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

CREDIT SCORING USING GENETIC PROGRAMMING

by

David Micha Horn

Internship report presented as partial requirement for obtaining the Master's degree in Advanced Analytics.

Advisor: Leonardo Vanneschi

November 2016

ACKNOWLEDGEMENTS

"I am not going to thank anybody, because I did it all on my own"

— Spike Milligan*, on receiving the British Comedy Award for Lifetime Achievement*

I would like to thank my advisor Professor Leonardo Vanneschi who sparked my interest in Genetic Programming and gave me important remarks on the thesis.

Furthermore, I would like to express my gratitude to Dr. Klaus-Peter Huber who gave me the opportunity to do an internship and write my thesis with Arvato Financial Solutions.

My thank also goes to my advisor Sven Wessendorf for giving me guidance and the very necessary suggestions for improvements in this thesis.

Special thanks to Dr. Karla Schiller for introducing and explaining the process of credit scoring to me. Additionally, I would like to thank her and Dr. Markus Höchststötter for letting me take part in their project, and thus providing me with a topic for this thesis.

I gratefully acknowledge the support given by Achim Krauß and Hannes Klein towards my understanding of the data used in this thesis as well as Isabell Schmitt whose knowledge of administrative mechanisms and resources within Arvato Financial Solutions proved extremely beneficial for my work.

Finally, I would like to thank everyone at NOVA IMS and Arvato Financial Solutions I met and worked with.

ABSTRACT

Growing numbers in e-commerce orders lead to an increase in risk management to prevent default in payment. Default in payment is the failure of a customer to settle a bill within 90 days upon receipt. Frequently, credit scoring is employed to identify customers' default probability. Credit scoring has been widely studied and many different methods in different fields of research have been proposed.

The primary aim of this work is to develop a credit scoring model as a replacement for the pre risk check of the e-commerce risk management system risk solution services (rss). The pre risk check uses data of the order process and includes exclusion rules and a generic credit scoring model. The new model is supposed to work as a replacement for the whole pre risk check and has to be able to work in solitary and in unison with the rss main risk check.

An application of Genetic Programming to credit scoring is presented. The model is developed on a real world data set provided by Arvato Financial Solutions. The data set contains order requests processed by rss. Results show that Genetic Programming outperforms the generic credit scoring model of the pre risk check in both classification accuracy and profit. Compared with Logistic Regression, Support Vector Machines and Boosted Trees, Genetic Programming achieved a similar classificatory accuracy. Furthermore, the Genetic Programming model can be used in combination with the rss main risk check in order to create a model with higher discriminatory power than its individual models.

KEYWORDS

Risk Management; Credit Scoring; Genetic Programming; Machine Learning; Optimization

INDEX

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

E-commerce vendors in Germany have to deal with a peculiarity. Commonly used payment types like credit cards or PayPal represent a relatively low market share and the majority of orders are processed using open invoice instead. Using open invoice, a vendor bills his customers for goods and services only after delivery of the product. Thus, the vendor grants his customer a credit to the extend of the invoice. Usually, the vendor sends his customers an invoice statement as soon as the products are delivered or provided. The invoice contains a detailed statement of the transaction. Because the customer receives his purchase before payment, it is called 'open' and once the payment is received the invoice is closed.

Around 28% of customers in Germany choose open invoice as payment type, a market share that comes with reservations for the vendors (Frigge, 2016). Customers are used to pay using open invoice in an extend that it is a payment type selection requirement. Incidentally, around 30% of customers who abort the purchasing process do so because their desired payment type is unavailable and 68% of customers name open invoice the most desired payment type next to the ones already available (Fittkau & Maaß Consulting, 2014; Wach, 2011). However, open invoice is most prone to payment disruptions. Among the most common reasons vendors find that customers simply forget to settle the bill or delay the payment on purpose but around 53% of vendors named insolvency as one of the most common reasons for payment disruption (Weinfurner et al., 2011). The majority of cases that conclude in default on payment are assigned to orders with open invoice as payment type, with more than 8% of all orders defaulting (Seidenschwarz et al., 2014). E-commerce vendors find themselves in a conflict - offering open invoice decreases the order abortion rate but increases the default on payment rate. The former has a positive effect on revenue while the latter drives it down. Additionally, default on payment has a negative impact on the profit margin, due to costs arising through provision of services and advance payments to third parties. In order to break through this vicious circle vendors can fall back on a plethora of methods. Many tackle this conflict by implementing exclusion rules for customer groups they consider especially default-prone, for instance customers who are unknown to the vendor or whose order values are conspicuously high. Another approach, used by more than 30% of e-Commerce vendors in Germany, is to fall back on external risk management services (Weinfurner et al., 2011).

Risk management applications aim at detecting customers with a high risk of defaulting. Those applications are frequently build using credit scoring models. Credit Scoring analyzes historical data in order to isolate meaningful characteristics that are used to predict the probability of default (Mester, 1997). However, the probability of default is not an attribute of potential customers but merely a vendors' assessment if the potential customer is a risk worth taking. Over the years, credit scoring has evolved from a vendors' gut decision over subjective decision rules to a method based on statistically sound models (Thomas et al., 2002).

Among the providers of risk management services in Germany is the risk management division of Arvato Financial Solutions (AFS), which provides a number of services including identification of individuals, evaluation of credit-worthiness and fraud recognition. The AFS databases consist of 21 million solvency observations totaling information of 7 million individuals in Germany, address and change in address information, bank account information as well as phone numbers, email addresses and device information. The risk management service for e-commerce is called risk solution service (rss). Rss covers the entire order process and provides a number of services for every stage during the order process. The main services for evaluation of customers default probability is called risk check and split into a pre

risk check and a main risk check in order to satisfy different demands in an international environment. The main risk check is based on a credit agency score that uses country specific solvency information on individuals. Hence, the main risk check is inoperable in countries without accessible solvency information. Contrarily, the pre risk check was designed to be always operable and to ensure that the risk check returns an evaluation of the customers' default probability. For this purpose, the pre risk check uses data transmitted by the customer during the order process. However, the pre risk check is based on a generic model without statistical sound backup. In this work Genetic Programming is used to build a credit scoring model to replace the existing rss pre risk check.

Proposed by Koza (1992), Genetic Programming is an evolutionary computation method in the research area of optimization that searches for a solution in a search space. For this purpose a population of potential solutions is created and then submitted to an optimization process that makes use of the idea of survival of the fittest. Inspired by Darwin's theory about evolution, Genetic Programming employs evolutionary mechanisms such as inheritance, selection, crossover and mutation in order to gradually evolve new solutions to a problem. In a credit scoring environment, Genetic Programming initializes a population of discriminant functions in order to classify customers into bads and goods. This population is subsequently submitted to the genetic operators to find the best discriminant function.

Marques et al. (2013) state five major characteristics of computational intelligence systems such as evolutionary computation that are especially appealing in credit scoring:

- Learning
- Adaption
- Flexibility
- Transparency
- Discovery

The first characteristic, learning, describes the ability to learn decisions and tasks from historical data. Adaption represents the capability to adapt to a changing environment. Hence, there is no restriction to specific situations or economic conditions. The flexibility of computational intelligence systems allows for utilization even with incomplete or unreliable data sets. Furthermore, Marques et al. (2013) describe computational intelligence systems to be transparent, is a sense that resulting decisions are explainable. The outputs may be revisable in general, but for example Goldberg (1989) criticizes the poor comprehensibility of Genetic Algorithms which can be transfered to Genetic Programming. Taking into account that Marques et al. (2013) point out the potential importance to provide explanations of how decisions have been made due to legal reasons, transparency as characteristic has to be called into question. Lastly, discovery represents the ability to discover previously unknown relationships.

Narrowing down the scope from computational intelligence systems to Genetic Programming, Ong et al. (2005) argues that Genetic Programming has a number of inherit characteristics that make it attractive for the application in credit scoring. First, it is a non-parametric tool and not restricted to specific situations or data sets but can be used in a vast context. Second, it automatically and heuristically determines the most fitting discriminant function. Lastly, Genetic Programming selects the important variables automatically. Those characteristics are specific consequences from the characteristics for computational intelligence systems as stated by Marques et al. (2013).

Research repeatedly showed the benefits of Genetic Programming and its utility in Credit Scoring. However, credit scoring is usually employed using data from the financial sector and other sectors are rarely considered. While credit scoring was originally implemented by mail-order companies, it is unclear if benefits remain present but underutilized. Hence, the motivation and aim of this work is to extend current research in credit scoring by employing Genetic Programming on a data set that contains orders from e-commerce vendors, a field that has so far been neglected.

This work is organized as follows. In Section 2 the literature of credit scoring with an emphasis on the application of Genetic Programming is reviewed. Section 3 describes the risk solution services, analyzes the current state of the pre risk check and defines the reasoning and target of this work. Section 4 describes the dataset, the respective preparation steps, the overall work flow and reviews Genetic Programming including the specifics used. Section 5 presents the analytical results of a credit scoring model based on Genetic Programming, Section 6 presents the conclusions and Section 7 suggestions for future research.

2. THEORETICAL FRAMEWORK

Credit Scoring is a widely used application by financial institutions for the determination of applicants default probability and the subsequent classification into a good applicant group (the ”goods”) or a bad applicant group (the ”bads”) (Thomas et al., 2002). Consequentially, applicants are rejected or accepted as customers based on the classification. Credit scoring represents a binary classification problem that enables the usage of methods that deal with binary response data (Henley, 1995). To evaluate the credit risk of loan applications, Credit Scoring targets at isolating effects of applicants' characteristics on their default probability. The default probability is mapped to a numerical expression – the score – that indicates the creditworthiness of the applicant and enables the creditor to rank order applicants. The relationship between default probability and scores is depicted in figure 1. The mean scores over all

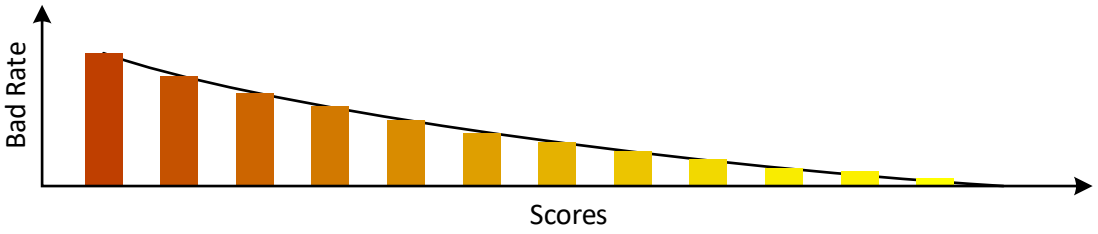

Figure 1: Relationship between Default Probability and Scores

applicants who perform well should be higher than the mean scores over all applicants who perform badly. The better the mean scores separate good and bad applicants, the higher the discriminatory power of the model (Mester, 1997). Figure 2 shows the distributions of bads and goods for four example models, with the scores on the x-axis and the number of observations on the y-axis. The bads are represented by the red line, whereas the goods are represented by the green line. The dashed line depicts the mean values for bads and goods. In model (a) bads and goods are non-overlapping, and hence the model has maximum discriminatory power. Model (b) shows slightly overlapping goods and bads. Thus, the discriminatory power is reduced compared to the maximum, but still on a high level. In model (c) goods and bads are heavily overlapping which implies low discriminatory power. Finally, the mean scores of goods and bads in model (d) are identical, and hence the model incorporates no discriminatory power. The classification and hereupon decision if an applicant is being rejected or accepted is

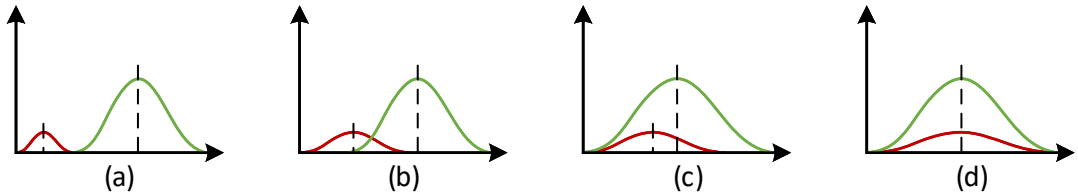

Figure 2: Example distributions of goods and bads

taken by comparing the applicants credit score with a predefined threshold. Thus, the creditworthiness is not an attribute of the applicant but an assessment if the applicant represents a risk willing to be taken by the lender (Thomas et al., 2002). Scoring Models are built upon historical data of applications, including application details and information about the true outcome of an accepted application, called performance. However, unlike application information, information about the performance of rejected applicants is usually unavailable. Developing a model only on accepted application data does not take

into account risk characteristics of rejected applications. Models build solely on data about accepted applications should only be used to assess the probability of default for a similar population of accepted requests and thus contradicting the problem specifications of credit scoring. In order to decrease the bias in data, an approach called ”reject inference” is often times employed. Reject Inference targets at inferring the true status of rejected applications (Thomas, 2000; Henley, 1995).

The benefits of a credit scoring model for financial institutions are threefold. First, the application of a credit scoring model reduces the time needed for the approval process. The possible time savings vary depending on how strictly the proposed cut-off threshold is followed. If the threshold is followed strictly, credit applicants above or below the threshold are automatically accepted or rejected. Contrariwise, if the threshold is not followed strictly, the applications within a certain range around the threshold can be reevaluated. In either case the efficiency of credit scoring can improve greatly because applicants far away from the threshold are automatically detected and categorized. Second, because applying and handling the application takes less time, both parties save money. Third and lastly, both creditor and applicant benefit from an increased objectivity in the loan approval process. Using a credit score model ensures that the same criteria are applied to all applicants regardless of the personal feelings from the creditor’s person in charge towards the applicant. Additionally, the creditor gains the capability to document factors with a disproportionally negative effect on certain groups of applicants (Mester, 1997).

The objectives in credit scoring for mail-order and e-commerce vendors differ from finance and banking. Rejected individuals are not denied from purchasing goods but restricted to secure payment methods at point of sale. Secure payment methods require the customer to pay the ordered goods before shipment or on delivery, e.g. advanced payment, credit card, pay on delivery or PayPal. Denying open invoice and other insecure payment methods increases the abortion rate during the payment process. Individuals retained in the payment process are usually measured by the conversion rate, i.e. the ratio of individuals finishing the order process to individuals entering the payment process. Hence, credit scoring in e-commerce aims at lowering default on payment by rejecting individuals with a high probability of default from purchasing goods using insecure payment methods whilst retaining a maximum conversion rate.

Sackmann et al. (2011) divide the risk of default in payment into two dimensions. The first dimension is represented by individuals who are unable to settle their bills because of insolvency or illiquidity and the second dimension is represented by individuals who are unwilling to settle their bills because they engage in fraud. Using those dimensions, they define four risk categories.

1. Ability and Willingness to settle bill
2. Inability to settle bill
3. Unwillingness to settle bill
4. Inability and Unwillingness to settle bill

The effect of both dimensions for the vendor is the same, i.e. disruption or default in payment. The causes, however, differ and demand adjusted prevention approaches. Inability to pay due to insolvency or illiquidity are usually identified using classical credit scoring models as described above. Unwillingness to pay can be detected by identifying fraud pattern such as irregular order values and volumes. Fraud is also regularly committed using account data of public institutions or charity organizations and

by impersonating others for which adequate prevention measures exists. Fraud prevention can also be integrated into credit scoring models, which covers the fourth risk category (Sackmann et al., 2011). Additionally, those approaches may have a positive effect with individuals who enter their information wrongly by accident but are both able and willing to settle their bills. Denying them from a purchase based on insecure payment methods may prompt them to review their input data and thus decrease wrong deliveries and dunning costs.

The ability to combine the identification of those who are unable and those who are unwilling to settle their bills elevates credit scoring to a very capable risk management tool in e-commerce. Incidentally, the history of credit scoring goes back to mail-order companies (the predecessor of e-commerce vendors) in the 1930s. Credit decisions were made on the basis of subjective judgment by credit analysts. Thus, applicants may be rejected by a credit analyst but accepted by another. In an effort to diminish the respective inconsistencies mail-order companies introduced a numeric scoring system. Shortly thereafter, Durand (1941) used Fisher's linear discriminant introduced by Fisher (1936) to differentiate between good and bad loans. Results were not utilized in a predictive setting but to extract attributes that indicate good and bad borrowers. Credit scoring was boosted with the start of World War II because credit analysts were drafted into military service leading to a shortage in personnel with respective knowledge. Hence, credit analysis provided the rules of thumb they used in the credit decision process for non-experts to be used (Thomas et al., 2002).

After the war, credit scoring slowly emerged as target of scientific research. Wolbers (1949) investigated the effect of credit scoring in a department store chain and McGrath (1960) for an automobile dealer. Both report major decline in credit loss. Myers and Forgy (1963) developed a scoring system to replace a numerical scoring system based on pooled judgment of credit analysts. Although they noted that their scoring system was adopted they do not offer a comparison with the existing system but only between different systems they proposed. In their conclusion they remark that they did not select the system with the highest performance due to unacceptable weights assignments. In 1956, Bill Fair and Earl Isaac founded Fair, Isaac, and Company, the first consultancy to provide a credit scoring system and nowadays known as FICO. Incidentally, FICO is the predecessor of the risk management division of AFS.

The decisive milestone in the history of credit scoring was the Equal Credit Opportunity Act (ECOA) in the United States of America in 1974. Under the ECOA race, color, national origin, religion, martial status, age, sex or level of involvement in public assistance is prohibited to be used in credit scoring. Furthermore, the ECOA classifies scoring systems into those that are "empirically derived and statistically valid" and those that are not. As a result, judgmental scoring systems were weakened and scoring systems based on statistically valid models generally accepted (United States Code, 1974; Thomas et al., 2002; Mays, 2001).

In 1975, the Basel Committee on Banking Supervision (BCBS) was founded as a forum for regular cooperation on banking supervisory matters. Originally consisting of the group of ten ("G-10"), the committee grew over the years and currently lists members from 27 countries as well as the European Union. The committee aims at enhancing financial stability by formulating supervisory standards and guidelines without legal force. With the document about international convergence of capital measurement and capital standards (more commonly known as Basel II Accords), a regulatory requirement for banks and financial organizations to adopt advanced credit scoring models was established and officially recommended in 2004. The Basel II Accords state that capital allocations have to be credit risk dependent and

that credit risk has to be quantified based on formal methods using data. Hence, it is virtually impossible for organizations in member countries to employ risk sensitive capital allocations without the use of credit scoring techniques (Khashman, 2010; Oreski et al., 2012; Marques et al., 2013). As a result, credit scoring attracted additional notice from scientists and lead to an increase in research. The Basel Accords only affect banking and financial organizations which directs the focus of published papers on these research areas.

Over the years a great number of different approaches to obtain a satisfactory credit scoring model have been proposed. The classical approaches involve methods such as linear discriminant models (Reichert et al., 1983), logistic regression (Wiginton, 1980; Henley, 1995), k-nearest neighbors (Henley and Hand, 1996) and decision trees (Davis et al., 1992), but more sophisticated approaches such as neural networks (Desai et al., 1996; Malhotra and Malhotra, 2002; West, 2000) and genetic programming (Ong et al., 2005; Abdou, 2009) have also been utilized.

One of the few papers using genetic programming for credit scoring by Ong et al. (2005) compared artificial neural networks, classification and regression tree, C4.5, rough sets and logistic regression with plain genetic programming using two well known data sets, namely, credit data sets from Australia and Germany. Genetic programming was used in order to determine the correct discriminant function automatically and to use the model with both big and small data sets. Comparing the error rates, the authors showed that genetic programming outperforms the other models in both data sets but noted that artificial neural networks and logistic regression also performed well. The authors concluded that GP is a non-parametric tool that is not based on any assumptions concerning the data set, which makes GP suitable for any situations and data sets. Additionally, the authors stated that GP is more flexible and accurate than the compared techniques.

Huang et al. (2007) investigated the credit scoring accuracy of three hybrid Support Vector Machines (SVM) based data mining approaches on the Australian and German data set. The SVM models were combined with the grid search approach in order to improve model parameters, the F-Score for features selection and Genetic Algorithms in order to obtain both the optimal features and parameters automatically at the same time. The results of the hybrid SVM models are compared to other data mining approaches based on Artificial Neural Networks, Genetic Programming and C4.5. The authors found that the best results were achieved by Genetic Programming, followed by their hybrid SVM approach. They pointed out that GP and their hybrid SVM approach used less features than input variables due to the automatic feature selection.

Zhang et al. (2007) apply Genetic Programming, Backpropagation Neural Networks and Support Vector Machines on credit scoring problems using the Australian and German data sets. Additionally, they construct a model by combining the classification results of the other models using majority voting. The accuracy is used as evaluation criteria. They show that the different models obtain good classification results but their accuracies differ little. Furthermore, the combined model shows better overall accuracies.

Abdou (2009) compared two Genetic Programming models with Probit Analysis and Weight of Evidence in the Egyptian Public Banking Sector. The data set was provided by Egyptian commercial public sector banks. The author used both the average correct classification rate and the estimated misclassification cost as evaluation criteria. While the average correct classification rate measures the proportion of cor-

rectly classified cases, the estimated misclassification cost takes into account the costs that arise from different types of error. In his conclusion the author stated that the preferred model depends on which evaluation criterion is used but that the results are close.

Chen and Huang (2003) used credit scoring models based on artificial neural networks (NN) and genetic algorithms (GA) on the Australian and German data sets. The NN model was used for classification, followed by a GA-based inverse classification. The latter was implemented to gain a better understanding of the rejected instances and to reassign them to a preferable accepted class, which balances between cost and preference. They conclude that NNs are a promising method and GAs incorporates attractive features for credit scoring.

A two-stage GP model to harvest the advantages of function-based and induction-based methods is implemented by Huang et al. (2006) using the Australian and German data sets. In the first stage, GP was used to derive if-then-else rules and in the second stage, GP was used to construct a discriminative function from the reduced data set. The two-stages split was implemented to restrict the model complexity and therefore ensured comprehension of the decision rules. Subsequently, only few general rules for every class were derived instead of every rule for the whole data set. The second stage derived the discriminative function for forecasting purposes. The results were compared to a number of data mining techniques but showed no significant improvement of two-stage GP over plain GP. However, the authors pointed out that additional to general GP advantages like automatic feature selection and function derivation, two-stage GP provided intelligence rules that can be independently used by a decision maker.

Fogarty (2012) argues that GAs produce results that are only slightly better than traditional techniques but have little additional utilization in the industry and are thwarted by transparency legislation and regulation. Although he briefly acknowledges advantages of GAs stated by other researchers, he fails in including those advantages into his overall assessment of GAs. Therefore, he employed Genetic Algorithms (GA) as a maintenance system for existing scoring systems instead of using GAs for model development. Maintaining credit scoring models becomes necessary once a shift in population leads to deterioration of the models' discriminative capabilities. Fogartys approach included the employment of GAs on 25 models that were originally developed using logistic regression. Models and data sets were provided by a unnamed finance company. The performances of the resulting models were compared to the source models. Because legislative and regulatory constraints still hold for revised models, better or similar performing models were only marked for redevelopment. The author found that 18 models performed significantly better than their source model, compared to 4 models that performed equally good. In 3 cases did the models not perform satisfactory due to overfitting the data and no model performed worse than its source model. The author concluded that GAs are a useful approach as a credit scoring maintenance system.

One of the major criticism of Genetic Algorithms and figuratively of Genetic Programming is their poor comprehensibility (Goldberg, 1989). Hoffmann et al. (2002) and extending Hoffmann et al. (2007) remark that research in credit scoring usually focuses on developing models with high predictive accuracy but omits the reasoning for classifications. In both paper they employ a genetic fuzzy rule based system and add a boosting mechanism to generate a set of fuzzy classification rules. However, the approach in Hoffmann et al. (2002) is based on approximate fuzzy rules while in Hoffmann et al. (2007) they extend previous work by using a descriptive fuzzy system. The major difference in approximate and descrip-

tive fuzzy system is that in the latter fuzzy rules are associated with linguistic concepts, which makes them more intuitive, more humanly understandable and increases the comprehensibility considerably. Hoffmann et al. (2002) use data from a major Benelux financial institution and Hoffmann et al. (2007) additionally uses the publicly available Australian and German data sets as well as data about breast cancer and diabetes. The evaluation measure in both papers is the accuracy. Hoffmann et al. (2007) conclude that both approximate and descriptive fuzzy rules yield comparable results. They reason that the boosting algorithm employed compensates for weaknesses in the individual rules and hence evens out the results. Comparing their approaches with other techniques shows no clear indication of a superior method. Results are either indistinguishable or the superior method depends on the data set. Hoffmann et al. (2007) remark an important trade-off between the fuzzy rule systems. The approximate fuzzy rules yield few rules but with low comprehensibility while the descriptive fuzzy rules yield a larger rulebase with more intuitive linguistic rules. Their works provide one of several useful approaches for the extraction of humanly readable rules that provide explanations of how decisions have been made (Goldberg, 1989). Subsequently, their approach closes the gap in the characteristics of computational intelligence systems stated by Marques et al. (2013).

The literature review indicates a severe lack of data in credit scoring research. All but two papers use the German and Australian data sets for the model development, with the German data set being around since 1997 and the Australian data since 1987. This observation holds for the whole research field of credit scoring and hence Lahsasna et al. (2010) calls for more cooperation between academic researchers and financial institutions. Furthermore, the literature indicates a clear focus on financial institutions but neglect of other areas by research in credit scoring. Finally, the results of different credit scoring techniques employed in the literature consistently do not differ significantly. Even more sophisticated methods like the two-stage GP model of Huang et al. (2006) or the combined model of Zhang et al. (2007) show little increase in performance. While these models use the same data set which could explain the similar results, Abdou (2009) observes the same effect on a data set from Egyptian commercial public sector banks. The reason may be found in a combination of the high analyst domain knowledge as stated by Fogarty (2012) and the flat maximum effect, i.e. different approaches lead to relatively similar results as first described by Lovie and Lovie (1986). All three points are addressed in this work. Not the German nor the Australian data set is used for model development but a real world data set. The data set contains order requests from e-commerce vendors and is provided by AFS. Based on this data set, different models are developed and compared. Hence, data set specific limitations that may have negatively effected research on the German and Australian data set can be ruled out.

3. RISK SOLUTION SERVICE

Risk solution service (rss) is a risk management service that aims at covering the whole order process of ecommerce retailers' customers.

Its objectives are threefold. First, increase of conversion rate and customers' retention in the E-Shop by improving differentiation and managing of payment methods. Next, enhance cost control by providing innovative pricing models and configurable standard solutions in different service levels. Last, improve discriminatory power by combining current and historical customer information.

The rss system works as shown in figure 3. Customer actions are registered by the client system and passed on to the rss backend. Depending on the customer action, service calls are triggered as depicted in figure 4 and explained below. The data content of the service calls is passed on to the ASP platform that carries out the scoring of the customer. The ASP platform saves the request data on a logging database from where it is passed on to an archive and a reporting database. From the latter, the data is passed on to the data warehouse and merged with additional information from an extra rss database. The reporting system in the portal obtains the data from the data warehouse. Additionally, the client specific configuration is retained on the portal for easy access by the client. On every data transfer, the data is reduced and partly transformed. Hence, the data in the data warehouse does not entirely match the data used in the service calls.

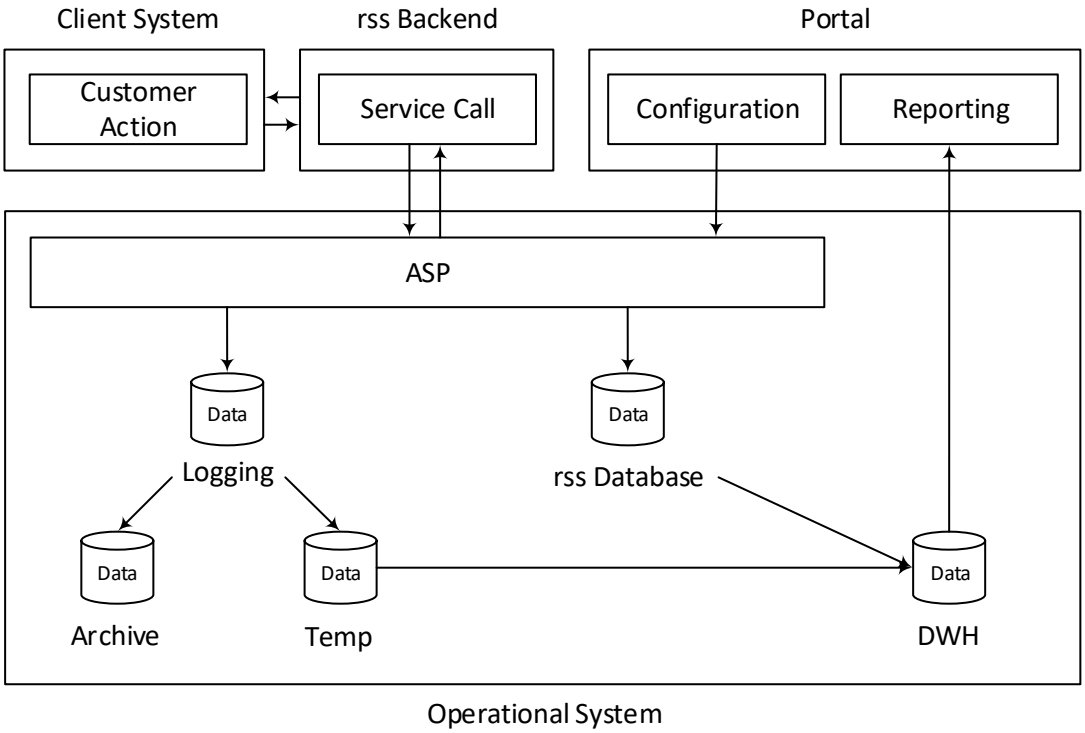

Figure 3: rss System

Rss consists of a number of different risk management services whose usage depends on the current customers' order process stage, as mapped in figure 4. Before the order process starts, upon registration of the clients' customer, rss offers an address check. The account check verifies the validity of the entered address data by:

1. validation of correct postal syntax
2. verification and if necessary correction of mail address
3. verification of deliverability by checking against a list of known addresses

Once the customer placed the order with the client, a risk check deploys and returns a risk assessment to the client. Based on the risk assessment, the client offers his customers selected payment methods from which the customer may choose whichever he prefers. In case the customer chooses direct debit, an account check deploys and checks the customers' submitted account data. The account check verifies the validity of the entered account data by:

1. validation of correct syntax
2. comparison with whitelist of existing bank identification numbers
3. comparison with blacklist of publicly visible bank accounts, e.g. public institutions or charity organizations

In case the customer is accepted an order confirmation is transmitted and the order gets registered in the rss system.

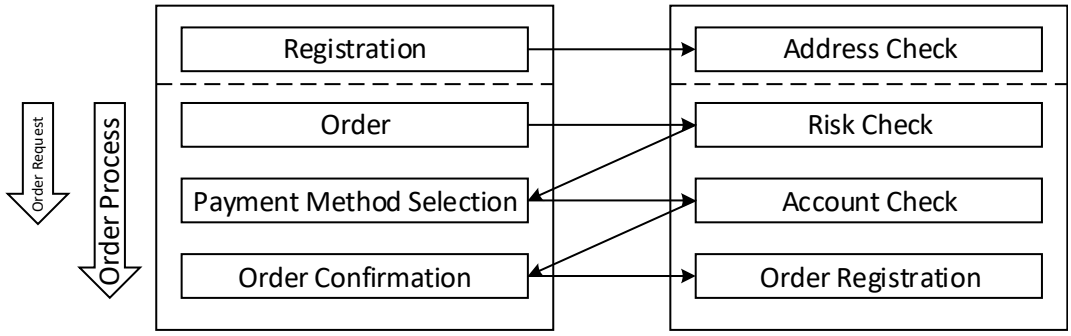

Figure 4: rss Risk Management Services

Rss is designed to work not only with the existing scoring system AFS provides but also with scoring systems of other providers. Hence, in order to ensure operationally in absence of the AFS credit agency score, the Risk Check was split into two modules that work in combination and on its own. The risk check consists of the pre risk check and the main risk check, as depicted in figure 5. The former being a combination of a pre check and a pre score and the latter calling different credit agency scoring systems. Added together, they compose the rss score.

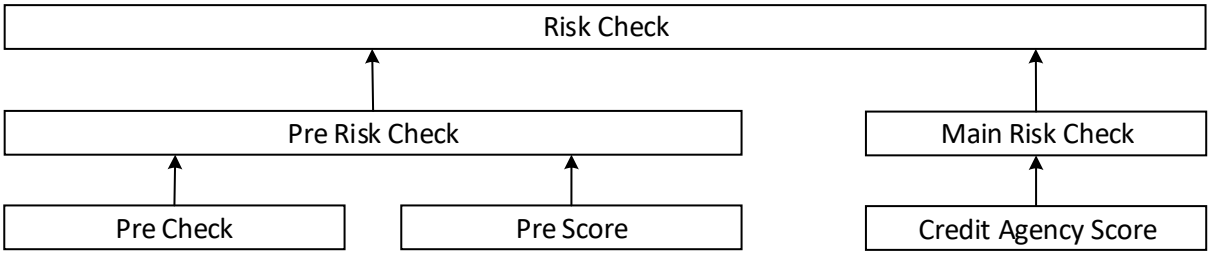

Figure 5: Risk Check

The rss pre risk check was developed in 2011 and a vast number of ad hoc updates were deployed over

the years. However, neither the original design nor any update was based on a statistically sound model. Instead, a generic scorecard was developed, which is generally done whenever data is unavailable.

The pre risk check consists of a pre check and a pre score. The former is a combination of exclusion rules that lead to the rejection of a request if triggered or otherwise to a pass on to the pre score, which assigns a score according to the request data. The latter is centered on 0 in order to work as an "on top" addition or deduction to the credit agency score. The pre check matches requests with a blacklist, compares the request basket value to a predetermined limit and employs a spelling and syntax review of the request input data. Around 30% of requests were rejected by the pre check and subsequently were not assigned a pre score.

The pre score is based on a score card that distributes score points for a number of distinct features. Within a certain score range, the assigned score triggers one of three possible actions as depicted in figure 6. Below a certain threshold the customer is rejected and above a certain threshold the customer is accepted. Between those threshold the customer is passed on to the main risk check for further evaluation.

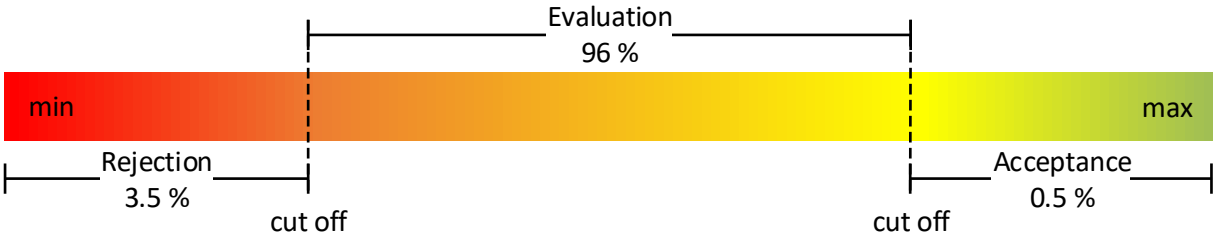

Figure 6: Prescore Behavior in Score Range

Taking into account that the pre risk check was designed to be operational both in combination with and without the main risk check, the very design of the pre risk score seem to incorporate a critical flaw. The requests that are passed on for further evaluation require the main risk check or otherwise those requests have to either be accepted or rejected. However, around 96% of the requests evaluated by the pre score are passed on to the credit agency score, around 0.5% of the requests were accepted and 3.5% of the requests were rejected. Therefore, the overwhelming majority of requests require additional evaluation.

Mapping of the distribution as well as the average default rate to their corresponding pre score ranges in figure 7 shows the deficit in discriminatory power of the pre score. The score values are not spread apart over the score ranges but heavily clumped together in few score groups that contain a majority of observations. Additionally, both good and bad requests center around a similar score range, with the number of good requests being continuously higher than the number of bad requests. However, the default rate does show a difference over the score ranges. Specifically, the default rate is higher in the rejected range than in the range up for further evaluation and the default rate in the latter is higher than in the accepted range.
Both the rejection and the acceptance area have a negative trend but the area up for further evaluations has a positive trend. Accordingly, the default rate increases with increasing score values between the threshold score for rejection and acceptance while increasing score values are supposed to map the decreasing probability of default. As a result, the pre score lowers the discriminatory power of the credit agency score it is merged with, assuming the latter works properly.

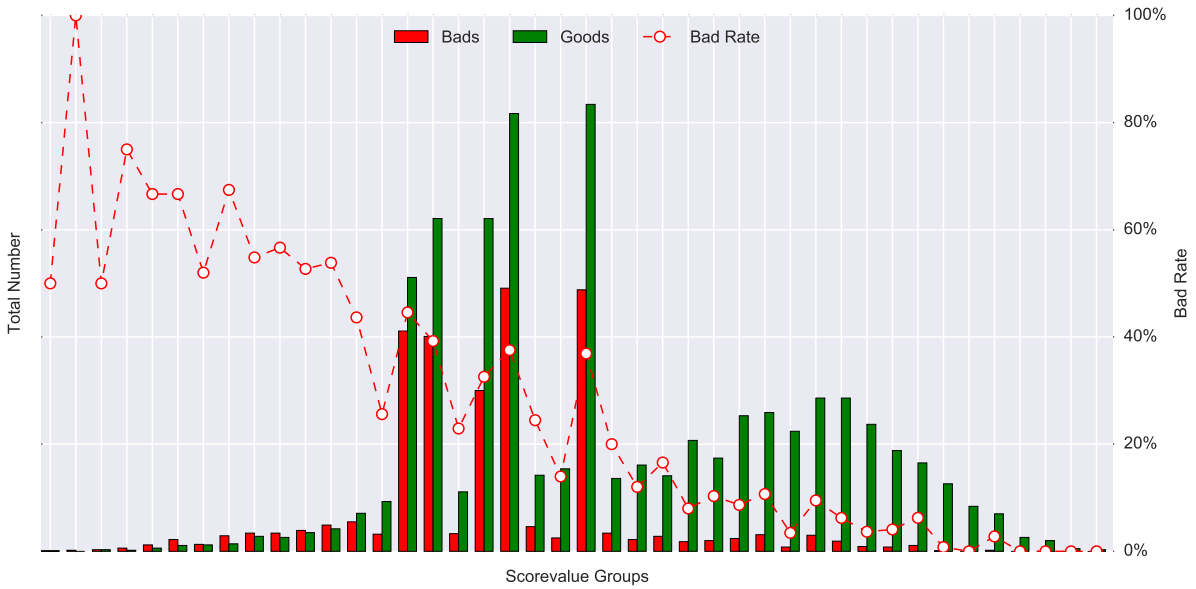

Figure 7: Prescore Distribution of Goods and Bads

The rss pre risk check provides an important part of the rss system. It is designed to work both in combination with a credit agency score within the main risk check and on its own. However, the current implementation reduces the discriminatory power of the main risk check and offers little benefit on its own.

The objective of this work is to revise the pre risk check and to incorporate both pre check and pre score into a single score with a higher discriminative power than the existing pre score. Additionally, the score needs to be operational on its own for usage in an international setting and in combination with a credit agency score.

4. RESEARCH METHODOLOGY

The dataset used in this work consists of order requests processed by rss and is provided by AFS. It is cleaned up before the remaining 56 669 order requests are subject to a stratified random split into a training set with 31 669 ($\approx$ 56%) observations, a validation set with 15 000 ($\approx$ 26%) observations and a test set with 10 000 ($\approx$ 18%) observations. The cleaning is needed because the data set includes variables and observations that cannot be used during modeling. For example, many variables in the data set are meta data variables used by the system and some observations are logged for testing purposes. The data removed during cleansing does not incorporate any meaning in a predictive setting. The size of training and test set are determined by taking into consideration the subsequent calibration phase for which the validation set is used. Hence, the validation set accounts for 15 000 observations. To ensure a minimum number of observations in small score groups, 10 000 observations are retained in the test set. The slightly crooked proportion percentages are a result of the visually more pleasing decision to split the data set by absolute numbers.

The training set is fed into the GP modeling processes and used to learn the best discriminant function as explained in section 4.2. The process of GP is employed 5 times and the final output of every run is validated using the validation set. Section 4.3 explains the best discriminant function of the 5 runs.

The resulting scores do not represent an accurate estimate of the probability. In credit scoring, people are sorted against probability of default and accepted or rejected according to their relative position in respect to a threshold. Calibrated scores allow to rank the score values and to interpret their differences. With uncalibrated scores, a better score value may inhibit a higher default in payment. Or little difference between score values correspond to a big difference in probability of default for some score values but to little difference in probability of default for others. Hence, the results are calibrated using Isotonic Regression as explained in section 4.4. In order to validate the performance of Isotonic Regression, stratified k-fold cross-validation is applied on the calibration training set, which is previously used as validation set. The folds are stratified in order to retain the bad rate over all folds. Following Kirschen et al. (2000), k is chosen to be 10 to keep bias and variance low while containing a reasonable number of observations in every fold. In the process of cross-validation the data set is divided into 10 non-overlapping subsets. In every iteration 9 subsets are allocated to train the model and the remaining subset is used to test the model. The process is repeated 10 times, and thus every observation is in the test set once and 9 times in the training set. The estimates over all iterations are averaged for the output estimates (Seni and Elder, 2010). The test set is never subject to any learning but only to model application.

In section 5 the calibrated results are used to analyze the discriminatory power of the GP model on its own and in combination with the already existing credit agency score of AFS, named ABIC. Additionally, models using Logistic Regression, Support Vector Machines and Boosted Trees are developed as comparison for Genetic Programming. All models are developed using the Python language in the version 2.7, with the modules DEAP for GP and scikit-learn for Logistic Regression, Support Vector Machines and Boosted Trees (Fortin et al., 2012; Pedregosa et al., 2011).

The binary response variable represents a default in payment by the customer or potential default in payment for order requests that have been declined. §286 (3) BGB (Germany) defines delay of payment as the non-settling of bills within a 30-days period (assuming no other payment target between

vendor and customer is contractually scheduled). §178 (1b) in Regulation (EU) No 575/2013 considers a default as occurred once an obligor is past due more than 90 days. Hence, a customer who did not settle his account within a period of three months upon receipt is considered as a default in payment. This approach renders the introduction of an intermediate class of customers whose status is unclear unnecessary. Potential default in payment is defined as follows: Individuals whose order requests were declined but were officially reported insolvent during the period of observation. Hence, information about the performance of rejected applicants is available and subsequently the respective bias in data is dispelled and no further reject inference necessary (Thomas, 2000; Hand and Henley, 1997).

This chapter is organized as follows. The dataset used with the preprocessing steps taken and variables used in the model is explained in section 4.1. Next, section 4.2 explains Genetic Programming in detail before its application is depicted in section 4.3. Finally, the process of calibrating the GP results is explained in 4.4. The overall work flow is depicted in figure 8.

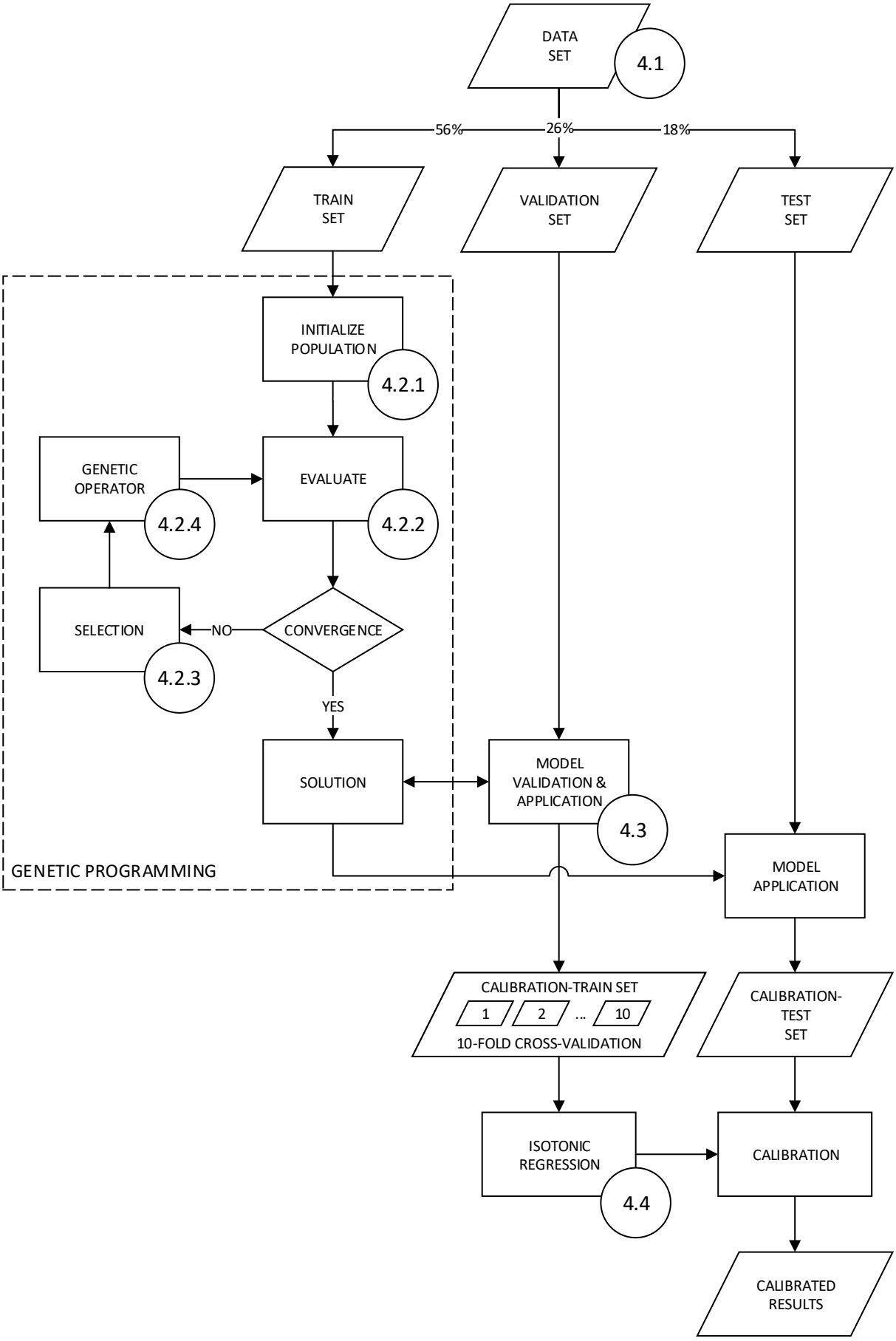

Figure 8: Overall Work Flow

4.1 Dataset

The original dataset consists of 56 669 orders processed by rss between 2014-10-01 and 2015-12-31 with 19 variables. The data set is obtained from the data warehouse (see figure 3). Hence, there are differences between the data used during service calls and the data subsequently used for model development. Most notably, some variables that are assumed to incorporate high discriminatory power are unavailable for model development.

Data in Credit Scoring includes personal information about the individuals processed. To account for the sensitivity of the data, that is to protect the confidentiality and to concur with data protection regulation, alterations are conducted. Hence, the variables are renamed to IN0 - IN18 and their meaning is only explained briefly and in a general sense. Additionally, the data set is weighted to incorporate more observations who turned out to be bad than in the real population. A benefit of the weighted data set is that because credit scoring mainly aims at the identification of bads in the population, it is important to include an adequate volume of bads to capture their structure in the model development process. The bads are represented by a performance measure, with 15 335 ($\approx 27\%$) defaults on payment.

Brief explanation of the variables used

- IN0
  Binary variable with information whether a customer is known to the client.
- IN1
  Binary variable with information whether shipping address and billing address match.
- IN2
  Elapsed time in days since the customer was registered by the system for the first time.
- IN3
  Validation of the billing address, i.e. whether the address exists and a customer is known to live at the stated address
- IN4 - IN5; IN10; IN11; IN13 - IN15; IN18
  Information about the order history of a customer
- IN6 - IN8
  Information of the dunning history of a customer.
- IN9
  Discretized order value in relation to the clients' average order value.
- IN12
  This variable is designed as means of fraud prevention. A fraudulent customer may employ a couple of orders with little order value to be known by the system as a good customer and subsequently employ expensive orders using insecure payment methods. Hence, the order value is compared to the average of past orders by the customer.
- IN16
  Clock hour of order time.
- IN17
  Age of customer in years.

In order to efficiently obtain the discriminant function the 13 continuous variables need to be discretized (Ong et al., 2005). The discretization is done using the Weight of Evidence (WoE) measure to assess the predictive power of each attribute. The WoE measures the difference between the distribution of goods and bads and therefore represents an attributes' power in separating good and bad observations (Siddiqi, 2006). The WoE is frequently used in credit scoring to convert a continuous variable into a discrete variable and as such the used method at AFS.

The WoE is calculated as follows:

$$\text{WoE} = \left[\ln\left(\frac{\text{Distribution Good}}{\text{Distribution Bad}}\right)\right] \times 100 \tag{1}$$

A bin with a WoE around zero has the same probability of observing a default in payment as the sample average. Contrarily, if a bin has a WoE above or below zero, the probability of observing a default in payment is above or below sample average respectively. However, more important than the absolute WoE is the difference between bins. The variables predictive power relates to the difference in WoE between bins. Monotone bins offer a direction of effect over the bins. With monotone increasing WoE the probability of default is always higher in a greater bin and vice versa with monotone decreasing bins (Siddiqi, 2006). The interval boundaries of the bins are selected such that the bins are monotone increasing or decreasing, the differences in WoE over the bins are roughly equal, the bins are as evenly sized as possible and a minimal number of observations is maintained in each bin. Missing values are not submitted to the bin selection phase but treated as a separate bin exhibiting a value of -1. The reasoning for treating missing data this way is to account for its predictive value. Table 1 shows the discretization of continuous variables with the value ranges for the respective bins. Figure 9 shows the

Table 1: Discretization of Continuous Variables

Variable	-1	1	2	3	4
IN2	∅	$[-\infty, 90)$	$[90, 380)$	$[380, 600)$	$[600, \infty)$
IN9	∅	$[-\infty, 150)$	$[150, 300)$	$[300, \infty)$	
IN17	∅	$[-\infty, 25)$	$[25, 30)$	$[30, 40)$	$[40, \infty)$
IN14	∅	$[-\infty, 18)$	$[18, \infty)$		
IN15	∅	$(\infty, 7)$	$[7, 4)$	$[4, 2)$	$[2, -\infty)$
IN13	∅	$[-\infty, 1)$	$[1, 4)$	$[4, \infty)$	
IN4	∅	$[-\infty, 1)$	$[1, 4)$	$[4, 10)$	$[10, \infty)$
IN5	∅	$[-\infty, \infty)$			
IN10	∅	$[-\infty, 5)$	$[5, 20)$	$[20, \infty)$	
IN11	∅	$(\infty, 6)$	$[6, -\infty)$		
IN12	∅	$(\infty, 50)$	$[50, 25)$	$[25, 10)$	$[10, -\infty)$
IN16	∅	$[-\infty, 9)$	$[9, 14)$	$[14, \infty)$	

discretized variables with their WoE values for every bin. The variables are ordered decreasing by their Information Value (IV). The IV is a measure of how well the binary response in a performance variable is being distinguished by the variable. The lower the IV, the lower the predictive power of a variable. As a rule of thumb, variables with an IV of less than 0.05 add little meaningful predictive power to the model (Henley, 1995; Hand and Henley, 1997).

The IV is calculated as follows:

$$\text{IV} = \sum_i \left(\text{Distribution Goods}_i - \text{Distribution Bads}_i\right) \times \ln\left(\frac{\text{Distribution Goods}_i}{\text{Distribution Bads}_i}\right) \tag{2}$$

From the 12 discretized variables, 3 have an IV of less than 0.05, namely 'IN9', 'IN16' and 'IN5'. Nonetheless, they are not removed from the data set, because while they seem insignificant by themselves, they might become more important in interaction with others. Furthermore, GP selects the important variables automatically (Ong et al., 2005). The final dataset is shown in Table 2.

Table 2: Input Variables

Variable	Type	Min	Mean	Max	Std
IN0	Bool	0	0.47	1	0.50
IN1	Bool	0	0.05	1	0.22
IN2	Categorical	-1	1.13	4	1.83
IN3	Categorical	-1	-0.48	2	1.06
IN4	Categorical	-1	1.25	4	1.90
IN5	Categorical	-1	-0.83	1	0.56
IN6	Categorical	0	0.26	4	0.62
IN7	Categorical	-1	-0.41	3	1.14
IN8	Categorical	-1	1.08	3	2.00
IN9	Categorical	-1	1.32	4	0.65
IN10	Categorical	-1	0.46	3	1.53
IN11	Categorical	-1	-0.12	2	1.29
IN12	Categorical	-1	0.85	4	1.98
IN13	Categorical	-1	0.50	3	1.64
IN14	Categorical	-1	0.09	2	1.14
IN15	Categorical	-1	0.21	4	1.88
IN16	Categorical	-1	2.43	3	0.80
IN17	Categorical	-1	1.56	4	2.13
IN18	Categorical	-1	-0.50	2	0.59

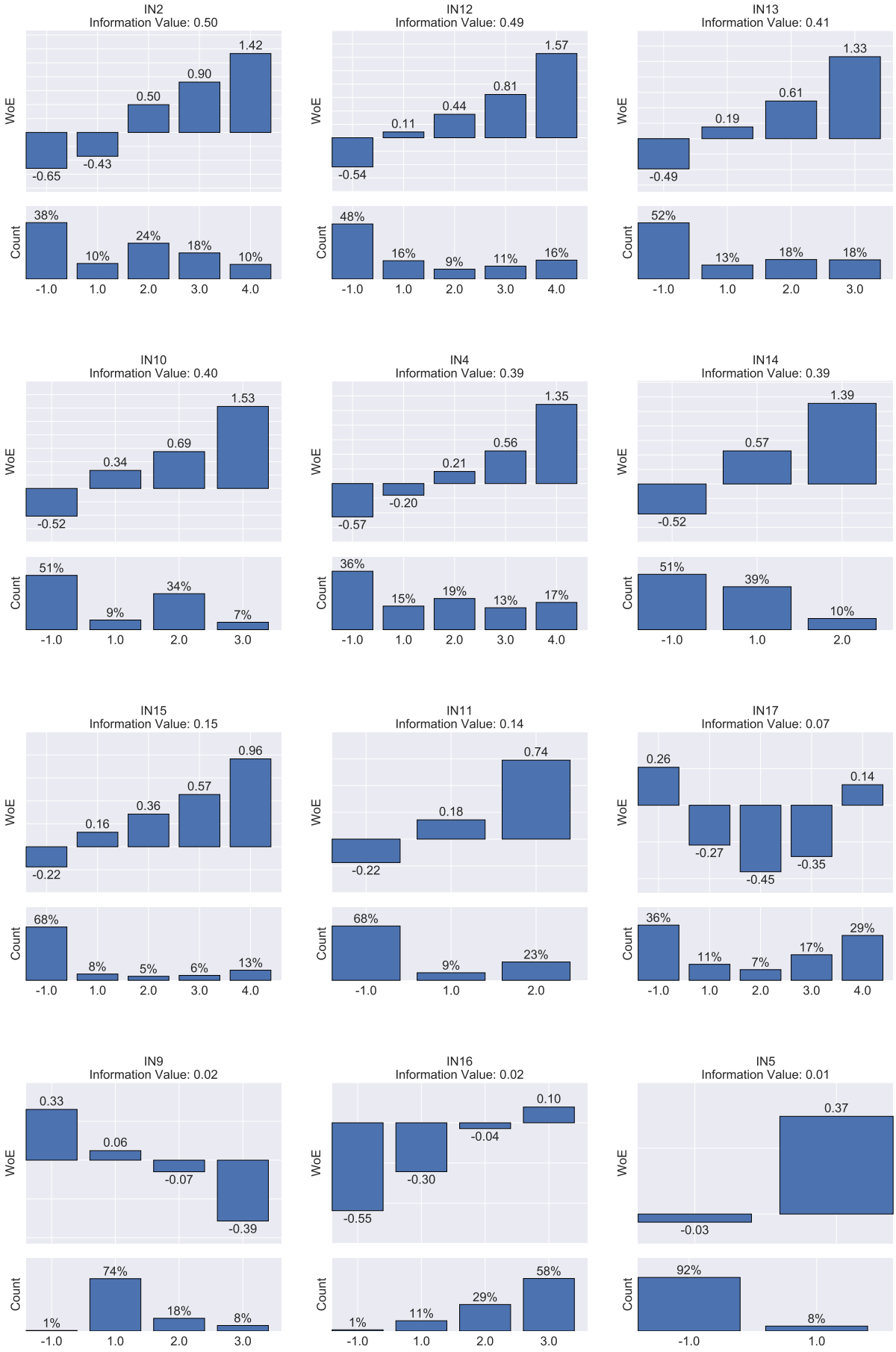

Figure 9: Discretization Plot

4.2 Genetic Programming

Genetic Programming (GP) is an optimization method in the research field of Evolutionary Computation, that adopts the Darwinian principles. Next to GP, the field includes a number of different methods among which are Genetic Algorithms, Evolution Strategies, and Evolutionary Programming (Eiben and Schoenauer, 2002; Eiben and Smith, 2015).

Genetic Programming (GP) is a domain independent evolutionary computation method that automatically finds a solution in a predefined search space. This is done by creating a population of computer programs that is stochastically transformed over time into a new population of computer programs by employing evolutionary mechanisms such as inheritance, selection, crossover and mutation, that are inspired by Darwin's theory about evolution. GP computer programs are usually depicted in a tree representation, with variables and constants as leaves of the tree and arithmetic operations as internal nodes. In the GP setting, the former are called terminals and the latter functions. Together they form the primitive set. All computer programs that can be constructed by a primitive set define the search space of possible solutions (Koza, 1992). For illustration, figure 10 represents the syntax tree of the program $(var1 + var2) * 2$, with $var1$, $var2$ and 2 as terminals and $+$ and $*$ as functions. In credit scoring,

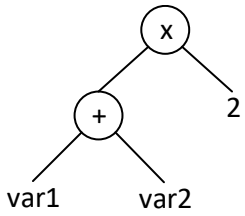

Figure 10: Example Syntax Tree

these computer programs are discriminative functions that aim at assigning the associated probability of default to a customer. Hence, the problem is defined as a symbolic regression problem. Based on its output value, an appropriate threshold value can be chosen to classify customers into bads and goods. In this work, the function set consists of arithmetic operators, equality operators, relational operators and logical operators. Constants are included as terminals according to the values in the variable ranges from -1 to 4. Additionally, in order to allow for operations that decrease the impact of parameter values, decimal point constants between 0 and 1 are included as well. The GP parameter settings are shown in Table 3 and thoroughly explained below.

Table 3: GP Parameter Settings

Parameter	Value
Population Size	300
Initialization	Ramped Half-and-Half
Fitness Function	Area Under the ROC Curve
Function Set	$\{+, -, \times, \div, \leq, \geq, =, \neq, if - then - else\}$
Terminal Set	$\{Attributes, \frac{p}{10} \mid p \in \mathbb{N} \wedge p < 10 \cup x \in \mathbb{Z} \wedge -1 \leq x \leq 4\}$
Maximum Number of Generations	500
Selection	Tournament
Crossover Rate	0.9
Mutation Rate	0.1

The process of GP is as follows. First, an initial population is created (Subsection 4.2.1). Next, GP enters the evolving process, starting with the evaluation of the population (Subsection 4.2.2). Based on the evaluation, functions are selected from the population (Subsection 4.2.3). Finally, the selected population is evolved using genetic operators (Subsection 4.2.4). The evolving process repeats until a satisfactory function is evolved or a maximum number of 500 generations have passed. The process of GP is depicted in figure 11 and thoroughly explain below.

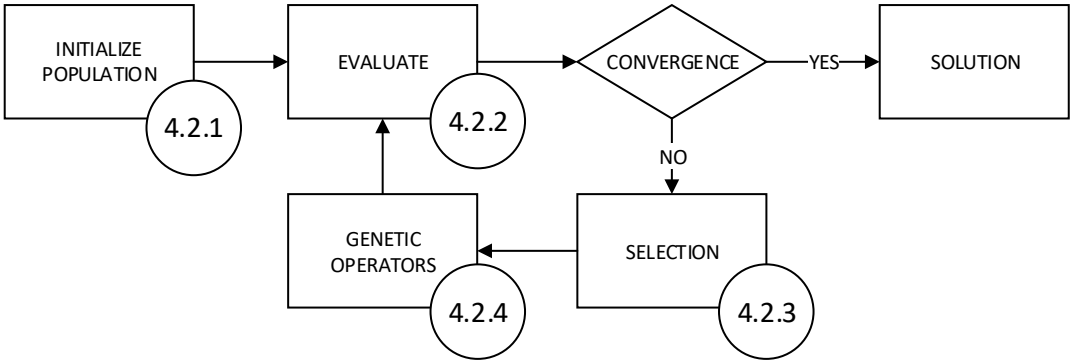

Figure 11: GP Process

4.2.1 Initialization

The initialization process in GP creates a population of random functions. Used in this work is one of the more popular approaches called ramped half-and-half, a hybrid technique that creates half of the population using an algorithm called full and the other half using an algorithm called grow. Full creates a population of trees with all leaves at the same depth and grow creates trees with leaves at possibly different depths. The depth of a tree is defined by the depth of the deepest leaf. In both full and grow the maximum tree depth is pre defined. For tree construction using the full method nodes are randomly chosen from the function set until the maximum tree depth is reached. The trees are then closed by randomly chosen terminals. Contrarily, in the grow method nodes are chosen randomly from both function and terminal set until the maximum tree depth is reached. Subtrees with a function as last leaf are closed of by randomly chosen terminals. Because neither full nor grow make for a high population variety they are combined into one (”half-and-half”) and the dept limit is varied during initialization (”ramped”) (Koza, 1992). One drawback of the initializing methods presented above is that size and shape of the trees are strongly affected by the ratio of number of functions to number of terminals. In this work, the number of functions (9) is lower than the number of terminals (34), which might lead to short trees (Poli et al., 2008). To counter this problem, many different initialization mechanisms have been introduced. For example, Chellapilla (1997) uses the length of the tree instead of the depth and Langdon (1999) introduced a method called Ramped Uniform Initialization. Poli et al. (2008) states that more uniform initializations may be advantageous with asymmetric problems that have dominant variables. As shown in figure 9 in subsection 4.1 this is clearly the case in the data used. Nonetheless, the ramped half-and-half method is used in this work, because DEAP only supports full, grow and ramped half-and-half as presented above. The population is initialized with 300 functions.

4.2.2 Evaluation Criteria

A classification model maps observations to predicted classes. That is, a model predicts if an observation is good or bad. The GP model returns a continuous output to which different threshold, or cut-off points, can be applied in order to allocate the output to a class. Evaluating the prediction of an observation in respect to its true class provides four possible outcomes:

- True Positive - observations classified as bads whose true class membership is bads
- False Positive - observations classified as bads whose true class membership is goods
- True Negative - observations classified as goods whose true class membership is goods
- False Negative - observations classified as goods whose true class membership is bads

Commonly, those outcomes are disposed in a confusion matrix (also called a contingency table), as depicted in figure 12. The outcomes with a 'true' in their names, that is True Positive and True Negative', represent correct classification and the outcomes with a 'false' in their names, that is False Positive and False Negative', represent incorrect classification. From the confusion matrix a number of metrics can be calculated. Some of those metrics are shown in equations 3 to 6 and explained below.

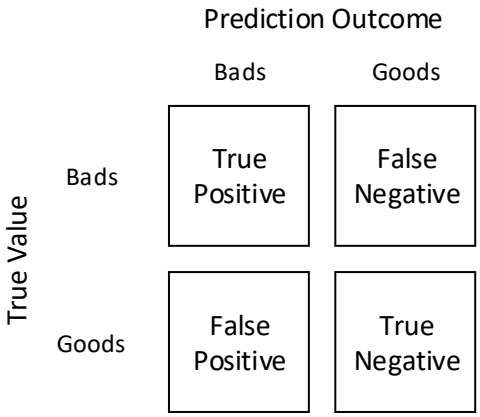

Figure 12: Confusion Matrix

$$\text{True Positive Rate (TPR)} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{3}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positive}}{\text{False Positive} + \text{True Negative}} \tag{4}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{5}$$

$$\text{Accuracy (ACC)} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive + False Negative + False Positive + True Negative}} \tag{6}$$

In order to evaluate how accurate a GP model predicts the outcome, a performance measure called fitness function is set-up. The fitness function gives feed back about which models are supposed to survive and which are to perish (Hilas et al., 2014). Among the performance measures regularly employed as fitness function is the area under the ROC curve (ROC-AUC), the single-scalar representation of the receiver operating characteristic (ROC) curve (Abdou and Pointon, 2011).

The ROC curve is a two-dimensional graph that quantifies the performance of a classifier in reference to all thresholds (Yang et al., 2004). For this purpose, the relative frequency of every unique score value specified by the True Positive rate (TPR), i.e. the proportion of actual positives predicted as positive as depicted in equation 3, and the False Positive rate (FPR), i.e. the proportion of actual negatives predicted as positive as depicted in equation 4, is plotted on the X and Y axis respectively (Baesens et al., 2003). Therefore, the ROC graph shows relative trade-offs between benefits and costs.

Figure 13 depicts an example representation for a ROC curve. The point in the lower left corner depicts a threshold that classifies all observations as goods. Such a classification induces no false positive error, but no true positives either. Contrarily, the point on the upper right corner corresponds to a threshold that classifies all observations as bads. Hence, both false positive and true positive are at their maximum value. A perfect classifier incorporates a point in the upper left corner that classifies bads as bads and goods as goods. In general, a point with higher TPR and lower FPR compared to another is said to be better (Fawcett, 2003). In order to compare different models the ROC curve can be reduced

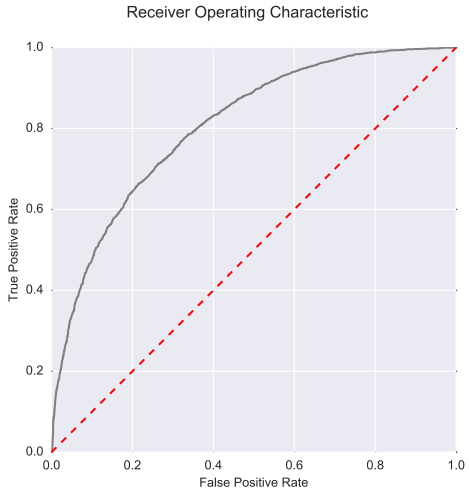

Figure 13: Example ROC curve

to its single-scalar representation by calculating the ROC-AUC using the equation in 7. Because ROC-AUC takes the whole area under the ROC curve into account, it is possible for a classifier to be better than another in a specific region but have a worse ROC-AUC (Fawcett, 2003). The ROC-AUC represents the probability that a randomly chosen positive observation is ranked higher than a randomly chosen negative observation, which corresponds to the quantity of a Wilcoxon test (Hanley and McNeil, 1982; Baesens et al., 2003).

$$\text{Area under the ROC curve (ROC-AUC)} = \int_{-\infty}^{\infty} \text{TPR(T)FPR}'\text{(T)}\, \text{dT} \tag{7}$$

where T in (7) is a threshold parameter.

Because the ROC space is a unit square, the ROC-AUC value range is between 0 and 1. However, a random classifier produces a diagonal line that splits the ROC space into two equal triangles with an area of 0.5. Therefore, a classifier below 0.5 performs worse than a random classifier. But because the ROC space is symmetrical about the diagonal line of the random classifier, negating a classifier below 0.5 produces its counterpart above 0.5. Overall, the ROC-AUC value ranges between 0.5 for an random classifier and 1 for a perfect classifier.

Since with advancing generations models grow in size until they reach a point from which it becomes increasingly difficult for humans to read and understand, measures to control model size have to be implemented. First, the models are being pruned once they reach a certain size. Second, the fitness is being expanded to a multi layered fitness by incorporating a secondary fitness on basis of model size. However, because ROC-AUC is the primary fitness function, only models with equal ROC-AUC are being compared on their size (Eggermont et al., 2004). In addition to keeping models small, size as fitness keeps variables out that have no effect on the models' predictive power. Without size as fitness, models with attributes that have neither positive nor negative effect on the models' predictive power have the same chance of survival as the models without. Contrarily, if size is implemented as fitness, models with attributes that have no predictive power have a lower chance of survival than models without. Therefore, size as fitness does not only keep models small but reduces the number of low performing attributes.

4.2.3 Selection

After initializing the population of solutions and assigning a fitness value to them, the evolutionary process of creating a new population of solutions can be started. In a first step solutions are being selected to be submitted to subsequent processing. The selected solutions represent the basis for the functions in the next generation. They are probabilistically selected from the population on the basis of their fitness, i.e. the higher the fitness of a function, the higher the probability that it gets selected. As a result, solutions with high fitness values have a higher chance of survival and propagation of their genetic material. The degree of preference to which a better solution is selected is called selection pressure. The selection pressure impacts the convergence rate, i.e. the rate of approaching an optimal solution. However, if the convergence rate becomes to high the search space is not explored thoroughly. In this case, the loss in diversity results in the premature convergence on a local optimum and prevents the evolutionary process of reaching the global optimum. On the other hand if the selection pressure is too low the time needed to converge to the global optimum increases which may lead to the stagnation of the evolutionary process.

Selection methods can be divided into proportionate-based and ordinal-based selection (Miller and Goldberg, 1996). Proportionate-based methods select solutions on the basis of their fitness, while in ordinal-based selection solutions are ranked according to their fitness and selected on the basis of their position in the ranking. The most well known selection algorithms are proportional selection, linear ranking selection and tournament selection and many more are presented in research.

Originally, selection was introduces by Holland (1975) with proportional selection. In this method, the probability with which a solution i is selected is proportionate to its fitness value, i.e.

$$p_i = \frac{f_i}{\sum_{k=1}^{N} f_k} \tag{8}$$

Despite its fame, proportional selection is well-known for having a low performance and to incorporate some serious disadvantages. Solutions with an outstanding high fitness value take over the population and narrow the search space down, resulting in premature convergence. This effect can regularly be observed at the begin of the evolutionary process when many solutions have a low fitness. Contrarily, some generations in the evolutionary process the selection pressure frequently collapses because the worst solutions have perished and the surviving solutions incorporate a similar fitness (Goldberg and

Deb, 1991; Blickle and Thiele, 1995a; Eiben and Smith, 2015). Grefenstette and Baker (1989) introduced linear ranking selection to overcome the drawbacks of proportional selection. In linear ranking selection the solutions are ranked by their fitness values, with the weakest solution ranked first and the strongest solution ranked last. Different ranks are distributed even for solutions with equal fitness. The selection probability is linearly assigned according to their rank, i.e.

$$p_i = \frac{1}{N}\left(2 - \eta^+ + \left(2\eta^+ - 2\right)\frac{i-1}{N-1}\right); i \in \{1, ..., N\} \tag{9}$$

η^+ is the selection bias to adjust the selection pressure, with $1 \leq \eta^+ \leq 2$ (Blickle and Thiele, 1995a). The selection pressure increases with increasing η^+.

The most common selection method, called tournament selection is being used in this work. Although tournament selection is shown to be superior in regards to many other selection methods including the ones mentioned above, the reasons for the continuing popularity are found in its inherit features (Blickle and Thiele, 1995b). Tournament selection is very efficient for both parallel and non-parallel systems and it does not require the population of solutions to be sorted. Thus, it has a linear time complexity of $O\left(N\right)$ (Fang and Li, 2010).

Goldberg and Deb (1991) attributes the origins of tournament selection to "unpublished work by Wetzel" that was cited by Brindle (1981). In tournament selection, a tournament is being held between a number of functions in which their fitness values are compared. The function with the highest fitness value is declared winner and inserted into a mating pool. The functions in the mating pool incorporate a higher average fitness value than the entire population (Koza, 1992; Poli et al., 2008; Goldberg and Deb, 1991). However, because tournament selection selects the functions to compete randomly, every function in the population has a chance of being selected, which lowers the chance of getting stuck in a local optimum, which is known as premature convergence (Eiben and Schoenauer, 2002). In the process of comparing fitness values between functions, tournament selection only takes into account if a function is better than another but not by how much. This holds the degree to how much the better functions are preferred constant (Poli et al., 2008). This effect is called selection pressure (Miller and Goldberg, 1995). The advantage is that extremely powerful functions do not lead to loss in diversity and subsequently premature convergence in an suboptimal solution. However, the convergence rate is lower compared to methods with higher selection pressure and marginal advantages in performance are enough to be selected, which might exclude functions with helpful subtrees (Miller and Goldberg, 1995; Poli et al., 2008). Another way to influence the selection pressure is by altering the size of a tournament. While tournaments are regularly held with pairs of functions, the size may be increased. Decreasing the size to 1 does not make sense, because it results in uniformly random selection. Increasing tournament size has an boosting effect on selection pressure because the probability that a high-fitness function is selected increases (Goldberg and Deb, 1991; Miller and Goldberg, 1995; Eiben and Smith, 2015). However, an increase of tournament size also increases the loss of genetic diversity. This work induces only low additional selection pressure by increasing the tournament size to include three functions.

4.2.4 Genetic Operators

The search process in GP is employed by evolving discriminant functions through the use of genetic operators. The objective is to alter the population of functions in order to find a new population of

functions with a higher discriminative power within the search space. The main genetic operators are called crossover and mutation.

The objective of crossover is to merge interesting but different information from two functions into a new function that combines these information. Hence, two discriminant functions from the population are selected and parts of their syntax tree is exchanged between them. Applied in this work is one-point crossover, in which a subtree from a selected function is replaced by a subtree from another selected function. The crossover point is randomly chosen, hence crossover is a stochastic operator (Eiben and Smith, 2015; Poli et al., 2008). While many other types of crossover are possible, DEAP only supports one-point crossover. Figure 14 depicts the genetic operators. Next to crossover, mutation is applied on

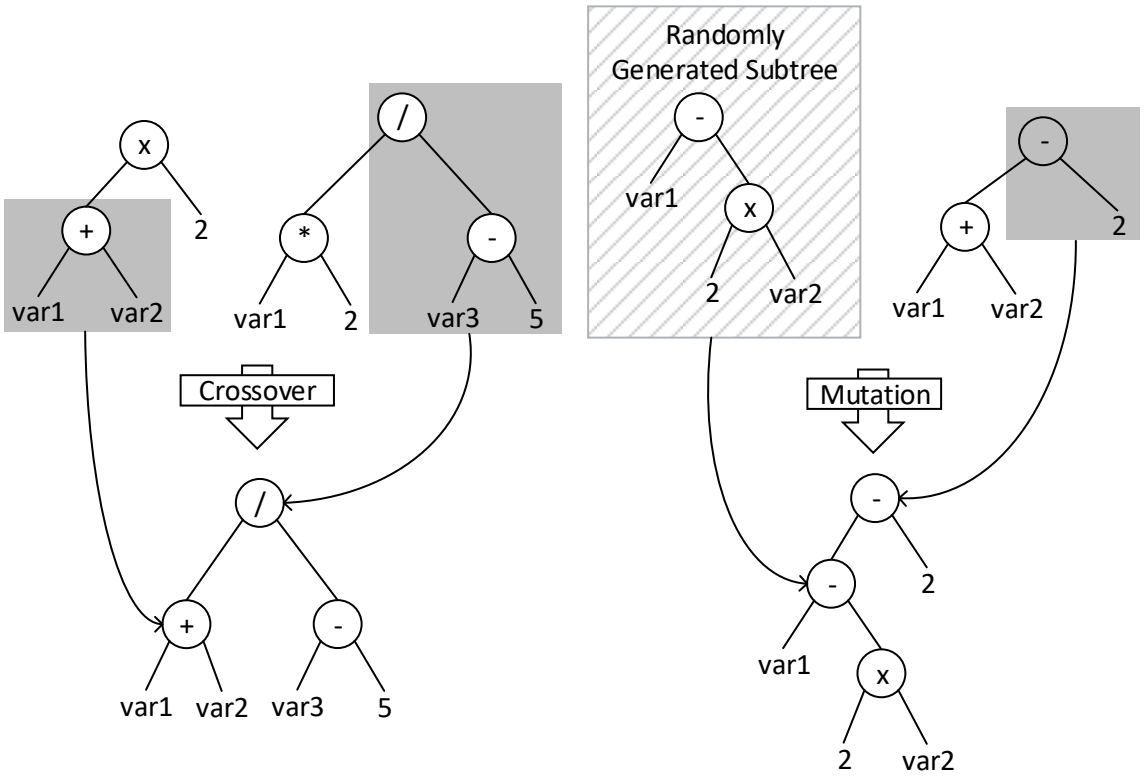

Figure 14: Genetic Operators

the population. Using mutation, a single discriminant function is randomly modified. Its objective is to maintain genetic diversity during the evolution. Mutation is particularly useful in cases where certain characteristics are needed, that are not contained in any solution in the population. During the evolutionary process it may happen that individuals carrying certain characteristics extinct. These particular characteristics may greatly improve the performance of some solutions in subsequent populations and might therefore be restored by mutation. Another example are characteristics that are not present in the initial solution and cannot be created by crossover. Mutation might provide these characteristics. In this work subtree mutation is applied in which a subtree is randomly chosen and replaced by a randomly generated subtree (Koza, 1992). Figure 14 depicts the process of crossover and mutation.

The last genetic operator is reproduction, which simply copies a parent function into the next-generation population. It is only employed in cases in which the crossover or mutation operator violate evolutionary constraints, for example size limitations.

Koza (1992) argued that the use of mutation in GP is of no avail and only crossover should be used. However, a number of studies showed that mutation does have utility and sometimes even outperformed crossover (White and Poulding, 2009; Angeline, 1997). While crossover and mutation can be used exclusively, they are not mutually exclusive and can be used in combination, which often times lead to better results (Luke and Spector, 1998). Hence, crossover is applied with 90% probability and mutation with 10% probability.

4.3 Applying Genetic Programming

Figure 15 shows the median fitness of the best functions in every generation over 5 runs during the evolutionary process on the training set as depicted in figure 11. The evolutionary process ran for 500 generations before being terminated. The fitness increases heavily for the first ≈200 generations. For the later ≈300 generations, fitness only increases marginally. The performances of the best functions of every run are validated on the validation set in order to verify their robustness.

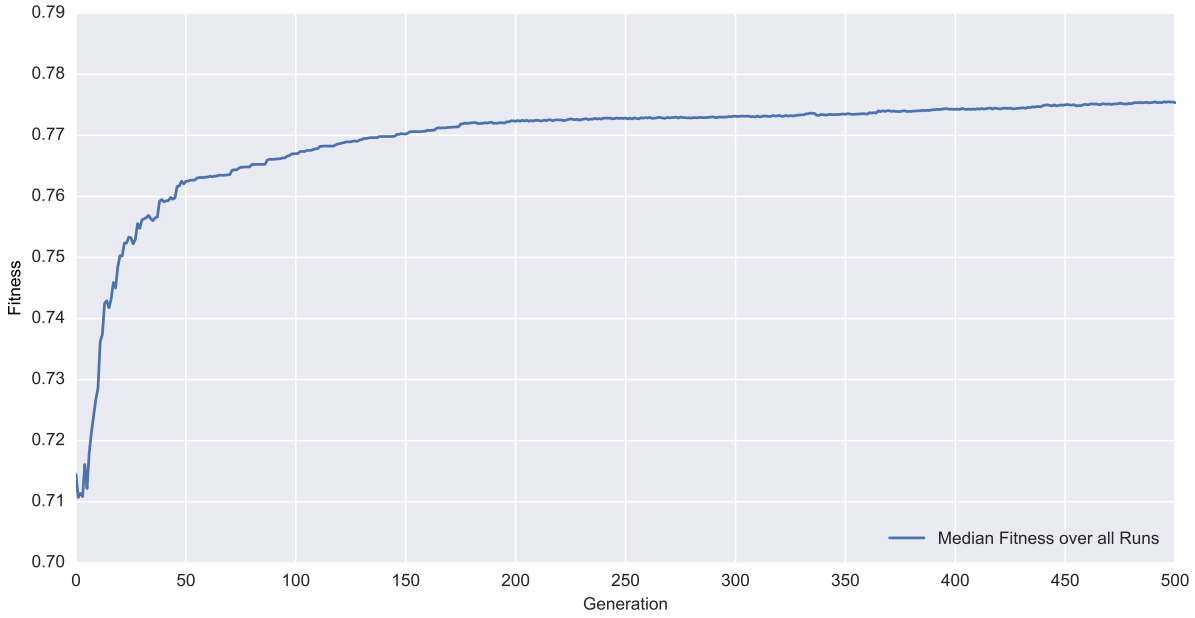

Figure 15: Median Fitness over all Runs

Figure 16 depicts the model with the highest fitness value after termination of the evolutionary process. The model has a bushy shape and a size of 220 nodes with a maximal tree depth equal to the depth limit of 17. Bushy models are frequently observed using ramped half-and-half as initialization method (Poli et al., 2008). The two subtrees from the root are very different sized, with 193 and 27 nodes.

Examining the tree shows that pruning can be employed at a number of nodes in order to reduce the size of the model. Pruning the model yields a size of 154 nodes and a maximal tree depth of 16 nodes. Most noticeable, the if-then-else nodes yield no utility but additional size, because every if-then-else node is connected to a boolean True/False decision terminal. Those nodes can be replaced by their subsequent node, i.e. $if - then - else\,(False, \text{IN12}, \text{IN16}) \Rightarrow$ IN16. Additionally, some subtrees are concluded by operations between constant terminals that can be reduced to a single constant terminal node, i.e. $\times\,(0.7, 0.3) \Rightarrow 0.21$. The resulting constant terminal nodes 0.12, 0.21 and 5 depict values that are not available in the terminal set, from which only the values 0.3, 0.5, 0.7 and 2.0 are used.

Missing values are encoded as the value -1 that is not used in the model. Thus, the model does not specifically account for missing values.

Because GP automatically chooses important variables, one may conclude that the number of occurrences gives information about the predictive power of a variable. From the 19 variables in the function set, six are not used in the model or used in combination with the if-then-else nodes on the unused side, which effectively removes them from the model. Those variables are IN1, which contains information whether shipping address and billing address match, as well as variables IN5, IN11, IN14, IN15 and IN18 that contain information about the order history of a customer. The remaining order history variables, IN10 and IN13, are incorporated into the model but only occur a single time in the model. Hence, order history provides little to nothing in additional discriminatory power.

Among the most often used variables are IN7 and IN6 with 13 and 9 occurrences. These variables, together with IN8 which only occurs twice, contain information about the dunning history of a customer. A logical conclusion is that customers with prior payment difficulties are more likely to default than those without. Despite the low occurrence of IN8, dunning history provides a great deal of discriminatory power. With 10 occurrences, another heavily used variable is IN2 which contains information about the number of elapsed days since a known customer was registered by the system for the first time. This puts emphasis on continuous business connections between client and customers. Customer who have been customer for a long time are less likely to default on payment than first-timers. Finally, the fraud-prevention variable IN12 and IN16, which contains the order time, occur 5 times. The remaining variables are used between 1 and 3 times. Table 4 shows the variables with their respective occurrences in decreasing order.

Subsequently, the function with the highest fitness is passed on to be calibrated as depicted in figure 8.

Table 4: Occurrences of Variables

Variable	Number of Occurrence
IN7	13
IN2	10
IN6	9
IN12; IN16	5
IN3; IN4; IN17	3
IN0; IN8	2
IN9; IN10; IN13	1
IN1; IN5; IN11; IN14; IN15; IN18	0

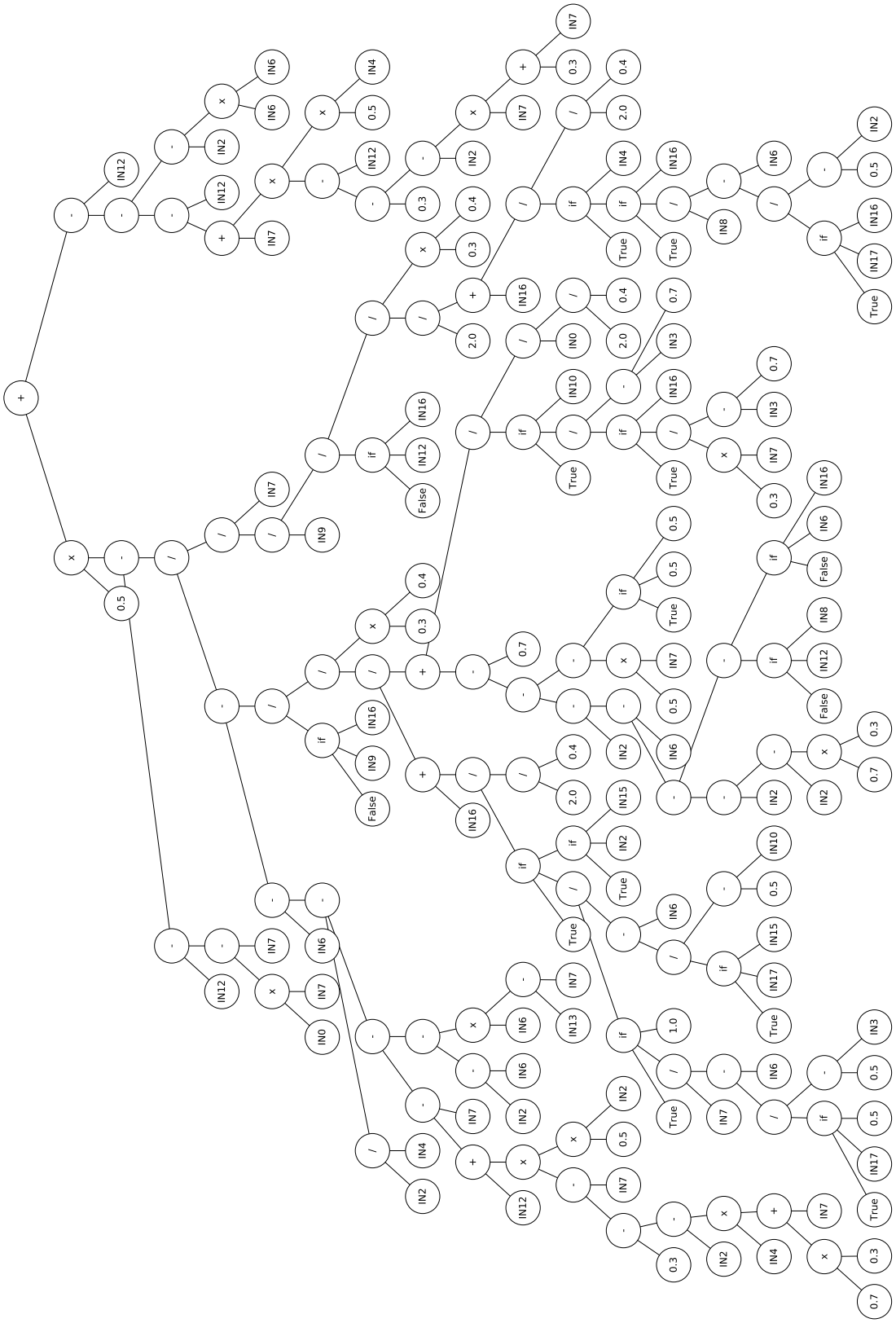

Figure 16: Model with highest Fitness Value

4.4 Calibration

The output from GP can be used to rank the order from least to most probable default. However, these scores do not represent an accurate estimate of the probability. Calibrating the outputs allows for the score to be interpreted as a probability. Probabilities are needed for two reasons. First, the GP outputs are supposed to be used as an on-top addition or deduction to the credit agency score which represents a probability. Second, the outputs are eventually not going to be used solely by itself but in combination with misclassification costs (Zadrozny and Elkan, 2001).

In order to calibrate the GP outputs a non-parametric type of regression called Isotonic Regression (IR) is used. Among the models' benefits is the lack of assumption about the form of the target function. However, IR assumes that the classifier ranks the observations properly. The IR problem is defined by finding a function that minimizes the mean-squared error (Zadrozny and Elkan, 2002; Niculescu-Mizil and Caruana, 2005):

$$\min \sum_i \left(y_i - f\left(x_i\right)\right)^2 \tag{10}$$

IR uses pair-adjacent violators (PAV) to find the best fitting stepwise-constant isotonic function, which works as follows. For every observation x_i, the value of the function to be learned $f\left(x_i\right)$ must be bigger or equal to $f\left(x_{i-1}\right)$. Otherwise the resulting function f is not isotonic and the observations x_i and x_{i-1} are called pair-adjacent violators, hence the name of the algorithm. In order to restore the isotonic assumption, both $f\left(x_i\right)$ and $f\left(x_{i-1}\right)$ are replaced by their average. This process is repeated with all observations until no pair-adjacent violators remain (Ayer et al., 1955).

In the setting of Credit Scoring, PAV works as follows. First, the observations are ordered according to their score values. Then, the $f(x_i)$ is set to 0 if x_1 belongs to the goods, and $f(x_i)$ is set to 1 if x_1 belongs to the bads, respectively. If the score ranks the examples perfectly, all goods come before bads and the values of f are unchanged. As a result, the new probability estimate is 0 for all goods and 1 for all bads. Contrarily, in a random scoring system that does not incorporate any information about the ordering, the probability estimate will be a constant function with the bads rate as value. PAV provides a set of intervals with a corresponding set of estimates $f(i)$ for every interval i. In order to use the results on new data, a probability estimate is mapped to an observation according to its corresponding interval, for which the score value is between the lower and upper boundary (Zadrozny and Elkan, 2001).

In order apply IR on the uncalibrated GP output data set, the validation set is used to train the IR classifier using 10-fold cross-validation. The classifier is subsequently employed on the test set as depicted in figure 8. Below, the calibration plots in figure 17 illustrate the process of calibration on the training set in the Isotonic Regression Plot, how well the resulting set of isotonic values is calibrated on the test set in a reliability curve and the respective distribution of score values in a histogram.
The Isotonic Regression diagram in figure 17 shows the process of calibration. Clearly visible are the intervals and their corresponding probability estimate to which the test set is mapped. An interesting observation is a jump in probability of default between 0.3 and 0.6. Hence, small differences in score value lead to high differences in probability of default. Furthermore, the interval with the highest probability of default extends over a big score range but for a comparatively low probability of default. The implication is that the GP model ranks incorrectly for high score values. This concurs with Zadrozny and Elkan (2001) who point out that in the general case PAV averages out more observations in score ranges where the scoring model ranks properly. Hence, the GP model works better in the score range for lower default probability.

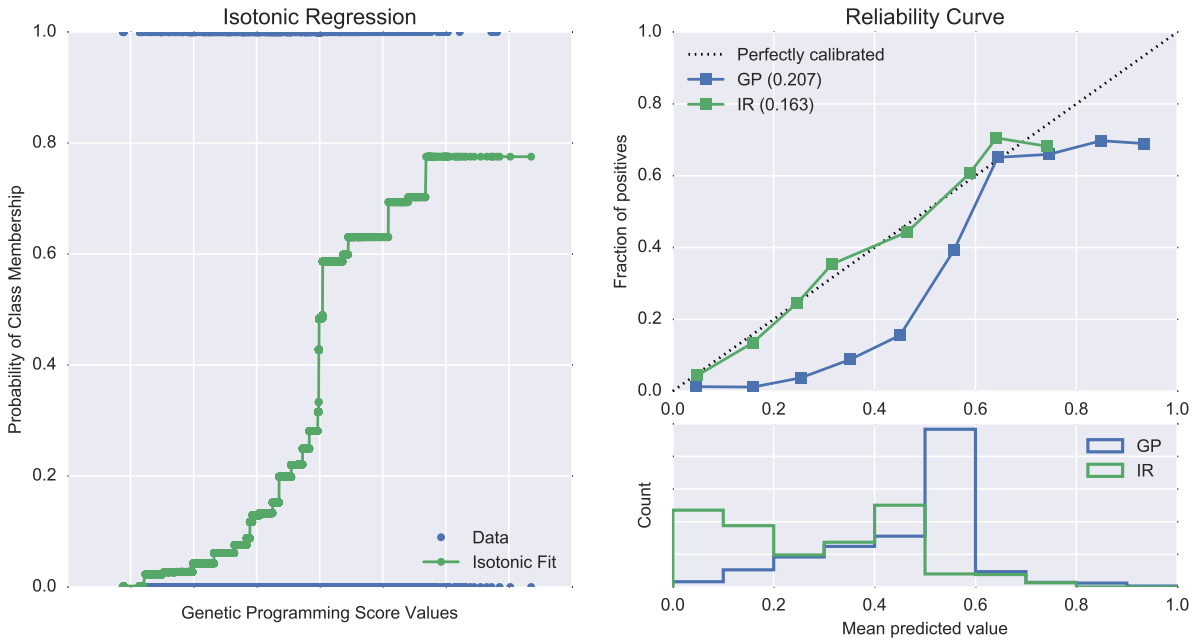

Figure 17: Calibration Plots

The reliability diagram visualizes how well calibrated the predicted probabilities from the normalized uncalibrated GP output and the IR calibrated GP probabilities are. For this purpose, the mean predicted values are plotted against the true fraction of positives. Due to the high number of observations, the score values are discretized into ten bins. A perfectly calibrated classifier has the same fraction of positives as mean predicted values for every class, which implies a score value corresponding to it's assigned default probability. Such a classifier is illustrated by a diagonal line in the reliability plot (Zadrozny and Elkan, 2002; DeGroot and Fienberg, 1983). In addition, the Brier Score is used as verification measure to assess the calibration accuracy of both GP and IR calibrated GP. The Brier Score is defined by the mean squared error of the probability forecasts as depicted in equation 10, which means that a lower score corresponds to a higher accuracy (Brier, 1950).

The GP output values are not well calibrated initially, but improve heavily by applying IR. The normalized uncalibrated GP output values follow a sigmoid-like shape below the perfectly calibrated diagonal line. After calibration, the output values follow the diagonal of perfectly calibrated probabilities with slightly visible discrepancy but for the highest GP outputs whose fractions of positives visibly differs from their corresponding mean predicted value. Furthermore, the calibrated results are only distributed up to 0.7 default probability. This is a result of the constant value for fractions of positives at around 0.7 for the 4 bins with the highest uncalibrated GP output values. Hence, the highest uncalibrated GP output values have the same probability of default as explained above. The Brier Scores confirm the visual examination with an increase from 0.207 of the uncalibrated GP output to 0.163 of the IR calibrated probabilities.

The histogram of the GP output values show that most of the observations are scored in the middle and lower area of the score range. Furthermore, the score range with the highest number of observations is also best calibrated initially. After calibration, the GP output values are more consistently distributed in the lower score range but still incorporate few observations in the higher score range. The histogram further exhibits an inherent effect of IR. Calibrated scores shift towards the tails of the score range.

Concluding, IR is applied in order to learn a mapping from ranking scores to calibrated probability estimates. Transforming scores using IR yields significant improvement and the calibrated scores now translate to default probabilities.

5. RESULTS

In this work ROC-AUC as depicted in equation 7 is the main measure of discriminatory power and used as GP fitness function. However, the ROC curve is insensitive to class distributions and does not account for imbalance in data sets. Therefore, the closely related Precision-Recall (PR) diagram with PR-AUC as measure is included as an additional evaluation criteria.

Datasets have a class imbalance in a wide range of scientific areas. In credit scoring, more people pay their bills than default on payment and therefore an abundance of goods can be observed. In these cases, moving the threshold to gain few more true positives may have a large impact on the number of false positives. As a result, FPR increases slightly but TPR increases a lot more. Such a classifier receives its predictive power by rejecting a lot of goods in order to gain few bads. The greater the imbalance towards goods, the greater the concession in terms of falsely rejected goods. In order to account for the imbalance in the data set, an additional evaluation criteria called precision is used. Precision is the proportion of classified bads that are bads as depicted in equation 5. Hence, the effect of a large number of goods is not included anymore. Precision is a useful measure to evaluate the relevancy of a classifiers'results and depicted with the TPR, called recall in this context, in a Precision-Recall diagram. A good classifier depicts both high values for precision and high values for recall. However, precision and recall incorporate an intuitive trade-off. Assuming that a higher score translates to a lower probability of default and vice versa, the trade-off can be generalized as follows. Lowering the threshold means to classify bads with a higher probability of default, which increases the precision. Additionally, the number of classified bads decreases, which decreases the recall as well. Subsequently, high precision and low recall translates to a high probability of default (low score). In such a model those who are classified as bads are largely bads but many bads were misclassified as goods. Contrarily, low precision and high recall translates to a low probability of default (high score). In such a model those who were classified as bads are largely goods but almost all bads were classified as such. Simplified, the more bads a model classifies as bads the more goods are classified as bads as well. In a PR diagram a classifier that has higher precision values for every recall value than another classifier is usually assumed to be better. However, in settings with imbalanced datasets like credit scoring, that incorporates more goods than bads, it is usually preferable to employ a low threshold point and therefore emphasize the lower recall values.

Unlike in ROC space, in PR space values cannot linearly interpolated. Precision and recall have both True Positives in the numerator but False Positives and False Negatives in the denominator respectively. Hence, precision does not necessarily change linearly as recall varies. Subsequently, linearly interpolating in the PR space is incorrect and yields too high performance values. Interpolation is possible by approximating the interpolation between two points A and B using a method described in Davis and Goadrich (2006). For every integer value between TP_A and TP_B the corresponding FP values can be calculated by adding the local skew $\frac{\mathsf{FP}_B - \mathsf{FP}_A}{\mathsf{TP}_B - \mathsf{TP}_A} * \mathsf{x}$ to FP, with x being the integer values between TP_A and TP_B such that $1 \leq \mathsf{x} \leq \mathsf{TP}_B - \mathsf{TP}_A$. The corresponding precision and recall values are calculated as follows.

$$\text{Recall} = \frac{\mathsf{TP}_A + \mathsf{x}}{\mathsf{TP} + \mathsf{FN}} \tag{11}$$

$$\text{Precision} = \frac{\mathsf{TP}_A + \mathsf{x}}{\mathsf{TP} + \mathsf{x} + \mathsf{FP} + \frac{\mathsf{FP}_B - \mathsf{FP}_A}{\mathsf{TP}_B - \mathsf{TP}_A} * \mathsf{x}} \tag{12}$$

To take into account both precision and recall simultaneously, a single-scalar evaluation measure similar to ROC-AUC, namely PR-AUC, is implemented.

$$\text{Area under the PR curve (PR-AUC)} = \int_{-\infty}^{\infty} \text{Precision(T)TPR}'\text{(T) dT} \tag{13}$$

Optimizing the ROC-AUC does not optimize the PR-AUC, despite the close relationship between ROC space and PR space. Hence a classifier with excellent ROC-AUC may have poor PR-AUC. Subsequently, both ROC and PR space have to be analyzed together (Raghavan et al., 1989; Gordon and Kochen, 1989; Zhu, 2004; Davis and Goadrich, 2006).

In this section the GP results are analyzed and compared to other classifier. Furthermore, the GP model is compared to the existing prescore and the combined performance with the main score is analyzed.

5.1 Discriminatory Power of GP and Comparison to other Classifier

Figure 18 depicts the distribution of goods and bads as well as the average default rate in their corresponding score ranges. Goods and bads are clearly separated with bads mainly in the low score range and goods mainly in the high score ranges. Furthermore, the arithmetic means of goods and bads are reasonably far apart from each other as depicted by the black arrows. The lowest score groups have more than 70% bads while the highest score ranges contain few bads overall. Most bads are distributed among three score groups, which also incorporate a high number of goods. The default rate is steadily decreasing with increasing score values which mirrors the calibration results in figure 17. Clearly visible is a decrease in score groups compared to the original prescore in figure 7, which mirrors a decrease in distribution spread over the score range.

Concluding, GP visibly separates goods and bads but provides a decrease in score range. The model

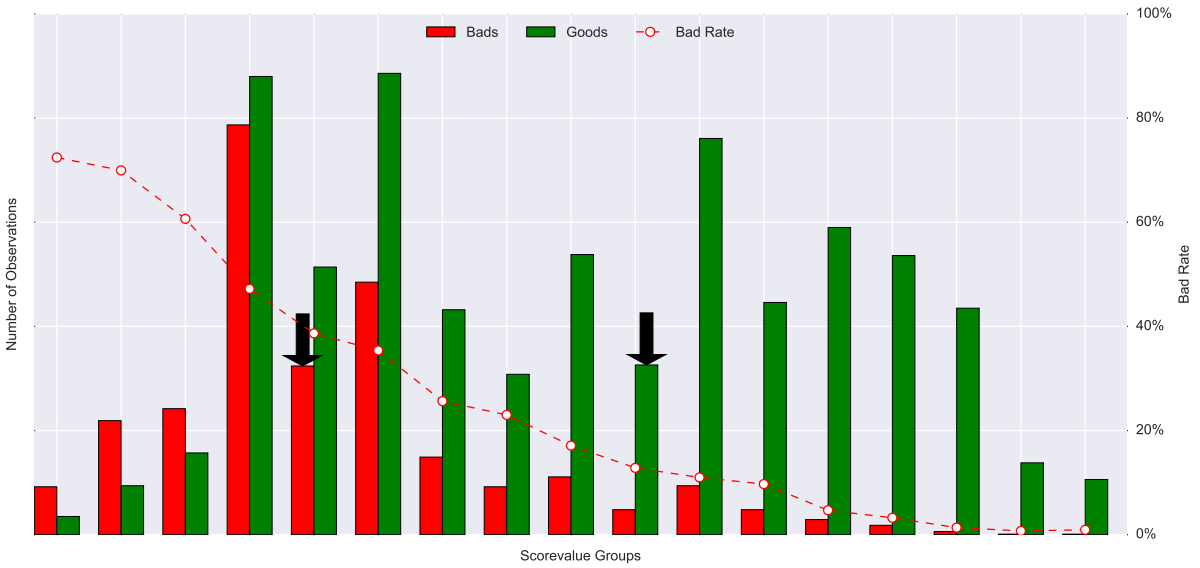

Figure 18: GP Distribution

seems to discriminate better among goods than bads, providing some score groups with mainly goods but no score groups with only bads. Following, GP is compared to Logistic Regression (LR), Support

Vector Machines (SVM) and Boosted Trees (BT). Similar to GP, SVM and BT are subject to IR in order to ensure properly calibrated outputs. LR outputs well calibrated predictions already, and thus IR is not applied. Employing IR on well calibrated methods does not improve the results but may hurt the performance (Niculescu-Mizil and Caruana, 2005). The effect of IR on ROC-AUC and PR-AUC is shown in table 5. The effect is only marginal, and thus can be neglected.

Figure 19 shows the ROC curve with the ROC-AUC as numeric measure and the PR diagram with the PR-AUC.

Table 5: Effect of IR on GP, SVM and BT

	Uncalibrated ROC-AUC		Calibrated ROC-AUC	Uncalibrated PR-AUC		Calibrated PR-AUC
GP	0.779	→	0.777	0.541	→	0.545
SVM	0.754	→	0.756	0.56	→	0.564
BT	0.78	→	0.779	0.54	→	0.54

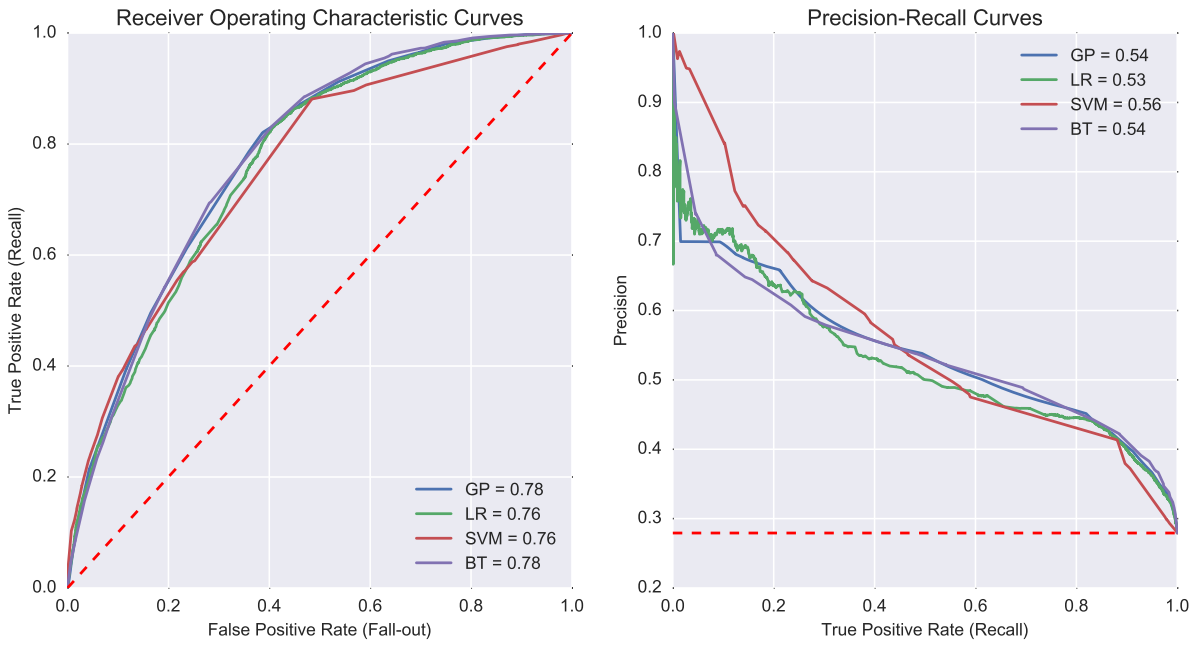

Figure 19: ROC curve and PR diagram

The GP model seems to perform better in the more liberal area of the ROC space, that is the area on the upper-right hand side. Liberal classifiers make positive classification with weak evidence, hence they classify a majority of bads correctly (high TPR) but also classify a high number of goods as bads (high FPR). In contrast, classifier that are strong in the lower right hand side are called conservative, because they classify only with strong evidence (Fawcett, 2003). Subsequently, they have a small TPR but a small FPR as well. Classifiers are desired to be rather conservative than liberal when working with imbalanced data that includes a lot of negative instances. A similar behavior can be observed in the PR diagram. The precision for low recall values is rather low but drops comparably lightly with increasing recall values as

well. This mirrors the distribution of goods and bads as depicted in figure 18. The lowest score group shows a bad rate of around 70% which can be mapped to the precision in low recall values. Moving the threshold to include the three score groups with most bads lowers the precision due to the decreasing bad rate. Moving the threshold even further mainly rejects goods, which leads to a rapid decrease of precision. Subsequently, GP shows a strong performance for high recall values.

Comparing the GP results to the other models reveals that the overall performances of all classifiers are very similar. Nonetheless, some differences can be observed.

The LR model has a similar performance to the GP model in the liberal area. However, in the very conservative area LR seems to be a slightly bit better than GP while from the conservative area to the liberal area the performance is visibly worse. The ROC-AUC for GP (0.78) is slightly higher for GP over LR (0.76). In the PR diagram, LR shows a better precision for very low recall values and similar precision for very high recall values compared to GP. In between, the precision is slightly worse which overall leads to a nearly identical PR-AUC value, with 0.54 for GP and 0.53 for LR.

The SVM model reveals a strong performance in the conservative are of the ROC diagram, with a ROC curve visibly above the other models. However, while leaving the conservative area the curve falls below the other models where it stays throughout the diagram. Overall, the SVM performance in the ROC plot shows the lowest performance of all the models with a ROC-AUC of 0.75. Contrarily, the performance in the PR diagram is by far the best with a PR-AUC value of 0.56. The high performance comes from the lower recall values for which SVM shows a very high precision far above the other classifiers. However, from the central recall values forward, the precision drops heavily below the performance of the other classifiers.

The BT model shows a similar performance to GP with a ROC-AUC value of 0.78. In the PR diagram BT shows a moderate performance similar to GP, with an PR-AUC value of 0.54. Starting very strong the precision is very high for very low recall levels, but drops rapidly for the lower recall values. However, the precision increases again and stays above the other models for the higher recall values.

The analysis shows that while the performances of the models are very similar, their performance in the different areas of the ROC space differs greatly. GP and BT show a consistent performance over the whole ROC space. LR performs moderate in the very liberal and very conservative areas but poor in between. SVM performs strong in the the conservative and visibly poor in the liberal area of the ROC space. Contrarily, BT performs worst in the conservative area but good in the rest of the ROC space with very good performance in the liberal area. A similar behavior can be observed in the PR diagram. While the PR-AUC values do not differ much the models reveal different performance in different areas of the diagram. An interesting observation can be done about the SVM model. The performance in the ROC space is visibly worse than that of its competitors, while the performance in the PR space is visibly better than that of its competitors. For low recall values, the SVM model provides the most relevant results. The results become less precise and therefore less relevant for high recall values.

Next, the performance of the GP model in collaboration with the main score is analyzed.

5.2 Discriminatory Power of GP in Collaboration with the Credit Agency Score

While the prescore needs to be operational on its own in countries that do not or cannot offer a credit agency score based on country specific solvency information, an additional requirement is the utilizability in combination with the main score. Hence, the prescore needs to have an enhancing effect on the discriminatory power of the main score. Both scores are combined by taking the arithmetic mean of both score values. Following, the effect of the combination of both scores on the score distribution and the performance is analyzed.

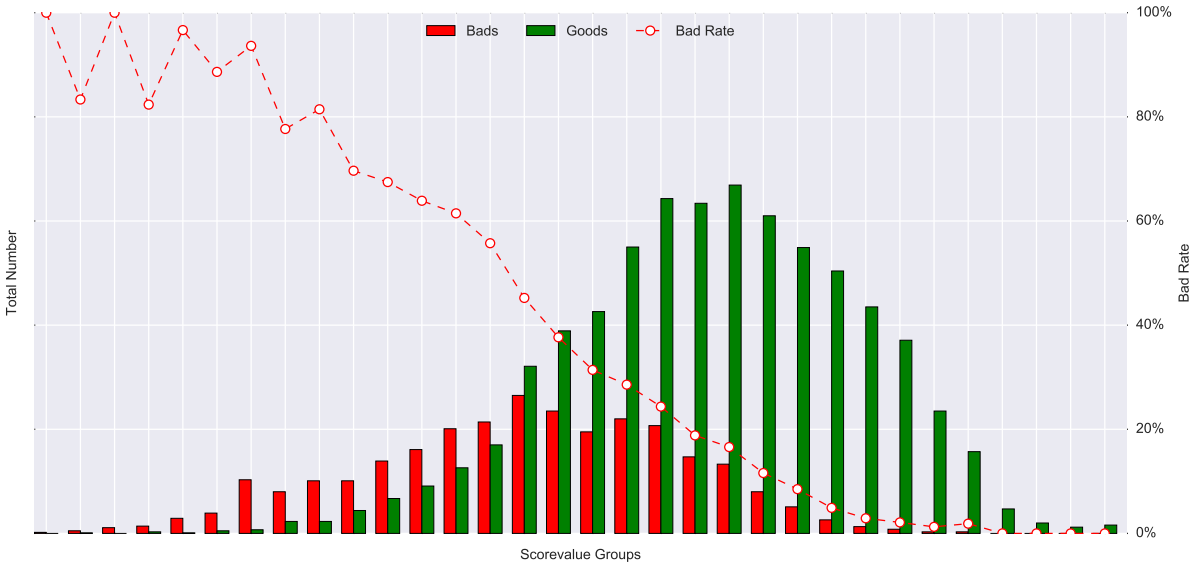

Figure 20: Score Distribution of ABIC and GP

Figure 20 shows the score distribution of the combination of ABIC and GP. The discriminatory capabilities are clearly visibly. The bads, depicted by red bars, are further on the low score value range than the goods, depicted by green bars. However, the bads are spread out rather evenly, with the differences in between score groups only shifting slowly. Contrarily, the goods are strongly centered around their mean value with smaller spread towards the outer score groups. The bad rate, depicted by a red line on the secondary axis, shows high bad rates between 80% and 100% for the first quarter of the score range and very low bad rates below 5% for the forth quarter of the score range. Correspondingly, the bad rate drops heavily and linearly from around 80% to around 5% within the second and third quarter of the score range. Compared to GP, ABIC + GP incorporates a greater score range, with better distributed scores. The bad rate in the low score range is higher and approaches score groups with mainly bads. Similar to GP, the bad rate in the high score range is near zero. Subsequently, the score groups in the high score range mainly consist of goods.

5.2.1 Varying Threshold

In figure 21 are the discrimanatory powers of GP, ABIC, ABIC + GP and the original pre score pictured. An exhaustive analysis of GP can be found in the subsection above. The original prescore reveals to be outperformed by GP with an ROC-AUC of only 0.7. The pre score is especially weak in the more conservative areas but gains some discriminative power in the very liberal area. While its ROC curve runs parallel to GP for most of the ROC space, it also lies constantly below GP and only approaches in the very

liberal area. A similar picture can be seen in the PR diagram. While the PR curve of the original prescore approaches the GP curve for both very low and very high recall values its curve is clearly dominated by the GP curve. Hence, The PR curve reveals a clear concave shape while every other classifier displays a more convex-like shape.

ABIC shows a slightly better performance for the conservative area of the ROC diagram, but drops heavily in performance in the liberal area of the diagram. The ROC-AUC for ABIC is low (0.71). However, combining GP and ABIC boosts the performance of both considerably to a ROC-AUC of 0.81. The ROC curve is high in both conservative and liberal area, resulting from the contradictory curve movements of ABIC and GP. Additionally, while both score perform comparably in the conservative area combining both scores still boosts the performance. Hence, the scores do not identify the same bads but different ones. Contrarily, the performance in the liberal area of the ROC space does not increase from the GP performance. Incidentally, the PR diagram shows a similar picture. ABIC shows high precision in the lower recall area but drops linearly over the recall range. Consequently, ABIC shows the lowest precision in the high recall ranges and drops below GP midway, which has a high precision in the high recall area. The combination of ABIC and GP visibly dominates the PR diagram. While the hybrid score is only slightly better than ABIC in the low recall area, it closes the gap in the whole middle section before crossing GP in the high recall area.

The analysis of both ROC and PR diagram visibly demonstrated the lack of discriminatory power in the pre score and hence the reasoning for this work. Furthermore GP demonstrated its capabilities in separating bads from goods and the prospects of combining the model with the credit agency score withinn the main risk check. However, only slight performance differences to the other classification methods employed can be observed.

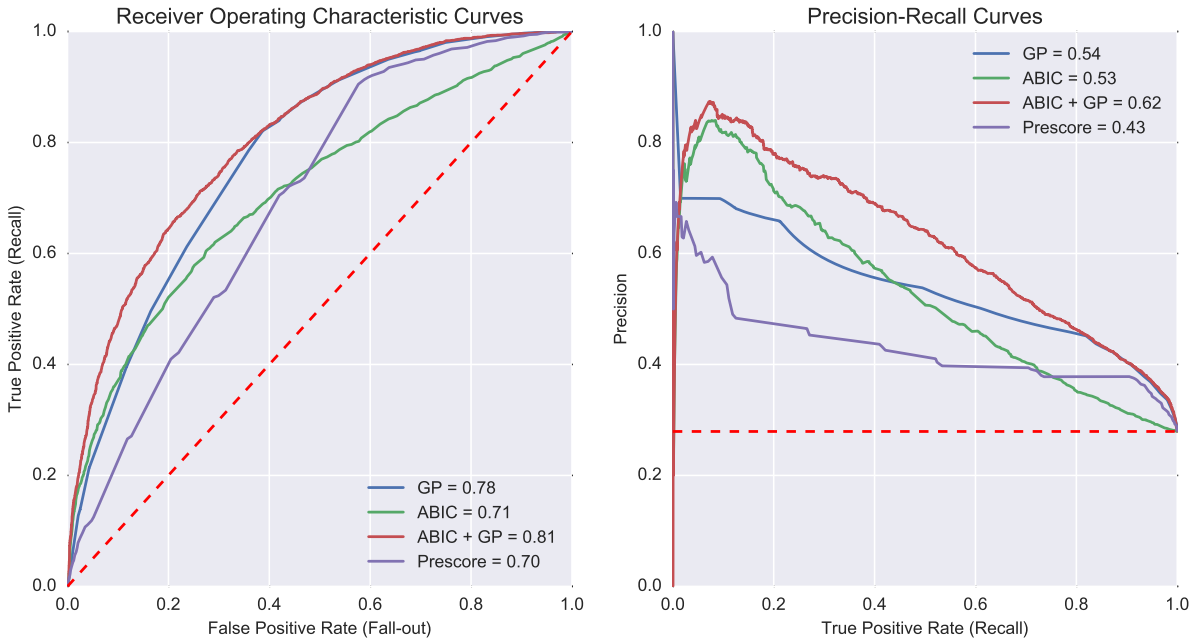

Figure 21: ROC curve and PR diagram for ABIC and GP

5.2.2 Fixed Threshold

Both ROC and PR diagram allow to compare the discriminative power of binary classifiers as their discrimination thresholds are varied. However, the discrimination threshold is essential for the application of a classifier. A number of different methods for quantifying the quality of predictions at a specific threshold exist. The method used most frequently in research is the accuracy (ACC), which is defined as the fraction of correct classifications. Equation 6 depicts the accuracy (García et al., 2014; Davis et al., 1992; Zhang et al., 2007).

The optimal threshold can be found by calculating the accuracy for every possible threshold and subsequently return the threshold with the highest accuracy. Table 6 shows the highest accuracy for GP, pre score, ABIC and ABIC + GP.

Table 6: Classifier Accuracies

Classifier	Accuracy
GP	75.0%
Pre Score	72.8%
ABIC	75.9%
ABIC + GP	78.5%

The accuracy results mirror the previous results. GP and ABIC have a comparable accuracy, with a slight advantage of ABIC (75.9%) over GP (75.0%) and together they form a more powerful classifier with the highest accuracy (78.5%). The pre score obtains an accuracy comparable to a classifier that classifies every request as good (72.8%).

While accuracy is the most used classification evaluation criteria, it is strongly biased for the majority class and therefore not recommended for unbalanced data sets (García et al., 2014). Additionally, accuracy and many other evaluation techniques assume symmetric misclassification costs for goods and bads. Often times however, the costs of accepting bads are higher than the costs of rejecting goods. Consequently, instead of using an evaluation technique based on the overall error, a cost function can be employed (Frydman et al., 1985; West, 2000). If C_1 denotes the cost of accepting bads, C_2 denotes the cost of rejecting goods and π_1 and π_1 are the ratios of goods and bads in the population, then the cost function is defined as Cost $= C_1 * \text{FNR} * \pi_1 + C_2 * \text{FPR} * \pi_2$ (Lee and Chen, 2005). While cost functions take into account the misclassification costs for goods and bads, the assume the order value to be constant across classes. Because the order values for the data set used are known, a profit based evaluation technique is employed in which order values are adjusted to include misclassification costs. The misclassification costs are defined as abortion rate and marginal return and provided by AFS. All requests that are classified as goods are reduced to 25% of their original value, which corresponds to the profit margin. Additionally, requests of goods that are classified as bads (FP) are reduced by another 30%, which corresponds to the abortion rate of potential customers who are unwilling to pay via the offered payment types. Bads who are classified as goods (FN) are reduced by the profit margin and deducted from the sum of the remainder, excluding bads who are classified as bads (TP) and subsequently rejected. In the latter case, it is assumed that rejected bads do not continue the order process. The

calculations are depicted by equations 14 to 18.

$$\text{TP} = 0 \tag{14}$$

$$\text{FN} = \text{Ordervalue} * (1 - \text{Profit Margin}) \tag{15}$$

$$\text{FP} = \text{Ordervalue} * \text{Profit Margin} * (1 - \text{Abortion Rate}) \tag{16}$$

$$\text{TN} = \text{Ordervalue} * \text{Profit Margin} \tag{17}$$

$$\text{OV} = \text{TN} + \text{FP} - \text{FN} \tag{18}$$

The bads have a mean order value that is 2% higher than the mean order value of goods. However, taking into account the misclassification costs changes the values considerably. Using TN as determination base yields FN that is 206% higher than TN and FP that is 30% lower (=Abortion Rate) than TN. This means that 1€ profit for goods who are classified as goods yield 0.7€ profit if they are classified as bads instead. Similarly, if they are classified as goods but turn out to be bads they yield a negative profit of 3.06€. Hence, falsely changing the classification from good to bad induces profit deduction of $1€ - 0.7€ = 0.3€$. Falsely keeping the classification as good induces profit deduction of $0€ - (-3.06€) = -3.06€$. Accordingly, for every falsely classified bads 10 goods can be falsely classified as bads.

To protect the confidentiality of the data, the real order values are concealed and only transformed deviations to bench marks are stated. The deviations are transformed to ratios by dividing them through the sum of order values. A pessimal classifier that rejects nothing (PES) and an optimal classifier that rejects only bads (OPT) serve as bench marks.

Results for GP, the original pre score, ABIC and ABIC + GP are depicted in table 7. In order to allow for interclass analysis, the results for bads and goods are shown supplementary to the totals. Additionally, the accuracy as defined in equation 6 is included for comparison purposes.

Table 7: GP Order Value Comparison

		GP		Pre Score		ABIC		ABIC + GP	
		OV	ACC	OV	ACC	OV	ACC	OV	ACC
PES	Bads	22.4%	2.9%	22.3%	2.1%	23.0%	5.5%	22.6%	9.0%
	Goods	-3.8%	-0.2%	-4.0%	-0.2%	-5.2%	-0.2%	-4.0%	-0.4%
	Total	18.6%	2.7%	18.3%	1.9%	17.9%	5.2%	18.6%	8.6%
OPT	Bads	-0.7%	-20.2%	-0.8%	-21.0%	0.0%	-17.6%	-0.5%	-14.1%
	Goods	-3.8%	-0.2%	-4.0%	-0.2%	-5.2%	-0.2%	-4.0%	-0.4%
	Total	-4.5%	-20.3%	-4.8%	-21.2%	-5.2%	-17.8%	-4.4%	-14.5%

The pessimal classifier accepts all bads and therefore has a high bads deduction on the overall order value. On the same time, the order value for goods is at a maximum, because no goods are rejected. The results show a strong decrease in bads deduction (increase in OV) for all models, with GP being only slightly better than the original pre score. ABIC shows an even better result than ABIC + GP, which is decreased by the lower value for GP. However, while ABIC has the highest order value decrease for goods, GP has the lowest, which leads to a moderate order value decrease for ABIC + GP. The pre score performs similar to ABIC + GP and only slightly worse than GP. Due to the imbalanced data set, the

impact of goods is higher than the impact of bads. Subsequently, ABIC has the worst performance as a result of the high order value deduction for goods. Nonetheless, ABIC + GP performs similar to GP, sharing the highest increase in order value compared to the pessimal classifier. The pre score performs only slightly worse resulting in an increase in order value between ABIC and the similar performing ABIC + GP and GP. Those results are in high contrast to the accuracy results, in which the pre score performs worst, followed by GP and outperformed by ABIC, which in turn is massively outperformed by ABIC + GP. Unlike the order value, the accuracy shows little deviation between the pessimal classifier and the models, with a minimal value of 1.9% for the pre score and a maximal value of 8.6% for ABIC + GP. Contrarily, comparing the models with the optimal value shows high deviations for accuracy but little for order value. The optimal classifier discriminates perfectly between goods and bads and therefore rejects all bads and accepts all goods. Subsequently, there is no order value deduction for bads and the order value for goods is at a maximum. The results show the differences between accuracy and order value as evaluation measure. Accuracy is shows the models to be rather similar to a pessimal classifier, while the order value shows the models to be rather similar to the optimal classifier. The previously observed pattern that the results between models show little difference is repeated. However, the order value result are only compared to bench marks so far.

Table 8: Deviation to GP

	Pre Score	ABIC	ABIC + GP
Bads	-15.0%	100%	30.9%
Goods	-1.6%	-10.3%	-1.4%
Total	-2.4%	-5.7%	0.1%

Table 8 shows the difference in order value between GP and Pre Score, ABIC and ABIC + GP. Mirroring previous results, the order values are continuously lower for the Pre Score than GP, with -15% for the bads, -1.6% for the goods and a total of -2.4%. However, the total value difference is comparatively small taking into account the huge differences shown in the ROC and PR diagram represented by figure 21. The combination of ABIC and GP offers yet again the best results. The order value of bads is 30.9% higher for ABIC + GP compared to GP, but the order value for goods is 1.4% smaller. The impact of the smaller order value for goods is higher than the impact of the higher order value for bads, because the data set is unbalanced. Hence, the total difference adds up to only 0.1% increase from GP to ABIC + GP. These results are effected by the ABIC results. For ABIC, the highest order value threshold is found on a level that effectively rejects all bads, but a lot of goods as well. Resulting, there is no discount from bads but a high difference to the GP results for goods (-10.3%) can be observed instead. Hence, ABIC has a total difference of -5.7%.

6. CONCLUSION AND DISCUSSION

The objective of this work is to replace the existing pre score with a model that needs to be operational both on its own and in combination with the credit agency score. The focus is upon developing a model based on Genetic Programming to predict the probability of default on payment.

Credit Scoring was originally implemented by mail-order companies but is nowadays mainly used by financial institutions. The particularities of e-commerce in Germany allow to revise the utility of credit scoring in this environment. The results show an increase in profit of around 18.6% by employing a credit scoring model based on GP, compared to not employing credit scoring at all. Subsequently, despite being neglected by research, credit scoring represents a necessity for e-commerce vendors in Germany.

In order to evaluate the discriminatory power of the GP model, it is compared to models based on Logistic Regression, Support Vector Machines and Boosted Trees using ROC and PR analysis. All models developed show comparable overall discriminatory power. The GP model shows a higher discriminatory power than the pre score using ROC and PR evaluation as well as using the error-based evaluation techniques. GP is evaluated not only with error-based evaluation techniques but also on profit. Analyzing the profit obtained by the GP model shows an increase of 2.4% over the pre score. Combining GP and the credit agency score into a single score increases its predictive power. In ROC, PR and accuracy analysis ABIC + GP shows the highest performance, while the 0.1% performance increase in profit obtained is insignificant.

On the basis of the empirical results, it can be concluded that the model has a higher discriminatory power than the pre score and works well with the existing credit agency score. Hence, the GP model is capable to work as a replacement to the pre score. Although the GP model improves the performance in regards to the generic pre score it does not in regards to other commonly employed methods. This observation concurs with similar observations in the literature as presented in section 2.

There are three potential reasons for the similar results.

- The flat maximum effect. This means that the predictive abilities of different classifiers are basically indistinguishable (Lovie and Lovie, 1986; Thomas, 2000).
- Legislation severely limits the use of data available and narrows the data basis. This may work as an catalyst for the flat maximum effect.
- It is briefly mentioned in section 3 that the data in the data warehouse does not entirely match the data used for services called. Hence, the data set used misses variables that are used in the operational system and in application of the pre score. These variables can only be accessed in the logging database as depicted in 3 and are unavailable in this work for which only data from the data warehouse is available. This is an especially severe drawback because AFS credit scoring experts assume that the missing variables incorporate high predictive powers.

Despite its superior performance in comparison to the originally pre score, it is highly unlikely that the GP model will be adopted as replacement.

The major reason is that effort of developing the model is disproportionate to the return. Although the results of the other models presented, that is Logistic Regression, Support Vector Machines and

Boosted Trees, have a slightly worse overall performance, they are much more easily deployable techniques. They are provided by a plethora of software, among which is drag and drop software such as SAS Enterprise Miner, Weka and KNIME as well as software that requires coding such as SAS Base, R and Python. More specifically, systems to easily develop and deploy a Logistic Regression Model are already in place at AFS. A model that is only slightly better most likely won't lead to a change in paradigm.

Furthermore, the development times and run times are very short, especially compared to GP. In this work, Logistic Regression, Support Vector Machines and Boosted Trees are developed using the Python module sci-kit learn. Model specific code consists of 7 lines. The models are fit on a consumer laptop utilizing a single CPU core. Genetic Programming is developed using the Python module DEAP with model specific code consisting of 221 lines. The evolutionary process is employed on a server utilizing 8 CPU cores. The run time for Logistic Regression is 0.29 seconds. Support Vector Machines need 308.16 seconds, Boosted Trees 37.21 seconds and Genetic Programming 5 days. Taking into account the run time of the models, it is hard to argue in regards to Genetic Programming. Ultimately, due to restrictions such as the flat maximum effect as explained above and the increase in time and effort needed for development and deployment, GP might not be the most economical data mining application in the current setting.

Finally, legislation and regulatory agencies require the transparency of credit score cards. However, GP models are not the most easy to comprehend and require more detailed explanation. Additionally, it is unclear if the data protection commissioner in whose jurisdiction AFS operates accepts elaborate credit scoring systems.

7. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

There is a number of ways in that this research can be expanded. Most importantly, the lack of variables has to be addressed. Although it is an obvious proposal to increase a models discriminatory power by adding variables with new information value, it is not without clear reasoning. Both the fact that the proposed model is designed with less variables than its predecessor and the fact that the analysts of AFS identified the missing variables as powerful illustrates the importance.

Additionally, the implementation of Genetic Programming may be extended to Geometric Semantic Genetic Programming, a new branch in the development of Genetic Programming that has only recently emerged.

Classic Genetic Programming as applied in this work and originally introduced by Koza (1992) is designed to manipulate possible solutions on a syntactic level. Consequently, the effect and impact of modifications on the output of a solution cannot be anticipated. The branch of Genetic programming that is concerned with the output properties is called Semantic Genetic Programming. Semantics are the constraints that bind instruction's output to input. In a machine learning setting, where the data is given by input and desired output for observations, the semantics of a solution are defined as the vector of its output values. However, semantics do not include outputs for all possible solutions, but only for those present in the training set. The semantic of a solution is determined during the process of GP because a solution has to be employed on every observation in order to calculate the fitness of the solution (Krawiec and Pawlak, 2013; Vanneschi et al., 2014a).

The research field of methods that utilize semantics can be divided into three categories. Diversity methods are concerned with keeping diversity loss in check, indirect semantic methods that act on the solutions syntax but incorporate semantics as survival criteria and direct semantic methods that search directly the semantic space of solutions(Vanneschi et al., 2014a; Moraglio et al., 2012). Although research showed that all three categories yield superior performance to traditional methods, the most promising category is direct semantic methods.

Moraglio et al. (2012) defined genetic operators that directly search the semantic space by transforming the syntax of GP solutions, named geometric semantic operators. Solutions produces by geometric semantic crossover cannot incorporate a fitness below that of their parents and geometric semantic mutation incorporate the property of inducing an unimodal fitness landscape (Castelli et al., 2016). The fitness of a solution in the semantic search space is the distance to the optimum (Moraglio et al., 2012). Hence, a solution in the semantic search space cannot have both a lower distance to the optimum and a lower fitness than another solution. Vanneschi et al. (2013) point out that these properties hold independently of the data on which the solutions are evaluated. As a result, this holds also in the test set and a solution cannot incorporate fitness below that of its parents in the test set, even though it is produced on a train set. Consequently, Geometric Semantic Operators help control overfitting (Vanneschi et al., 2013).

Geometric semantic operators are defined by Moraglio et al. (2012) as follows:

Given two parent functions $T_1, T_2 : \mathbb{R}^n \to \mathbb{R}$ geometric semantic crossover generates

$$T_{XO} = (T_1 * T_{XO}) + ((1 - T_R) * T_2) \tag{19}$$

where T_{XO} is a random real function with codomain $[0, 1]$.
Given a function $T : \mathbb{R}^n \rightarrow \mathbb{R}$ geometric semantic mutation generates

$$T_M = T + ms * (T_{R1} - T_{R2}) \tag{20}$$

where T_{R1} and T_{R2} are a random real functions with codomain $[0, 1]$ and ms is a parameter called mutation step.

Geometric semantic operators produces offsprings that include their whole parent trees and one or more random trees. As a result the offsprings grow in size exponentially in the number of generations (Moraglio et al., 2012). With increasing generations individuals evolve to a size that slows fitness evaluation down until the system becomes unusable. Vanneschi et al. (2013) proposes a solutions that stores only the initial trees, the semantic for every individual and the reference to their parents. The initial solutions are evaluated by calculating the output values for every observation. These vectors of output values, the semantics of the solutions, are retained. After employing of crossover and mutation as depicted in equation 19 and 20, the new solutions are evaluated and their semantics are retained. For the next generation, only the semantic values are needed but not the syntactic structure. Hence, the latter is not retained. However, the solutions are returned in a concentrated fashion and in order to reconstruct any single solution, all solutions referenced up to this point have to be reconstructed. Therefore, information about the employed Genetic Semantic Operator and the parents of a solution is crucial. Depending on the solution the reconstruction might be unfeasible but at least computationally expensive. However, if the form of the solution does not matter, than it can be used as a 'black-box' solution.

After completion of the evolutionary process, the population in every generation can be scanned for solutions that are not ancestors of any solutions in the following population. This can also be done after every generation in order to thin out the populations and keep the storage space required down.

Even though Geometric Semantic Genetic programming is still in its infancy, a promising future in research is evident. Experimental results by Moraglio et al. (2012), Castelli et al. (2013), Vanneschi et al. (2013) and Vanneschi et al. (2014b) show that Geometric Semantic GP outperforms classic GP on both training and test data. Expanding the research on Credit Scoring by implementing Geometric Semantic Genetic Programming may enable the extraction of previous concealed knowledge. However, considering the flat maximum effect and other limitations discussed in the previous section might render the implementation of Geometric Semantic Genetic Programming in vain.

8. REFERENCES

Abdou, H. A. (2009). Genetic programming for credit scoring: The case of Egyptian public sector banks. *Expert Systems with Applications*, 36(9):11402–11417.

Abdou, H. A. and Pointon, J. (2011). Credit Scoring, Statistical Techniques and Evaluation Criteria: A Review of the Literature. *Intelligent Systems in Accounting, Finance & Management*, 18(2-3):59–88.

Angeline, P. J. (1997). Comparing subtree crossover with macromutation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1213:101–111.

Ayer, M., Brunk, H. D., Ewing, G. M., Reid, W. T., Silverman, E., Rpeid, W. T., and Silvermniant, E. (1955). An Empirical Distribution Function for Sampling with Incomplete Information. *Source: The Annals of Mathematical Statistics*, 26219247(4):641–647.

Baesens, B., Gestel, T. V., Viaene, S., Stepanova, M., Suykens, J., and Vanthienen, J. (2003). Benchmarking state of the art classification algorithms for credit scoring. *Journal of the Operational Research Society*.

Blickle, T. and Thiele, L. (1995a). A Comparison of Selection Schemes used in Genetic Algorithms. *Evolutionary Computation*, 2(11):311–347.

Blickle, T. and Thiele, L. (1995b). A Mathematical Analysis of Tournament Selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 9–16.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Brindle, A. (1981). *Genetic algorithms for function optimization*. PhD thesis, University of Alberta.

Castelli, M., Castaldi, D., Giordani, I., Silva, S., Vanneschi, L., Archetti, F., and Maccagnola, D. (2013). An efficient implementation of geometric semantic genetic programming for anticoagulation level prediction in pharmacogenetics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8154 LNAI:78–89.

Castelli, M., Vanneschi, L., and Popovič, A. (2016). Parameter evaluation of geometric semantic genetic programming in pharmacokinetics. *International Journal of Bio-Inspired Computation*, 8(1):42.

Chellapilla, K. (1997). Evolving computer programs without subtree crossover. *IEEE Transactions on Evolutionary Computation*, 1(3):209–216.

Chen, M. C. and Huang, S. H. (2003). Credit scoring and rejected instances reassigning through evolutionary computation techniques. *Expert Systems with Applications*, 24(4):433–441.

Davis, J. and Goadrich, M. (2006). The Relationship Between Precision-Recall and ROC Curves. *Proceedings of the 23rd International Conference on Machine learning – ICML'06*, pages 233–240.

Davis, R. H., Edelman, D. B., and Gammerman, A. J. (1992). Machine-learning algorithms for credit-card applications. *IMA Journal of Management Mathematics*, 4(1):43–51.

DeGroot, M. H. and Fienberg, S. E. (1983). The Comparison and Evaluation of Forecasters. *The Statistician*, 32(1):12–22.

Desai, V. S., Crook, J. N., and Overstreet, G. A. (1996). A comparison of neural networks and linear scoring models in the credit union environment. *European Journal of Operational Research*, 95(1):24–37.

Durand, D. (1941). Risk Elements in Consumer Instalment Financing. *NBER Books*.

Eggermont, J., Kok, J. N., and Kosters, W. a. (2004). Genetic Programming for Data Classification: Partitioning the Search Space. *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1001–1005.

Eiben, A. and Smith, J. (2015). *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2 edition.

Eiben, A. E. and Schoenauer, M. (2002). Evolutionary computing. *Information Processing Letters*, 82(1):1–6.

Fang, Y. and Li, J. (2010). A review of tournament selection in genetic programming. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6382 LNCS(M4D):181–192.

Fawcett, T. (2003). ROC Graphs : Notes and Practical Considerations for Data Mining Researchers. *HP Invent*, page 27.

Fisher, R. A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188.

Fittkau & Maaß Consulting (2014). 38. WWW-Benutzer-Analyse W3B: Kaufentscheidung im Internet.

Fogarty, D. J. (2012). Using Genetic Algorithms for Credit Scoring System Maintenance Functions. *International Journal of Artificial Intelligence & Applications*, 3(6):1–8.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, 13:2171–2175.

Frigge, D. (2016). Online-Payment 2016.

Frydman, H., Altman, E. I., and Kao, D.-L. (1985). Introducing Recursive Partitioning for Financial Classification: The Case of Financial Distress. *The Journal of Finance*, 40(1):269–291.

García, V., Marqués, A. I., and Sánchez, J. S. (2014). An insight into the experimental design for credit risk and corporate bankruptcy prediction systems. *Journal of Intelligent Information Systems*, 44(1):159–189.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1st edition.

Goldberg, D. E. and Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. *Foundations of Genetic Algorithms*, 1:69–93.

Gordon, M. and Kochen, M. (1989). Recall-Precision Trade-Off : A Derivation. *Journal of the American Society for Information Science*, 40(3):145–151.

Grefenstette, J. J. and Baker, J. E. (1989). How Genetic Algorithms Work: A Critical Look at Implicit Parallelism. In *Proceedings of the Third International Conference on Genetic Algorithms*, page 8, George Mason University, USA. Morgan Kaufmann Publishers Inc.

Hand, D. J. and Henley, W. E. (1997). Statistical Classification Methods in Consumer Credit Scoring: a Review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 160(3):523–541.

Hanley, J. A. and McNeil, B. J. (1982). The Meaning and Use of the Area under a Receiver Operating (ROC) Curvel Characteristic. *Radiology*, 143(1):29–36.

Henley, W. E. (1995). *Statistical aspects of credit scoring*. PhD thesis, Open University.

Henley, W. E. and Hand, D. J. (1996). A k-Nearest-Neighbour Classifier for Assessing Consumer Credit Risk. *The Statistician*, 45(1):77.

Hilas, C., Kazarlis, S., Rekanos, I., and Mastorocostas, P. (2014). A genetic programming approach to telecommunications fraud detection and classification. *Proceedings of the 2014 International Conference on Circuits, Systems, Signal Processing, Communications and Computers*, pages 77–83.

Hoffmann, F., Baesens, B., Martens, J., Put, F., and Vanthienen, J. (2002). Comparing a Genetic Fuzzy and a Neurofuzzy Classifier for Credit Scoring. *Computational Intelligent Systems for Applied Research*, pages 355–362.

Hoffmann, F., Baesens, B., Mues, C., Van Gestel, T., and Vanthienen, J. (2007). Inferring descriptive and approximate fuzzy rules for credit scoring using evolutionary algorithms. *European Journal of Operational Research*, 177(1):540–555.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An introductory Analysis with Applications to Biology, Control and Artificial Intelligence*.

Huang, C.-L., Chen, M.-C., and Wang, C.-J. (2007). Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications*, 33(4):847–856.

Huang, J. J., Tzeng, G. H., and Ong, C. S. (2006). Two-stage genetic programming (2SGP) for the credit scoring model. *Applied Mathematics and Computation*, 174(2):1039–1053.

Khashman, A. (2010). Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. *Expert Systems with Applications*, 37(9):6233–6239.

Kirschen, R. H., O'Higgins, E. A., and Lee, R. T. (2000). The Royal London Space Planning: An integration of space analysis and treatment planning. *American Journal of Orthodontics and Dentofacial Orthopedics*, 118(4):448–455.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press.

Krawiec, K. and Pawlak, T. (2013). *Locally geometric semantic crossover: A study on the roles of semantics and homology in recombination operators*, volume 14.

Lahsasna, A., Ainon, R. N., and Wah, T. Y. (2010). Credit scoring models using soft computing methods: A survey. *The International Arab Journal of Information Technology*, 7(2):115–123.

Langdon, W. B. (1999). Size Fair and Homologous Tree Genetic Programming Crossovers. *Proceedings of the 1st Genetic and Evolutionary Computation Conference, GECCO 1999*, 2:1092–1097.

Lee, T. and Chen, I. (2005). A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. *Expert Systems with Applications*, 28(4):743–752.

Lovie, A. D. and Lovie, P. (1986). The flat maximum effect and linear scoring models for prediction. *Journal of Forecasting*, 5(3):159–168.

Luke, S. and Spector, L. (1998). A Revised Comparison of Crossover and Mutation in Genetic Programming. *Proceedings of Genetic Programming*, pages 208–213.

Malhotra, R. and Malhotra, D. K. (2002). Differentiating between good credits and bad credits using neuro-fuzzy systems. *European Journal of Operational Research*, 136(1):190–211.

Marques, a. I., Garcia, V., and Sanchez, J. S. (2013). A literature review on the application of evolutionary computing to credit scoring. *Journal of the Operational Research Society*, 64(9):1384–1399.

Mays, E. (2001). *Handbook of credit scoring*. Global Professional Publishi.

McGrath, J. J. (1960). Improving credit evaluation with a weighted application blank. *Journal of Applied Psychology*, 44(5):325.

Mester, L. J. (1997). What's the Point of Credit Scoring ? *Business Review*, 3:3–16.

Miller, B. L. and Goldberg, D. E. (1995). Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems*, 9(3):193–212.

Miller, B. L. and Goldberg, D. E. (1996). Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise. *Evolutionary Computation*, 4(2):113–131.

Moraglio, A., Krawiec, K., and Johnson, C. G. (2012). Geometric semantic genetic programming. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7491 LNCS(PART 1):21–31.

Myers, J. H. and Forgy, E. W. (1963). The Development of Numerical Credit Evaluation Systems. *Journal of the American Statistical Association*, 58(303):799–806.

Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning - ICML '05*, number 1999, pages 625–632, New York, New York, USA. ACM Press.

Ong, C., Huang, J., and Tzeng, G. (2005). Building credit scoring models using genetic programming. *Expert Systems with Applications*, 29(1):41–47.

Oreski, S., Oreski, D., and Oreski, G. (2012). Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment.

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., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Poli, R., Langdon, W., and McPhee, N. (2008). *A Field Guide to Genetic Programing*. Lulu Enterprises.

Raghavan, V., Bollmann, P., and Jung, G. S. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7(3):205–229.

Reichert, A. K., Cho, C.-C., and Wagner, G. M. (1983). An Examination of the Conceptual Issues Involved in Developing Credit-Scoring Models. *Journal of Business & Economic Statistics*, 1(2):101–114.

Sackmann, S., Siegl, M., and Weber, D. (2011). Ein Ansatz zur Verbesserung der Steuerung des Zahlungsausfallrisikos im E-Commerce (B-to-C). *Zeitschrift für Betriebswirtschaft*, 81(2):139–153.

Seidenschwarz, H., Weinfurtner, S., Stahl, E., and Wittmann, G. (2014). Gesamtkosten von Zahlungsverfahren - Was kostet das Bezahlen im Internet wirklich?

Seni, G. and Elder, J. F. (2010). Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2(1):1–126.

Siddiqi, N. (2006). *Credit risk scorecards: Developing and implementing intelligent credit scoring*. John Wiley & Sons, 3 edition.

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, 16(2):149–172.

Thomas, L. C., Edelman, D. B., and Crook, J. N. (2002). *Credit Scoring and Its Applications*. Society for Industrial and Applied Mathematics.

United States Code (1974). Equal Credit Opportunity Act.

Vanneschi, L., Castelli, M., Manzoni, L., and Silva, S. (2013). A New Implementation of Geometric Semantic GP and Its Application to Problems in Pharmacokinetics. In *Proceedings of the 16th European Conference on Genetic Programming*, pages 205–216. Springer-Verlag Berlin Heidelberg, Vienna, Austria.

Vanneschi, L., Castelli, M., and Silva, S. (2014a). A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214.

Vanneschi, L., Silva, S., Castelli, M., and Manzoni, L. (2014b). Geometric Semantic Genetic Programming for Real Life Applications. In *Genetic Programming Theory and Practice XI*, number 3, pages 191–209. Springer New York.

Wach, E. P. (2011). Trends im eCommere 2011.

Weinfurner, S., Weisheit, S., Wittmann, G., Stahl, E., and Pur, S. (2011). Zahlungsabwicklung im E-Commerce.

West, D. (2000). Neural network credit scoring models. *Computers and Operations Research*, 27(11-12):1131–1152.

White, D. R. and Poulding, S. (2009). A rigorous evaluation of crossover and mutation in genetic programming. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5481 LNCS(November):220–231.

Wiginton, J. C. (1980). A Note on the Comparison of Logit and Discriminant Models of Consumer Credit Behavior. *The Journal of Financial and Quantitative Analysis*, 15(3):757.

Wolbers, H. (1949). The use of the biographical data blank in predicting good and potentially poor credit risks. *Unpublished master's thesis, University of Southern California*.

Yang, Z., Wang, Y., Bai, Y., and Zhang, X. (2004). Measuring Scorecard Performance. In Bubak, M., van Albada, G. D., Sloot, P. M., and Dongarra, J., editors, *Computational Science — ICCS 2004*, volume 3038, pages 900–906, Kraków. Springer-Verlag Berlin Heidelberg.

Zadrozny, B. and Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. *Icml*, pages 1–8.

Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining KDD 02*, pages 694–699.

Zhang, D., Huang, H., Chen, Q., and Jiang, Y. (2007). A comparison study of credit scoring models. *Proceedings - Third International Conference on Natural Computation, ICNC 2007*, 1(Icnc):15–18.

Zhu, M. (2004). Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2:1–11.

9. APPENDIX

```
# -*- coding: utf-8 -*-
"""
Python Code deployed for the thesis "Credit Scoring Using Genetic
Programming"

@author: David Micha Horn
"""

#-------------------------------------------------------------------------#
#-----------------------------IMPORTS-------------------------------------#
#-------------------------------------------------------------------------#

import numpy as np
import random
import operator
import itertools
import datetime
from sklearn import metrics
from scoop import futures
from deap import base
from deap import creator
from deap import tools
from deap import gp
import csv
import os
import pickle

#-------------------------------------------------------------------------#
#-----------------------------READ-IN-------------------------------------#
#-------------------------------------------------------------------------#

date = datetime.datetime.now().date()

inputpath = r'inputpath'
outputpath = inputpath.rsplit('\\',2)[0] + r'\output' + '\\'  + \
             str(date)
halloffamepath = outputpath  + r'\hall_of_fame'
file1 = r'filename.pkl'

with open(inputpath + '\\' + file1) as database:
    dataReader = csv.reader(database)
    inputData = list(list(float(elem) for elem in row)
                     for row in dataReader)
```

```
y_true = []
for row in inputData:
    y_true.append(row[-1])


# Dictionary with column names for arguments
with open(inputpath + '\\args.pkl', 'rb') as inp:
        args = pickle.load(inp)

lendf = len(args)

#-------------------------------------------------------------------------------#
#-----------------------GENETIC PROGRAMMING-------------------------------------#
#-------------------------------------------------------------------------------#

pset = gp.PrimitiveSetTyped("MAIN",
                            itertools.repeat(float, lendf),
                            float, "IN")

pset.renameArguments(**args)

# boolean operators
pset.addPrimitive(np.logical_and, [bool, bool], bool)
pset.addPrimitive(np.logical_or, [bool, bool], bool)
pset.addPrimitive(np.logical_not, [bool], bool)

# floating point operators
# Define a protected division function

def protectedDiv(left, right):
    with np.errstate(divide='ignore',invalid='ignore'):
        x = np.divide(left, right)
        if isinstance(x, np.ndarray):
            x[np.isinf(x)] = 1
            x[np.isnan(x)] = 1
        elif np.isinf(x) or np.isnan(x):
            x = 1
    return x

pset.addPrimitive(np.add, [float,float], float)
pset.addPrimitive(np.subtract, [float,float], float)
pset.addPrimitive(np.multiply, [float,float], float)
pset.addPrimitive(protectedDiv, [float,float], float)
```

```
# logic operators
# Define a new if-then-else function
def if_then_else(input, output1, output2):
    if input: return output1
    else: return output2

pset.addPrimitive(if_then_else, [bool, float, float], float)
pset.addPrimitive(np.less_equal, [float, float], bool)
pset.addPrimitive(np.greater_equal, [float, float], bool)
pset.addPrimitive(np.equal, [float, float], bool)
pset.addPrimitive(np.not_equal, [float, float], bool)

# Terminals
pset.addTerminal(-1.0, float)
pset.addTerminal(0.0, float)
pset.addTerminal(0.1, float)
pset.addTerminal(0.2, float)
pset.addTerminal(0.3, float)
pset.addTerminal(0.4, float)
pset.addTerminal(0.5, float)
pset.addTerminal(0.6, float)
pset.addTerminal(0.7, float)
pset.addTerminal(0.8, float)
pset.addTerminal(0.9, float)
pset.addTerminal(1.0, float)
pset.addTerminal(2.0, float)
pset.addTerminal(3.0, float)
pset.addTerminal(4.0, float)

pset.addTerminal(False, bool)
pset.addTerminal(True, bool)

creator.create("FitnessMulti", base.Fitness, weights=(1.0,-1.0))
creator.create("Individual", gp.PrimitiveTree,
               fitness=creator.FitnessMulti)

toolbox = base.Toolbox()
toolbox.register("expr", gp.genHalfAndHalf,
                 pset=pset, min_=1, max_=6)
toolbox.register("individual", tools.initIterate,
                 creator.Individual, toolbox.expr)
toolbox.register("population", tools.initRepeat, list,
                 toolbox.individual)
toolbox.register("compile", gp.compile, pset=pset)
```

```
def varAnd(population, toolbox, cxpb, mutpb):

    offspring = map(toolbox.clone, population)
    # Apply crossover and mutation on the offspring
    for i in range(1, len(offspring), 2):
        if random.random() < cxpb:
            offspring[i-1], offspring[i] =
                toolbox.mate(offspring[i-1], offspring[i])
            del offspring[i-1].fitness.values,
                offspring[i].fitness.values

    for i in range(len(offspring)):
        if random.random() < mutpb:
            offspring[i], = toolbox.mutate(offspring[i])
            del offspring[i].fitness.values

    return offspring

def ea(population, toolbox, cxpb, mutpb, ngen, mstats=None,
           halloffame=None, verbose=__debug__):

    logbook = tools.Logbook()
    logbook.header = "gen", "nevals", "fitness", "size"
    logbook.chapters["fitness"].header = "min", "avg", "max"
    logbook.chapters["size"].header = "min", "avg", "max"

    # Evaluate the individuals with an invalid fitness
    invalid_ind =
        [ind for ind in population if not ind.fitness.valid]
    fitnesses = toolbox.map(toolbox.evaluate, invalid_ind)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    if halloffame is not None:
        halloffame.update(population)

    record = mstats.compile(population) if mstats else {}
    logbook.record(gen=0, nevals=len(invalid_ind), **record)
    if verbose:
        print(logbook.stream)

    # Begin the generational process
    for gen in range(1, ngen + 1):
        # Select the next generation individuals
        offspring = toolbox.select(population, len(population))
```

```
        # Vary the pool of individuals
        offspring = varAnd(offspring, toolbox, cxpb, mutpb)

        # Evaluate the individuals with an invalid fitness
        invalid_ind =
                [ind for ind in offspring if not ind.fitness.valid]
        fitnesses = toolbox.map(toolbox.evaluate, invalid_ind)
        for ind, fit in zip(invalid_ind, fitnesses):
            ind.fitness.values = fit

        # Update the hall of fame with the generated individuals
        if halloffame is not None:
            halloffame.update(offspring)

        with open(halloffamepath + '\\' + \
                        'halloffame_gen_' + str(gen) + \
                        '.pkl', 'wb') as output:
            pickle.dump(halloffame, output, -1)


        # Replace the current population by the offspring
        population[:] = offspring

        # Append the current generation statistics to the logbook
        record = mstats.compile(population) if mstats else {}
        logbook.record(gen=gen, nevals=len(invalid_ind), **record)

        with open(outputpath + '\\' + 'logbook_stream_' + \
                    str(date) + '.pkl', 'wb') as output:
            pickle.dump(logbook, output, -1)

        if verbose:
            print(logbook.stream)

    return population, logbook

def evalFunction(individual):
    # Transform the tree expression in a callable function
    func = toolbox.compile(expr=individual)

    y_pred = []
    for row in inputData:
        y_pred.append(func(*row[:lendf]))
```

```
    result = metrics.roc_auc_score(y_true, y_pred)

    return result, len(individual)

toolbox.register("map", futures.map)

toolbox.register("evaluate", evalFunction)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("mate", gp.cxOnePoint)
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)
toolbox.register("mutate", gp.mutUniform,
                 expr=toolbox.expr_mut, pset=pset)


#-------------------------BLOAT CONTROL----------------------------#
toolbox.decorate("mate",
                 gp.staticLimit(key=operator.attrgetter("height"),
                                max_value=17))
toolbox.decorate("mutate",
                 gp.staticLimit(key=operator.attrgetter("height"),
                                        max_value=17))

toolbox.decorate("mate", gp.staticLimit(key=len,
                                        max_value=400))
toolbox.decorate("mutate", gp.staticLimit(key=len,
                                        max_value=400))

def main(popsize, generations):
    random.seed(10)
    pop = toolbox.population(n=popsize)
    hof = tools.HallOfFame(10)

    stats_fit =
        tools.Statistics(key=lambda ind: ind.fitness.values[0])
    stats_size =
        tools.Statistics(key=lambda ind: ind.fitness.values[1])
    mstats =
        tools.MultiStatistics(fitness=stats_fit, size=stats_size)
    mstats.register("avg", np.mean)
    mstats.register("std", np.std)
    mstats.register("min", np.min)
    mstats.register("max", np.max)


    pop, logbook = ea(pop,
```

```
                        toolbox,
                        0.9,
                        0.1,
                        generations,
                        mstats,
                        halloffame=hof,
                        verbose=True)

    return pop, mstats, hof, logbook

if __name__ == "__main__":
    if not os.path.exists(outputpath):
        os.makedirs(outputpath)

    if not os.path.exists(halloffamepath):
        os.makedirs(halloffamepath)

    pop, mstats, hof, logbook = main(popsize = 10,
                                     generations = 10)

    def save_object(obj, filename):
        with open(outputpath + '\\' + filename + '_' + \
                str(date) + '.pkl', 'wb') as output:
            pickle.dump(obj, output, -1)

    for obj, name in [(pop, 'pop'),
                      (hof, 'hof'),
                      (logbook, 'logbook')]:
        save_object(obj, name)
```