KADIR HAS UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

TOWARDS A DYNAMIC SYSTEM IN CREDIT SCORING

GRADUATE THESIS

SÜLEYMAN ŞAHAL

January, 2017

TOWARDS A DYNAMIC SYSTEM IN CREDIT SCORING

SÜLEYMAN ŞAHAL

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

in

MANAGEMENT INFORMATION SYSTEMS

KADIR HAS UNIVERSITY

January, 2017

KADIR HAS UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

TOWARDS A DYNAMIC SYSTEM IN CREDIT SCORING

SÜLEYMAN ŞAHAL

APPROVED BY:

Prof. Dr. Hasan Dağ	(Advisor)	(Kadir Has Uni.)
Assoc. Prof. Dr. Songül Varlı Albayrak	(Co-Advisor)	(Yıldız Technical Uni.)
Prof. Dr. Oktay Taş	(Jury Member)	(Istanbul Technical Uni.)
Assoc. Prof. Dr. Mehmet Nafiz Aydın	(Jury Member)	(Kadir Has Uni.)
Asst. Prof. Dr. Christophe Bisson	(Jury Member)	(Kadir Has Uni.)

APPROVAL DATE: 19 /01/2017

"I, Süleyman Şahal, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis."

SÜLEYMAN ŞAHAL

ABSTRACT

TOWARDS A DYNAMIC SYSTEM IN CREDIT SCORING

Süleyman Şahal

Master of Science in Management Information Systems

Advisor: Prof. Dr. Hasan Dağ

January, 2017

In credit scoring statistical methods like logistic regression have been successfully used for years. In the last two decades, several data mining algorithms have gained popularity and proved themselves to be useful in credit scoring. Most recent studies indicate that data mining methods are indeed good predictors of default. Additionally, ensemble models which combine single models have even better predictive capability. However, in most studies the rationale behind data transformation steps and selection criteria of single models while building ensemble models are unclear.

In this study, it is aimed to construct a fully automated, comprehensive, and dynamic system which gives the ability to a credit analyst to make credit decisions without any human interaction, solely based on the data set. With this system, it is hoped not to miss any valuable data transformation step and it is assumed that each built-in model in RapidMiner is a possible candidate to be the best predictor. To this end, a model comparison engine has been designed in RapidMiner. This engine conducts almost every kind of data transformation on the data set and gives the opportunity to observe the effects of data transformations. The engine also trains every possible model on every possible transformed data. Therefore, the analyst can easily compare the performances of models.

As the final phase, it is aimed to develop an objective method to select successful single models to build more successful ensemble models without any human interaction. However, research in this direction has not been able to achieve an objective method. Hence, to build ensemble models, six of the most successful single models are chosen manually and every possible combination of these models are fed to another engine: ensemble model building engine. This engine tries every combination of single models in ensemble modelling and provides the credit analyst with the ability to find the best possible combination of single models, and finally to reach the ultimate, most successful model that can later be used in predicting the creditworthiness of counterparties.

Keywords: Credit Risk, Credit Scoring, Data Mining, Machine Learning, Ensemble Model, Dynamic System, Automated System, RapidMiner

ÖZET

KREDİ SKORLAMADA DİNAMİK BİR SİSTEME DOĞRU

Süleyman Şahal

Yönetim Bilişim Sistemleri, Yüksek Lisans

Danışman: Prof. Dr. Hasan Dağ

Ocak, 2017

Kredi skorlamasında, lojistik regresyon gibi istatistiksel yöntemler yıllarca başarılı bir şekilde kullanılmıştır. Son yirmi yılda, bazı veri madenciliği algoritmaları popülarite kazanmış ve kredi skorlamasında kullanışlı olduklarını kanıtlamışlardır. Güncel araştırmalar, veri madenciliği yöntemlerinin gerçekten de iyi tahmin ediciler olduklarını göstermektedir. Ek olarak, tekil modelleri birleştiren gruplama modelleri daha da iyi tahmin yeteneğine sahiptir. Bununla birlikte, çoğu araştırmada, veri dönüştürme adımlarının arkasındaki mantık ve gruplama modellerini oluşturmada tekil modellerin seçim kriterleri yeterince açık değildir.

Bu çalışmada, bir kredi analistine yalnızca verilere dayanarak ve herhangi bir insan etkileşimi olmaksızın kredi kararları verme yeteneği kazandıracak, tamamen otomatik, kapsamlı ve dinamik bir sistem kurulması amaçlanmaktadır. Bu sistem ile herhangi bir değerli veri dönüşüm adımını kaçırmamak umut edilmekte ve RapidMiner'daki her yerleşik modelin en iyi tahminci olmaya aday olduğu varsayılmaktadır. Bu amaçla, RapidMiner'da bir model karşılaştırma motoru tasarlandı. Bu motor veri kümesi üzerinde hemen her tür veri dönüşümünü gerçekleştirmekte ve veri dönüşümlerinin etkilerini gözleme fırsatını vermektedir. Motor ayrıca mümkün olan her dönüştürülmüş veri üzerinde olası tüm modellerini de eğitir. Bu sayede, analist modellerin performanslarını kolayca karşılaştırabilmektedir.

Son aşamada, insan etkileşimi olmaksızın daha başarılı gruplama modelleri oluşturmada başarılı tekil modelleri seçmek için nesnel bir yöntem geliştirilmesi amaçlanmaktadır. Fakat, bu doğrultudaki araştırmalar nesnel bir yönteme ulaşılmasını sağlayamadı. Bu nedenle, gruplama modelleri oluşturmak için, en başarılı tekil modellerin altısı manuel olarak seçildi ve bu modellerin olası kombinasyonları başka bir motora beslendi: gruplama modeli oluşturma motoru. Bu motor, gruplama modelinde tekil modellerin her kombinasyonunu denemekte ve kredi analistine tekil modellerin mümkün olan en iyi kombinasyonunu bulma ve sonuç olarak karşı tarafların kredi değerliliği hakkında tahminde bulunabilecek en mükemmel ve en başarılı modele ulaşma olanağı sağlamaktadır.

Anahtar Kelimeler: Kredi Riski, Kredi Skorlama, Veri Madenciliği, Makine Öğrenme, Gruplama Modelleri, Dinamik Sistem, Otomatik Sistem, RapidMiner

Acknowledgements

I would like to thank to my advisor Prof. Dr. Hasan Dağ for his contributions throughout my master's education in Kadir Has University and for his support and guidance during my research.

I would also like to thank to Assoc. Prof. Dr. Songül Varlı Albayrak for accepting being my co-advisor in this research on late notice and for sparing her time to listen to my research proposal.

Finally, I would also like to thank all members of the thesis defense jury for their time and sincere contributions.

TABLE of CONTENTS

LIST of TABLES

LIST of FIGURES

LIST of ABBREVIATIONS

ANN	Artificial Neural Network
AUC	Area Under Curve
BRSA	Banking Regulation and Supervision Agency - BDDK
CV	Cross Validation
disc	Discretization
DL	Deep Learning
DT	Decision Trees
EMBE	Ensemble Model Building Engine
FN	False Negatives
FP	False Positives
FPR	False Positive Rate
imp	Imputation
KNN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
LR	Logistic Regression
MCE	Model Comparison Engine
ms	Milliseconds
NB	Naïve Bayes
NN	Neural Network
norm	Normalization
num	Numerical Conversion
OS	Optimized Selection
PCA	Principal Component Analysis
RF	Random Forest
ROC	Receiver Operating Characteristic
RT	Random Tree
STDEV	Standard Deviation
SVM	Support Vector Machines
TN	True Negatives
TNR	True Negative Rate
TP	True Positives
TPR	True Positive Rate
UCI	University of California, Irvine

1 INTRODUCTION

The main task of banking industry is lending money to borrowers. Decision to lend or not fundamentally depends on the riskiness of the borrower. In other words, will the borrower pay his/her debt fully in due times? To answer this question, vast amount of information regarding loan applicant, be it a real person or a company, have been gathered for decades and statistical and data mining (machine learning[1]) techniques deployed on them. This broad process is called credit scoring and the outcome of it is usually a binary type decision variable (such as good/bad, approve/reject) calculated based on the estimated likelihood of a nonpayment on prompt time. The benefits and adverse consequences of credit scoring almost entirely reflect in the performance of the financial institution, the former as being profit and the latter as loss. Additionally, a good credit scoring system with minimal error will speed up decision making processes. So, measuring the riskiness of a borrower as accurate as possible has outmost importance. To contribute to this endeavor, state-of-the-art data mining methods are implemented and their accuracy rates are compared in this study.

1.1 Study Focus

The main inquiry of this study is threefold: The first task is the comparison of prediction effectiveness of contemporary data mining methods in credit scoring. The second successive task is choosing appropriate successful models among these to construct a more successful heterogeneous ensemble model. Third and finally, building two engines in RapidMiner environment that can be used on any kind of credit data. First of these two engines transforms the data and compares the

[1] Although there are differences, within the context of credit scoring, data mining and machine learning concepts are used interchangeably throughout this study.

prediction performance of different models, and the second one tries different combinations of single successful models in order to build the final, more successful ensemble model.

Data mining and credit scoring are already extensive subjects by themselves. Hence, to be able to focus on evaluation of data mining methods while staying within the boundaries of such a concise study, some basic assumptions have been made and some restrictions are placed.

The first task in credit scoring is collecting data. The procedure of collecting data inherently involves domain knowledge of the experts in the field of credit scoring. That is, experts choose which data to record along with their form based on their experiences and priorities. Since this study is concentrated on the comparison of the capabilities of the data mining techniques, publicly available real world credit data sets have been chosen to investigate. The German and Australian credit data sets from University of California, Irvine (UCI) Machine Learning Repository are collections of real data comprised of 1,000 examples and 20 attributes (Hofmann n.d.) and 690 examples and 14 attributes respectively. These data sets have been widely used in similar studies and have become sort of a benchmark to judge whether new studies have significant improvements or not over the prevalent models. Examples in the data sets represent real people who had applied for loans and credit cards. Therefore, the data sets represent retail credit applications, not companies. Descriptive statistics of the data sets will be presented later in the study.

1.2 Objectives of the Study

The first objective is to detect the most successful data mining models in assessing creditworthiness of loan applicants. Together with finding successful models,

understanding the effects of data transformations on model accuracies is also one of the major objectives of this study. After finding effective models, the underlying reasons which might cause those specific models to be successful are scrutinized. Moreover, it is anticipated that ensemble models will yield better results. After conducting this experiment, it is also hoped to understand the contributions of ensemble models to single methods. Whether an objective method exists in selecting successful single models while building ensemble models is also one of the crucial points expected to be illuminated throughout this study. Finally, it is expected to develop a stable, accurate and precise credit scoring system which can efficiently be deployed to any kind of real world retail credit applications to help experts decide without much anxiety over the credit risk applicants might bear.

1.3 Methodology

Throughout this study a methodological approach has been strictly followed. Before delving into the research, a thorough scanning of existing studies has been conducted.

1.3.1 Literature Review

The first step was to review relevant literature where the effectiveness of various data mining models in the assessment of credit scoring has been investigated. Peer reviewed scientific journals, conference proceedings, books, theses, and dissertations have been scanned and approximately eighty sources are reviewed. A summary of reviewed source types is at Table 1.1. Contents and findings of these reviewed studies will be discussed in Section 2.3.

Table 1.1 Literature Review, Source Types

Source Type	Number
Articles in Scientific Journals	54
Books	7
Theses & Dissertations	8
Others	~10
Total	**~80**

1.3.2 Software

In every data analysis, a researcher needs some programming tools. For some basic data manipulations Microsoft Excel is used. For the advanced data analytics part of this study, RapidMiner has been extensively used. RapidMiner is a free visual workflow environment where one can use predefined data mining models and develop his/her programming scripts[2].

1.3.3 Data Analysis

Any kind of data analysis project should ideally encompass all or most of the following phases (Tufféry 2011):

1. Identifying the objectives
2. Listing relevant data
3. Collecting the data
4. Exploring the data
5. Preprocessing (blending, cleansing) the data
6. Transforming the data if necessary
7. Training data mining models
8. Tuning model criteria
9. Validating models
10. Deploy strategies consistent with model outcomes

[2] For further information please visit https://RapidMiner.com.

Since the focus of this study is the comparison the data mining models, the objective phase is already included in the project. To be able to compare the findings of this study, publicly available date sets have been chosen and they are already well organized real data. Thus, phases from 2 to 5 can be skipped. Remaining phases comprise the body of this study.

1.4 Outline of the Study

Chapter 2 presents the basics about credit scoring and data mining. Findings of the recent and relevant studies are also given here. Finally date sets used in this study are introduced.

Chapter 3 establishes the foundation of this study. Necessary data transformation steps, aspects of model training and evaluation metrics are clarified.

Chapter 4 presents the experimental setting and gives brief descriptions about data mining models.

2 BACKGROUND

This chapter introduces the main research topic that is credit scoring and gives a brief introduction to the data mining field. Finally, the data sets are presented and former studies around this subject are evaluated.

2.1 Credit Scoring

After every application for a loan, customers are evaluated by their personal information and historical performance with the aim of estimating their creditworthiness (Han et al. 2013). This is where credit scoring comes into play. Thanks to the recent developments in cloud technology and big data, there is tremendous amount of data collected about people. The main aim of credit scoring is to analyze those data and distinguish good payers from bad ones, utilizing characteristics such as account, purpose of loan, marital status etc. (Mues et al. 2004).

Although no single credit scoring model is perfect and they considerably fail to identify risky borrowers (Finlay 2011), it is the primary tool of lending institutions in making credit decisions. Scoring originally started based on subjective evaluations of experts. Later it was based on 5 Cs: character, capital, collateral, capacity, conditions. Then, statistical techniques emerged such as linear discriminant analysis, logistic regression. They had been applied successfully for some time, but the problem with those models is they usually require the data not to violate some distributional assumptions such as normality. In recent decades, many studies and applications supported the idea that new data mining techniques such as neural networks, support vector machines can be used as alternatives (Wang et al. 2011). In contrast to statistical methods they do not assume certain distributions. Credit scoring has been one of the most successful applications of data mining (Finlay 2011).

2.2 Data Mining

As being an area intertwined with so many disciplines it is argued that data mining is not an actual scientific branch. Rather, it is a set of methods for exploring and analyzing data sets in an automatic way in order to extract certain unknown or hidden rules, patterns, associations among those data. Data mining has basically two functions: descriptive and predictive. Descriptive techniques are designed to bring out information that is present but buried, while predictive techniques are designed to extrapolate new information based on present information (Tufféry 2011).

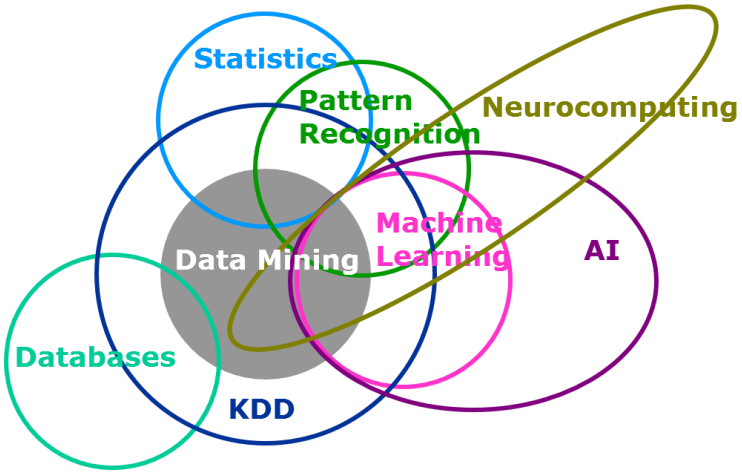

Figure 2.1 Data Mining Venn Diagram
Source: (Mitchell-Guthrie 1998)

The diagram which is created for a data mining course by experts in SAS Institute back in 1998 demonstrates the complicated nature of data mining field.

Extracting information from data set is an inductive process which means the models learn from the data. There are two common types of learning: unsupervised learning (learning without definite goals), supervised learning (learning with definite goals) (Kantardzic 2011).

2.2.1 Unsupervised Learning

Under this learning scheme there is no output or response variable and only the input or explanatory variables are provided to learner. It is expected from the learner to build the model on its own. Since there is no output variable and hence there is no relation to detect, the primary goal of unsupervised learning is to discover natural structure of the data set (Kantardzic 2011). The most common unsupervised learning method is called clustering.

2.2.1.1 Clustering

Cluster is a subset of data which are similar out of the whole data set. Clustering is the procedure to categorize each example in a data set into groups such that the members of each group are as similar as possible to one another (Sayad 2016). The chart Figure 2.2 is a demonstration of a clustering learner. The learner assigns every example in the data set to one of the two clusters. There are various models where the number of clusters set beforehand or left to optimization algorithms of learners.

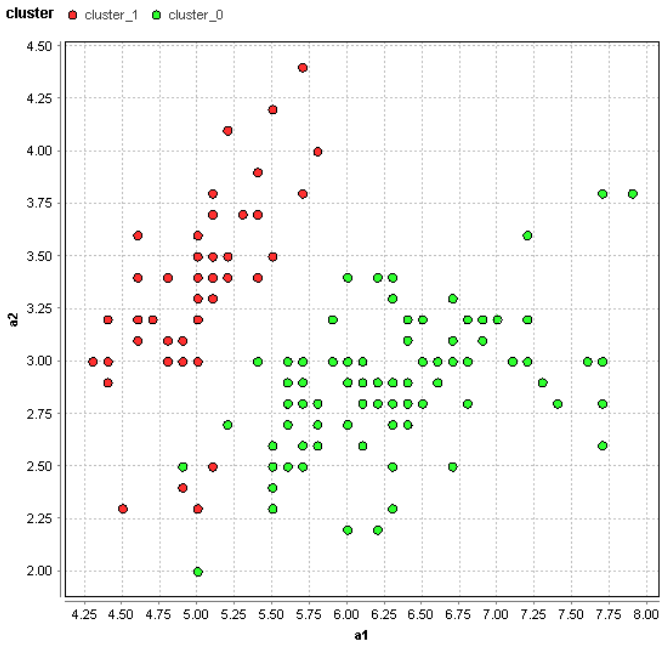

Figure 2.2 Clustering Example

2.2.2 Supervised Learning

Unlike unsupervised learning, in supervised learning there is definite output variable. The main goal is to reveal the hidden mapping between input variables and the output variable (Brownlee 2016). Most of the data mining tasks fall into this category. Supervised learning can be further classified into two fields: regression and classification.

2.2.2.1 Regression

The data mining task becomes regression when the output variable has real values. In very general terms, regression is describing and revealing the relationship between the output variable and one or more input variables. More technically, regression attempts to explain variability in the output variable with respect to movements in the input variables (Brooks 2008). In its simplest form the relationship between output variable and one input variable can be depicted with a linear line as shown in the Figure 2.3

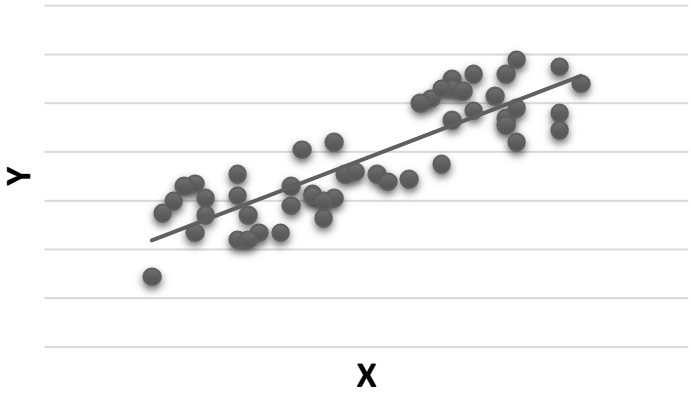

Figure 2.3 Example of Simple Linear Regression

2.2.2.2 Classification

Classification learners are used to reveal the relation between the class of the output variable and input variables. Like clustering learners, classifiers also assign each

example in a data set to a class. The difference is, in classification, classes are already defined. After revealing this relation, learner assigns new examples of which classes are not known to existing classes (Yazdani et al. 2014). Number of classes might be two or more. The special case is having two classes. It is also called binary output variable and this case is also the subject of this study.

In credit scoring the output variable, which is creditworthiness of an entity, takes only one of the values, good or bad. For binary classification tasks, commonly used learners are decision trees, Naïve Bayes, k-nearest neighbor (KNN), artificial neural networks (ANN), support vector machines (SVM), logistic regression, linear discriminant analysis (LDA), random trees and forests, deep learning. Each of these learners will be performed and their relative successes are evaluated in following sections.

2.3 Former Studies

The financial industry utilizing credit scoring systems directly experience the results of their systems, as benefits or losses. Also, because of the central role of the banking industry in the national and the global economy, credit granting policies and systems are highly controlled by local authorities (like BRSA in Turkey) and by BASEL regulations as well. Because of these reasons, the whole industry together with scientists in associated fields try to reach the best model for decades. There are numerous studies only focusing on comparing of statistical and data mining algorithms. A selection of researches in which German and Australian credit data sets has been used is presented at Table 2.1. In addition to this selection, around six studies per year were published in 2010 and 2011 as well.

Table 2.1 Selection of Researches on Learner Comparison in Credit Scoring

Research	Reference	# Learners
Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research	(Lessmann et al. 2015)	41
Combining cluster analysis with classifier ensembles to predict financial distress	(Tsai 2014)	21
Improving experimental studies about ensembles of classifiers for bankruptcy prediction and credit scoring	(Abellán & Mantas 2014)	5
Consumer credit risk: Individual probability estimates using machine learning	(Kruppa et al. 2013)	5
Two-level classifier ensembles for credit risk assessment	(Marqués et al. 2012b)	17
Exploring the behavior of base classifiers in credit scoring ensembles	(Marqués et al. 2012a)	35
Relevance vector machine based infinite decision agent ensemble learning for credit risk analysis	(Li et al. 2012)	5
Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method	(Hens & Tiwari 2012)	4
An experimental comparison of classification algorithms for imbalanced credit scoring data sets	(Brown & Mues 2012)	9
An empirical comparison of conventional techniques, neural networks and the three stage hybrid Adaptive Neuro Fuzzy Inference System (ANFIS) model for credit scoring analysis: The case of Turkish credit card data	(Akkoç 2012)	4

Although the data sets are definite and the same across these studies, data transformation steps, such as normalization, applied are numerous, but the underlying rationale is mostly unclear. That is, researchers either assume that the benefit of some data transformation steps are widely known and an explanation is not needed or they apply a specific transformation because the models require so. Because of these reasons a comprehensive approach is followed in this study and almost every kind of data transformations is applied on the data sets in order to be able to observe the effects of different transformations.

Secondly, in most studies models compared are already limited, that is, they are already shortlisted out of a bigger model set. Yet, the underlying reason behind this

selection is unclear. That is why a comprehensive approach in model selection is followed also. Each 43 built-in model in RapidMiner is trained on the distinctly transformed data sets and the outcomes are evaluated. The details of these data transformations and model selection are explained in Section 4.1.

Lastly, it is now clear that ensemble models (mostly weighted voting), which combine single models to increase the overall prediction success, are more successful (Lessmann et al. 2015). However, every ensemble model can possibly include a different combination of single models. The number of choices is practically uncountable. The rationale behind the selection of single models while building the ensemble model is mostly ambiguous among these studies. In other words, there is no objective and definite method which enables practitioners to find the optimum combination of single models to build ensemble model. This is one of the main inquires of this study. If we can find an objective method to select single models than it might be possible to build a totally automated and dynamic system which takes any kind of credit data sets and applies data transformations, tries every model available on them and finally build the most successful ensemble model. Yet, as will be explained in the final chapter, we failed to find an objective method and the selection of single models to build the most successful ensemble model still requires human interaction.

2.4 Data Exploration

As mentioned before the German credit data set belongs to real loan applicants and Australian credit data set belongs to credit card applications. German data has 1,000 examples and 20 attributes without any missing values. Among 1,000 examples 300 ones are bad credits. So, the ratio of good to bad credits is 7:3 which should not create an imbalanced data set problem. Australian data has 690 examples and 14

attributes, with some missing values. But missing values are replaced by means and modes of corresponding attributes. The ratio of good to bad credits is about 44:56 which is very good. In Australian data set all attributes and values are converted to meaningless symbols to protect privacy. Types of attributes are presented at Table 2.2 and Table 2.3. Detailed description of the data sets and value ranges of attributes are presented at Appendix A: Description of the German credit dataset and Appendix B: Description of the Australian credit dataset.

Table 2.2 German Credit Data Attributes

No	Attributes	Short Name	Type	Value Range	Mean/Mode
1	status of existing checking account	EXIST_AC	Categorical	{A11, A12, A13, A14}	A14
2	duration in month	DURTN_MH	Numerical	[4,72]	20.90
3	credit history	CRD_HIST	Categorical	{A30, A31, A32, A33, A34}	A32
4	purpose	PURPOSE	Categorical	{A40, A41, A42, A43, A44, A45, A46, A48, A49, A410}	A43
5	Credit amount	CRD_AMNT	Numerical	[250,18424]	3271.26
6	savings account/bonds	SAV_ACCN	Categorical	{A61, A62, A63, A64, A65}	A61
7	present employment since	EMPLOYMT	Categorical	{A71, A72, A73, A74, A75}	A73
8	installment rate in percentage of disposable income	INS_RATE	Numerical	{1, 2, 3, 4}	2,97
9	personal status and sex	STAT_SEX	Categorical	{A91, A92, A93, A94}	A93
10	other debtors / guarantors	GUARANTR	Categorical	{A101, A102, A103}	A101
11	present residence since	RESIDENC	Numerical	{1, 2, 3, 4}	2,85
12	property	PROPERTY	Categorical	{A121, A122, A123, A124}	A123
13	age in years	AGE_YEAR	Numerical	[19,75]	35.56
14	other installment plans	INS_PLAN	Categorical	{A141, A142, A143}	A143
15	housing	HOUSING	Categorical	{A151, A152, A153}	A152
16	number of existing credits at this bank	EXIST_CR	Numerical	{1, 2, 3, 4}	1,41
17	job	JOB	Categorical	{A171, A172, A173, A174}	A173
18	number of people being liable to provide maintenance for	PEOP_LBL	Numerical	{1, 2}	1.16
19	telephone	TELEPHON	Categorical	{A191, A192}	A191
20	foreign worker	FRN_WORK	Categorical	{A201, A202}	A201
Label		CRD_RISK	Categorical	{bad, good}	good

Table 2.3 Australian Credit Data Attributes

Attributes	Type	Value Range	Mean/Mode
A1	Categorical	{0, 1}	1
A2	Numerical	[13.75,80.25]	31.57
A3	Numerical	[0,28]	4.76
A4	Categorical	{1, 2, 3}	2
A5	Categorical	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}	8
A6	Categorical	{1, 2, 3, 4, 5, 6, 7, 8, 9}	4
A7	Numerical	[0,28.5]	2.22
A8	Categorical	{0, 1}	1
A9	Categorical	{0, 1}	0
A10	Numerical	[0,67]	2.40
A11	Categorical	{0, 1}	0
A12	Categorical	{1, 2, 3}	2
A13	Numerical	[0,2000]	184.01
A14	Numerical	[1,100001]	1018.39
Label	Categorical	{bad, good}	bad

3 PREDICTIVE MODELLING

As discussed earlier data mining has basically two functions: description and prediction. Describing a data set usually means exploring its statistics, understanding distributional properties etc. Prediction, on the other hand, is related to the future or the unknown rather than what we have at hand. As any kind of data analysis project, predictive modelling, starts firstly with collection of data. After exploring the data if it seems necessary data preprocessing and data transformation is also performed. Afterwards, model is trained until reaching a stable classifier algorithm which can find the hidden pattern in the data set. Then comes validating the findings of the model with the facts. After successful completion of the validation step a model becomes ready to deploy, that is, to predict (Imanuel 2016). The structure of the overall predictive modelling is illustrated in Figure 3.1.

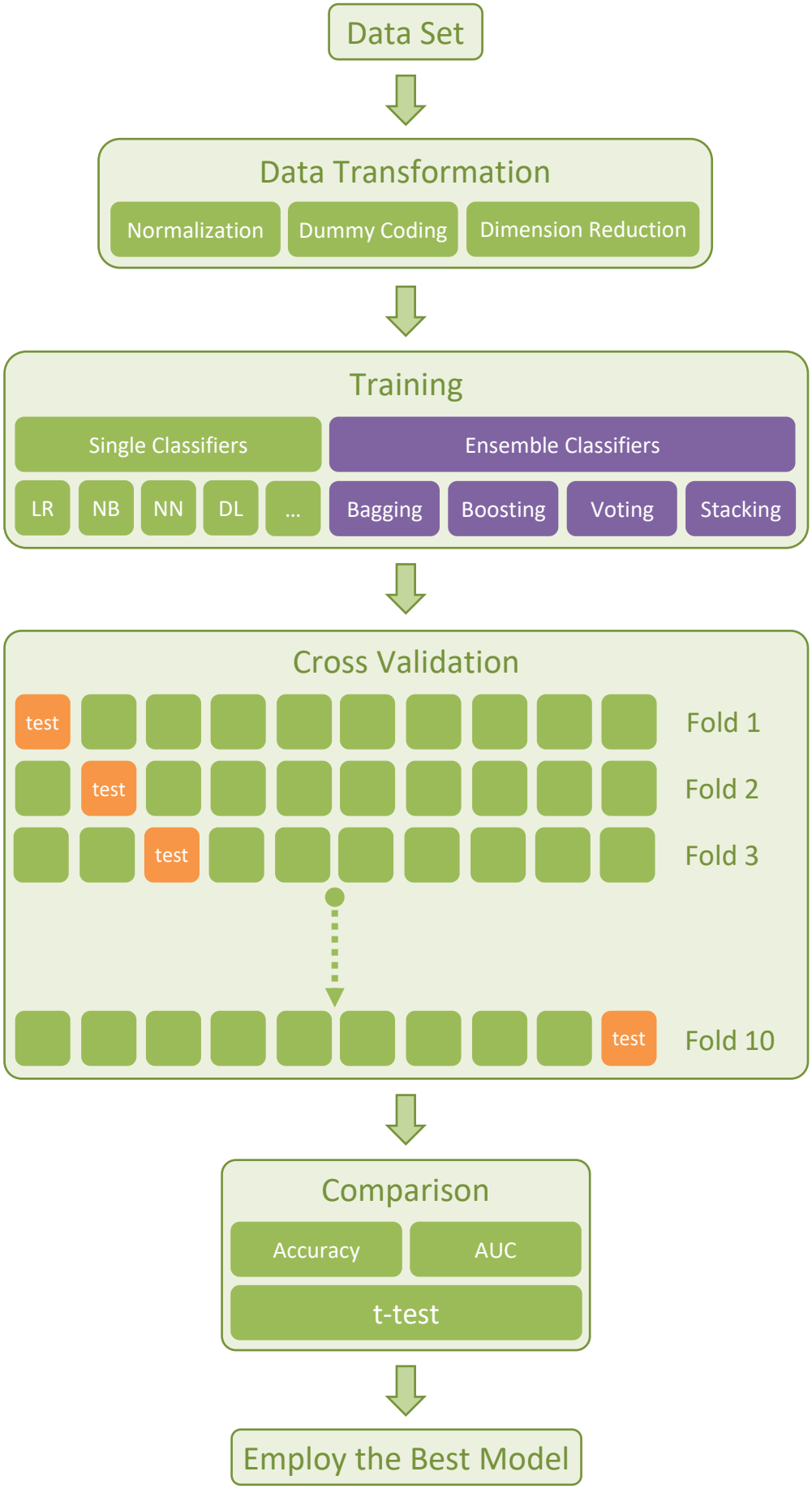

Figure 3.1 Predictive Modelling General Process

3.1 Data Transformation

Transforming the data is often performed to improve the capabilities of data mining models in finding the patterns. The type of transformation required depends on the learning method (unsupervised, supervised etc.) and the data itself (Baesens et al. 2009). Major data transformation methods are missing value imputation, normalization, discretization (binning, coarse categorization), dummy coding, and dimension reduction.

3.1.1 Missing Value Imputation

If there are some attributes in the data set which have missing values, this should be dealt first since models will fail to extract information from those examples. The simplest method would be taking out the attribute with missing values entirely. However, removal of an entire attribute together with its examples which carry information causes the learning process to be incomplete and biased. Instead of removing, missing values can be replaced with pseudo values in order to provide the learning algorithm with a full data set without any loss of information. The first method would be replacing missing values with the averages (or medians/modes depending on the data) of corresponding attributes. This method brings some benefits, but it is very risky also. If the number of missing values are relatively high, then replacement with average values can change the whole distribution of the attribute values and causes the algorithms to learn a different data set than the real one. Thus, less risky and precise methods are needed (Tufféry 2011).

Remedy for this is replacing missing values with statistical approaches and it is called missing value imputation. Depending on the type of the data, there are a couple of accepted techniques to follow. Discussion of these techniques are beyond

this study. Throughout this study whenever it necessary imputation of missing values will be conducted with KNN model as the most widespread statistical solution.

3.1.2 Normalization

Some data mining methods, especially those that are based on the distance computation between examples in n-dimension need normalized data in order to diminish the effects of extreme values and handle different attributes at similar scale. Among normalization methods most widely used one is the normalization based on standard normal distribution, also called standardization. In this method, each value of an attribute is converted according to the equation 3.1, where $\bar{X}$ is the average of the values of an attribute and S_X is the standard deviation of those values.

$$N_i = \frac{X_i - \bar{X}}{S_X} \tag{3.1}$$

After the conversion, the distributions of numerical attributes become similar to normal distribution, thus can be efficiently used by data mining methods which require normality assumptions.

3.1.3 Discretization

When the values of an attribute are in continuous numerical form, sometimes there are benefits of categorizing (binning) those values into finite set of values before the learning process. Though at first it seems losing information, discretization is especially useful in three cases. When missing values exist in the data set and imputation is conducted, it is a tricky situation. Imputation might impair the distribution of the true population. However, through discretization missing values can be categorized into a distinct set and evaluated accordingly. Second, the existence of extreme values affects the learning algorithms remarkably. For instance,

an extreme age value 110 may distort especially regression models. Instead of using this extreme value as it is, categorizing age values and putting this 110 into ">65 years old" class may solve the problem. Why 65? It comes from the domain knowledge of credit decision. Since the common retirement age is around 65 in most countries there is no harm in assuming that retired customers behave similarly and categorizing them into the same group. Finally, discretization improves the robustness of logistic regressions by reducing the possibility of overfitting (Tufféry 2011).

The crucial aspect of discretization is the final number of categories and the cut-off points of these categories. In the literature, three to six final categories are recommended (Mctiernan 2016). Cut-off points can be determined by domain knowledge as in the example above, or by fixing the number of categories, or by distributing the number of values evenly into the categories. Instead of choosing cut-off points, categories can be determined by automatic procedures. For instance, in this study discretization by entropy approach is followed where cut-off points are set so that the entropy is minimized in the categories. After applying this technique, the distribution of CRD_AMNT attribute has changed as shown in the Figure 3.2. The cut-off point 3913 has been automatically chosen which minimizes entropy of classes.

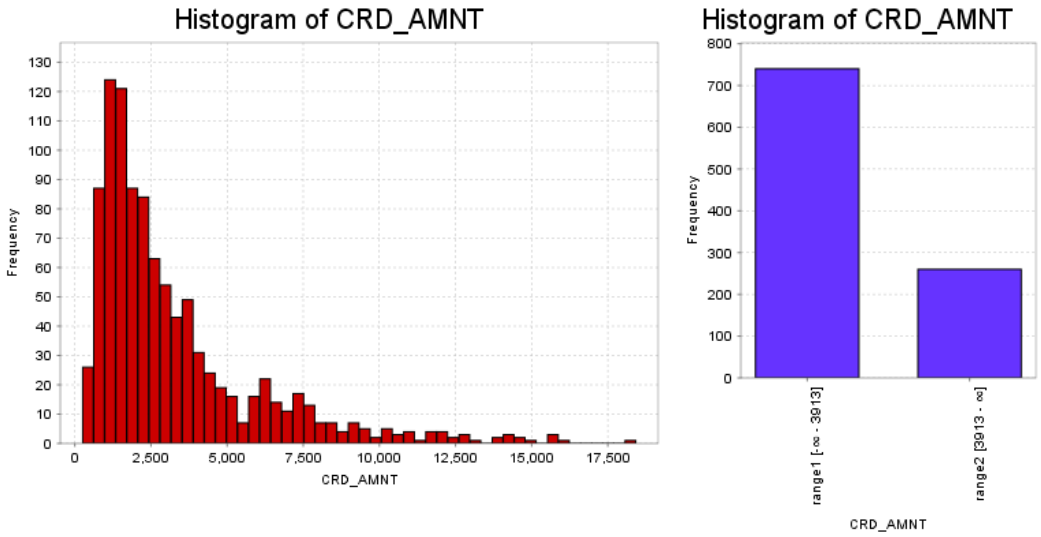

Figure 3.2 Discretization of CRD_AMNT Attribute

3.1.4 Dummy Coding

Some data mining methods like neural networks and support vector machines require the data set to fully in numerical form. In order to utilize these models categorical (nominal) values are converted to numerical values in dummy coding. In this coding style, for every existing value of a categorical variable a new binary attribute is created and take the values of 0 or 1. For instance, in German credit data set, EXIST_AC attribute normally has the values in Table 3.1.a and for each value of this variable new attributes are created in Table 3.1.b.

Table 3.1 Conversion of Categorical to Numerical Values

EXIST_AC
A11
A12
A13
A14

(a)

EXIST_AC = A11	EXIST_AC = A12	EXIST_AC = A13	EXIST_AC = A14
1	0	0	0
0	1	0	0
0	0	0	1
...	...	...	...

(b)

3.1.5 Dimension Reduction

As will be discussed further in the study two of the most successful and widespread data mining methods in credit scoring are logistic regression and support vector machine. Both suffer from high number of dimensions of the data set. The higher the number of attributes in a data set the more possible the model suffers from multicollinearity. Also it will take so much more time for the model to learn (Han et al. 2013). To decrease the number of attributes in a data set two basic approaches can be followed: principal component analysis and stepwise feature selection.

3.1.5.1 Principal Component Analysis (PCA)

The basic idea of PCA is to reduce the dimension number of a data set where there are large number of interrelated variables, while preserving as much as possible of the variation. This reduction is achieved through creation of new attributes from existing ones, which are uncorrelated, independent and ordered in terms of their explanatory power (Tsai 2009). In this study, when PCA is conducted new attributes are kept as long as their total variance is under 0.95. That is, any attribute which explains so little variability in the target attribute excluded from the model.

3.1.5.2 Stepwise Feature Selection

In stepwise feature selection, there are basically two directions along which the selection process can be conducted: forward or backward. Backward selection (elimination) procedure starts with the most complex model, which is including all explanatory attributes. After learning the most complex model and logging its performance measure, remaining attributes are excluded from the learning procedure one by one. The performance of the model which has less explanatory variables is tested against the model which has more explanatory variables through F-test. If reducing the number of explanatory variables significantly improves the overall

performance than those attributes are kept out of the model. Until there is no significant improvement this procedure continues where the model has minimum number of attributes eventually.

Forward selection is exactly the opposite of the backward selection. Attributes are added to the simplest model and larger models are tested against more simple ones. The number of attributes (20) of the data set of this study is relatively high, thus, forward selection will be much more time consuming. Thereby, during feature selection phase backward elimination approach will be followed.

3.2 Training

After exploring and making necessary transformations the data set becomes ready to be used in the training phase. In this phase a sample of examples is drawn from the population whose target values are known. The model tries to extract the hidden pattern between the target feature and the explanatory variables in this phase. After reaching a pattern the model is tested on a sample of examples unseen before whose target values are also known (Tufféry 2011). If the test results are promising the model may be deployed on new data to predict its target value.

The main objective of the training procedure is not only to extract the pattern between output and input variables but also to produce generalizable models which can be performed on new unseen data for predicting purposes. In some cases, especially when the size of the training set is small and the complexity of the model is high, the model might capture the exact representation of this small training set and perfectly predict target values, but it might fail to correctly predict the target values of unseen examples. This phenomenon is one of the fundamental problems of model training and is called overfitting (Giudici & Figini 2009).

3.2.1 Overfitting Problem

In other words, overfitting occurs when the model memorizes the pattern between the output and input variables even though this pattern does not exist in the whole population. Overfitting usually happens when the training data set is small and is especially observed with models decision trees and neural networks. When it happens the model typically fits to the examples too perfectly. It can easily be revealed by testing the model on unseen data. If there is significant discrepancy between the error rate of the training set and test set there is probably overfitting (Tufféry 2011). Graphical representation of an overfitting model can be observed at Figure 3.3. As can be seen, the right model perfectly fits to the training (existing) examples but it loses the ability to generalize. However, the left model has larger errors but it generalizes well. New unseen points will be closer to the left model.

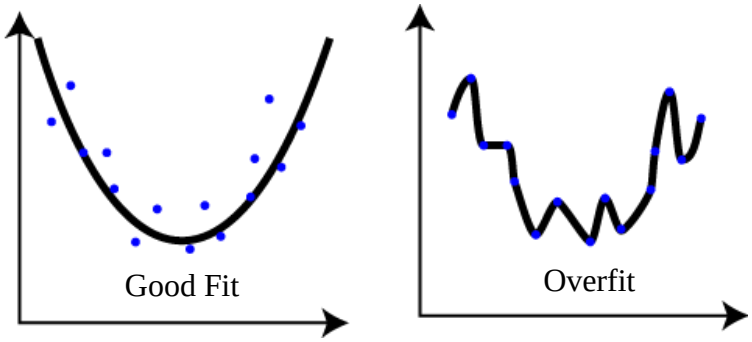

Figure 3.3 Fitting Examples of a Regression Model

Another aspect of overfitting is error rate which corresponds to the complexity of the model in addition to the training size. In data mining, bias means error rate on the training data set and variance denotes how dispersed the model results on the test set. The more complex the model, i.e. the more perfectly it fits, the less bias it will have at the expense of losing generalizability because of the high variance. There is an optimal point where the model has acceptable bias and variance. Figure 3.4 demonstrates this phenomenon. How can one find this optimal point? Although the

true optimal point is hard to find there is a way to validate the generalizability of the model called cross validation (CV) or more specifically k-fold cross validation.

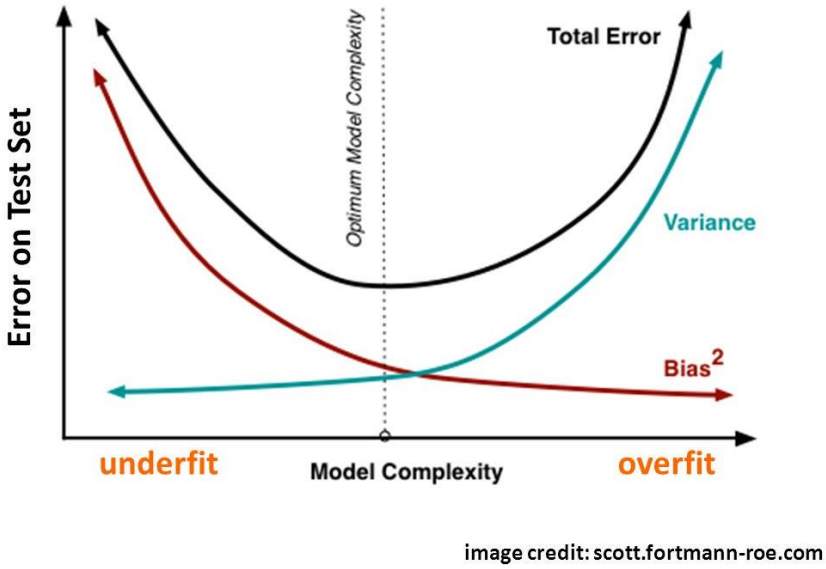

Figure 3.4 Bias-Variance Trade-Off

Source: scott.fortmann-roe.com/docs/BiasVariance.html (Fortmann-Roe 2012)

3.2.2 K-fold Cross Validation (CV)

As discussed in the previous part measuring only the error rate on the training set is not conclusive to estimate true model performance. What is required is to be able to measure the total error rate of the model. To achieve this the error rate of the model must be measured on new, unseen data, ideally more than once (Seni & Elder 2010). This is where cross-validation comes into play. “Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model.” (Refaeilzadeh et al. 2009).

K-fold cross validation is a special form of cross-validation where the whole data set is divided into k nonoverlapping parts (folds). Each of these k parts is used as test set while remaining k-1 sets are used for training. Therefore, every learnt model will be

tested on a different test. After performing training and testing k times the model is evaluated by the average error rate (or performance). The graphical representation of this procedure is illustrated in Figure 3.1. In the cross-validation rectangle each row (ten squares) represents the overall data set and corresponding one training and testing phase. Each orange square is left out as test set and remaining nine green squares are used as one whole training set. This process is iteratively repeated k times. In literature, k is usually set to 10. In this study this custom will be followed and whenever k-fold cross validation is performed it means 10-fold CV.

Validation of predictive capabilities of a data mining model obviously requires standard measurement units. In the next part, performance measures that are used in the evaluation and comparisons of the models will be briefly explained.

3.3 Measures of Performance

In binary problems, models use mathematical functions to map input variables to one of the two classes of the output variable. Most of the time, models' functions assign numerical values to examples possible to order and rank. These numerical outcomes are called confidence. They are basically probabilistic assessments and take values within the range [0,1]. Depending on the business strategy a threshold or cut-off point is chosen and examples are classified according to this threshold. Most of the time, 0.5 is chosen as threshold value. Mathematically,

$$if\ Confidence_i\ (c_i) > Threshold\ (\theta)\ , assign\ positive\ class \quad (3.2)$$

$$if\ Confidence_i\ (c_i) \leq Threshold\ (\theta)\ , assign\ negative\ class \quad (3.3)$$

In the end, each example in the data set has both an actual class value and a predicted class value. The distribution of these values is represented in a two-by-two contingency table called confusion matrix.

3.3.1 Confusion Matrix

The number of examples classified by a binary classification algorithm can be depicted as in the Table 3.2. In credit scoring terminology, positive values denote bad credits (defaults), while negative values denote good credits (healthy). That bad credits take the value positive may seem counter intuitive at first, yet 'positive' signifies the focus of the data mining problem at hand.

Table 3.2 Confusion Matrix

		Actual Classes	
		Good Credit	Bad Credit
Predicted Classes	Good Credit	True Negatives (TN)	False Negatives (FN)
	Bad Credit	False Positives (FP)	True Positives (TP)

From this table a couple of metrics can be calculated each of which conveys distinct meaning with respect to the success of the model. Most widely used accuracy metrics are calculated as follows.

$$Sensitivity = Recall = True\ Positive\ Rate = TPR = \frac{TP}{FN + TP} \quad (3.4)$$

$$Specificity = True\ Negative\ Rate = TNR = \frac{TN}{TN + FP} \quad (3.5)$$

$$False\ Positive\ Rate = FPR = \frac{FP}{TN + FP} = 1 - Specifity \quad (3.6)$$

$$Accuracy = \frac{TP + TN}{TN + FN + FP + TP} = \frac{TP + TN}{Total} \quad (3.7)$$

Sensitivity measures the proportion of positive examples, bad credits in credit scoring that are correctly identified. In credit scoring terms, this is the proportion of defaulted persons that are detected by the model. Specificity measures the proportion of negative cases that are correctly identified. In credit scoring terms, these are the examples of good credits correctly classified by the model. False positive rate represents the proportion of examples that are classified as default incorrectly. In statistical terms, it is called Type I error. Finally, accuracy measures the overall success of the model. It is the proportion of correctly classified cases, either default or not, to total cases.

3.3.2 Accuracy, ROC & AUC

Although accuracy measurement demonstrates how successful a model is in binary classification, the resultant distribution tabulated in the confusion matrix depends on the threshold chosen in the first place. Intuitively, θ is fixed at 0.5. But, strategically the selection of the threshold value depends on the cost misclassified cases bring about. For instance, in credit scoring scheme granting credit to a bad borrower (false negative) is worse than rejecting an applicant who is actually a good borrower (false positive). Therefore, threshold selection is very crucial and the effect of this selection on the overall accuracy of the model must be observed.

Mathematically, threshold value can be anything in the range (0,1). Hence, the number of the distributions of different classifications is practically uncountable. There exists a graphical tool which solves this problem which is called receiver operating characteristic (ROC). The name comes from the field where this tool is first created. In data mining, ROC acronym has widespread usage. ROC graph is basically a collection of success rates of a model for every possible threshold values. It illustrates the trade-off between true positive rate (sensitivity) and false positive

rate (1-specificity). A typical ROC graph can be observed in the Figure 3.5. The perfect model flawlessly discriminates two classes. For instance, (0,1) point means the model catches all defaults while making no mistake in healthy examples as well. Random model has no discriminatory power at all. The model outcomes in between the perfect and random model discriminate at varying degrees depending on the threshold value. Along the ROC curve every possible success rate couples are shown for every threshold values (Khudnitskaya 2010).

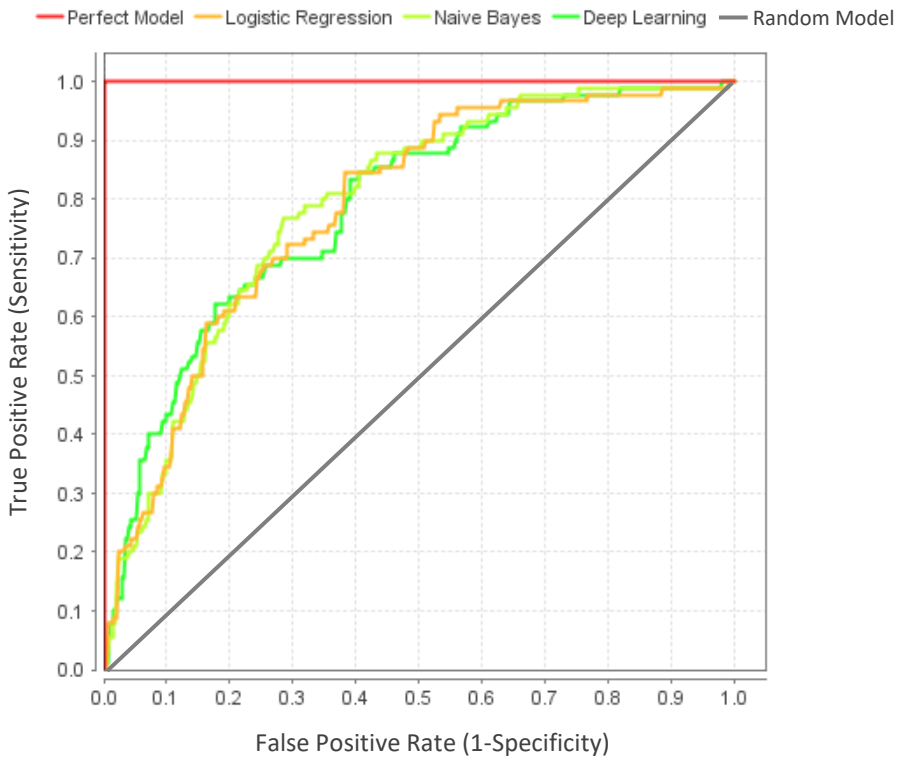

Figure 3.5 ROC Graph Example

The nearer a ROC curve to the upper left corner the more successful it is in discriminating classes. Since visual comparison does not provide conclusive result, we need scalar accuracy values to compare. The solution for that is calculating the area under the ROC curve. It is called AUC and is probably most widely used performance metric in binary classification problems. AUC will be primary

measurement metric throughout this study also. AUC value makes class distribution based on different threshold values irrelevant. Since the ROC graph is drawn on a unit square the area under the hypothetical perfect model sums up to 1. The area under the random model is 0.5. Therefore, it is expected from any binary classification model to have at least 0.5 AUC and as closer as possible to 1 (Kraus 2014).

3.3.3 Pairwise T-Testing

In order to evaluate performance outcomes of different models an accurate comparison system is also required. Because, one model outcome does not represent its true population mean and several trials will yield different results while converging the true performance of the model. So, if one performance outcome of a model is compared to another one the assessment might be wrong. To compare model outcomes consistently, t-test will be applied on AUC values. T-test is a statistical comparison technique which allows to test whether two groups of values comes from the same population. In other words, in credit scoring performance, t-test will allow us to distinguish only significantly successful data mining models.

4 EXPERIMENTATION

The process of the experimentation involves four phases: making necessary data transformations to improve model accuracies, setting the software environment to let the models learn on the training data and logging the performance results, comparing the models to each other, and finally building the ensemble model. To complete the first three phases efficiently a reproducible program is created, which can handle almost any kind of data and make necessary transformations, called model comparison engine (MCE).

4.1 Model Comparison Engine

As discussed in the latest chapter in most cases data needs to be cleansed and transformed in order the models to fully utilize it. In this study, since some models require the data to be in numerical/categorical form and some has comparative advantages on some data types and we wanted to create an exhaustive comparison universe where each model will be able to have all necessary conditions to reach its true potential and to maximize its performance results.

The first part of MCE checks the properties of the data set. If the data set has missing values in it, those missing values are replaced by missing value imputation technique described in section 3.1.1. However, at the end of this step the original data set which has missing values is not thrown away, rather two data sets are kept: (1) with missing values, (2) without missing values. The main reason to keep the original data set with missing values is to give the opportunity to the models which can handle missing values.

Since there are models based on the distance between data points, like support vector machines which try to find optimal lines or surfaces which divides the data set in the

n-dimensional space, the scale of the numerical data, thus, normalization is also important. In the second step, all numerical values are normalized by the method explained in the section 3.1.2. Again, the data set which has nonnormalized numerical attributes is not neglected, it will be kept together with normalized data set. Until now, if the data set has required missing value imputation also, we will now have four different data sets. This is the central idea of model comparison engine.

The next step is to create discrete categories out of numerical values. The underlying reason, method and possible benefits of discretization has been discussed in section 3.1.3. After completing discretization, all categorical variables are converted to numerical variable in dummy coding form described in section 3.1.4. As a last step dimension reduction is performed on all the attributes, without making any distinction between attributes which exist in the original data set and attributes created afterwards.

There are some steps which need more clarification. As discussed before, some models like support vector machines and neural networks require data set to be in numerical form. This was the main reason of dummy coding. Yet the sequential application of data transformations causes some meaningless data sets to emerge. For instance, a data set which is already in numerical form can be discretized and codded in dummy form afterwards. This obviously will cause data loss. Since designing the algorithm in RapidMiner accordingly is costly, all kinds of transformations applied regardless of their logical base. Corresponding results have less accuracy and therefore do no impair the general evaluation of the study.

After the data transformation is completed we might have 64 or less distinct data sets depending on the properties of the initial data set. Figure 4.1 illustrates the data transformation steps and illuminates in visual representation how the number of data sets increase exponentially. In this figure each rectangle represents a distinct data set from which models try to extract hidden pattern.

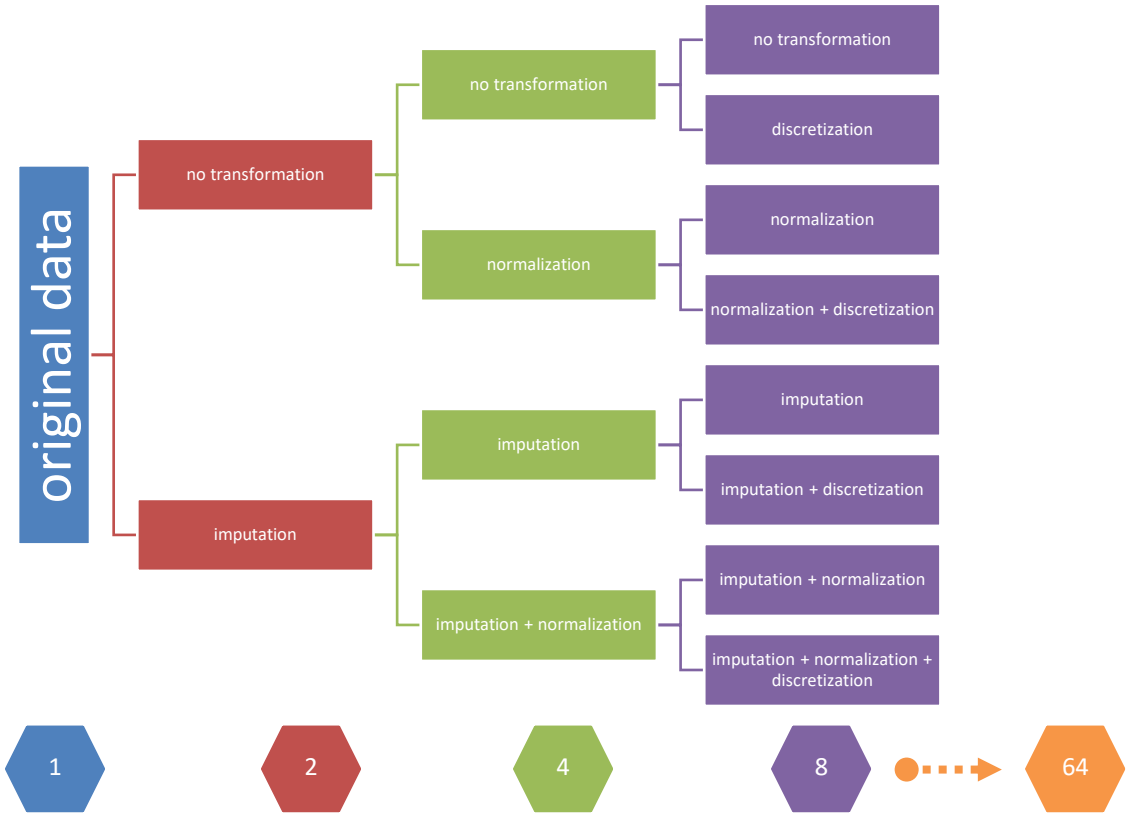

Figure 4.1 Permutations of Data Transformations and Number of Possible Data Sets

The resultant data sets of this transformation phase are fed to the learning algorithms. Similar to the transformation steps, in this study we wanted to try as many models as possible on every distinct data sets. To this end, learning algorithms of almost any kind are listed and grouped under classifier families. Since detailed descriptions and explanation of underlying mapping functions of these models are beyond the scope of this study, larger classifier families will be explained briefly.

4.2 Classifier Families

As discussed earlier in this study for data analytics purposes RapidMiner is used as software. RapidMiner has 43 built-in single models ready to deploy with default parameters. Each of this model is included in model comparison engine with default settings and grouped under a classifier family. Models and their corresponding classifier families are listed in Table 4.1.

Table 4.1 RapidMiner Classifier Families & Models

No	Classifier Families	Model Name
1	**Functions (Regressions)**	Gaussian Process
2		Generalized Linear Model
3		Linear Regression
4		Local Polynomial Regression
5		Polynomial Regression
6		Relevance Vector Machine
7		Seemingly Unrelated Regression
8		Vector Linear Regression
9	**Support Vector Machines**	Fast Large Margin
10		Hyper Hyper
11		Support Vector Machine
12		Support Vector Machine (Evolutionary)
13		Support Vector Machine (LibSVM)
14		Support Vector Machine (Linear)
15		Support Vector Machine (PSO)
16	**Logistic Regressions**	Logistic Regression
17		Logistic Regression (Evolutionary)
18		Logistic Regression (SVM)
19	**Bayesian Classifiers**	Naive Bayes
20		Naive Bayes (Kernel)
21	**Decision Trees**	CHAID
22		Decision Stump
23		Decision Tree
24		Decision Tree (Multiway)
25		Decision Tree (Weight-Based)
26		Gradient Boosted Trees
27		ID3
28		Random Forest
29		Random Tree
30	**Neural Networks**	AutoMLP
31		Deep Learning
32		Neural Net
33		Perceptron
34	**Discriminant Analysis**	Linear Discriminant Analysis
35		Quadratic Discriminant Analysis
36		Regularized Discriminant Analysis
37	**Lazy**	Default Model
38		k-NN
39	**Rules**	Rule Induction
40		Single Rule Induction
41		Single Rule Induction (Single Attribute)
42		Subgroup Discovery
43		Tree to Rules

So far, if each data transformation step is performed then there might be 64 distinct data sets. As the main strategy is trying as many data-model pairings as possible, each of these 64 data sets will be fed to each single model listed. Model comparison engine will try $2752\ (43 * 64)$ possible pairings during this phase and let the models learn the underlying pattern. The overall procedure lasts 20 minutes to 10 hours depending on the number of dimensions, number of examples and complexities of the models applicable. If the pairing causes an error, for instance when categorical variables are fed to support vector machine, MCE will log this error and pass the next trial. In the end, the performance values of every pairings which do not have any error will be logged and saved for comparison. In the coming parts, important classifier families will be briefly explained.

4.2.1 Functions (Regressions)

Data mining methods which belong to function or regression family were briefly reviewed in section 2.2.2.1. Regressions are basically used to relate output variable to input variables through a mathematical equation. This equation not only helps to predict target values of new input variables but also provides an understanding about the nature of the relationship between output and input variables. In its simplest form regression equation is linear and hence called linear regression. The equation below is a simple example of a linear regression.

$$Y = \beta_0 + \sum \beta_i * X_i + \varepsilon_i \qquad (4.1)$$

Here, X_i denotes the matrix of input variables, β_i denotes vector of coefficients for input variables, β_0 denotes constant coefficient and ε_i denotes error term. β_0 and β_i are calculated through an optimization method called ordinary least squares which minimizes the sum of

squared errors. Figure 2.1 depicts the line of this equation in its simplest form where there is only one input variable.

4.2.2 Support Vector Machines (SVM)

Compared to the other data mining classification methods, support vector machines (SVM) are newly developed in the nineties by Vladimir Vapnik. SVM is especially useful on linearly separable examples. When, however, data points are clustered in n-dimensional space there might be infinite number of linear boundaries which separate those clusters. What SVM does essentially is to find the optimal hyperplane that maximizes the width of the margin between observation clusters (Tufféry 2011). Figure 4.2 shows how the optimal hyperplane separates different classes by widest possible margin.

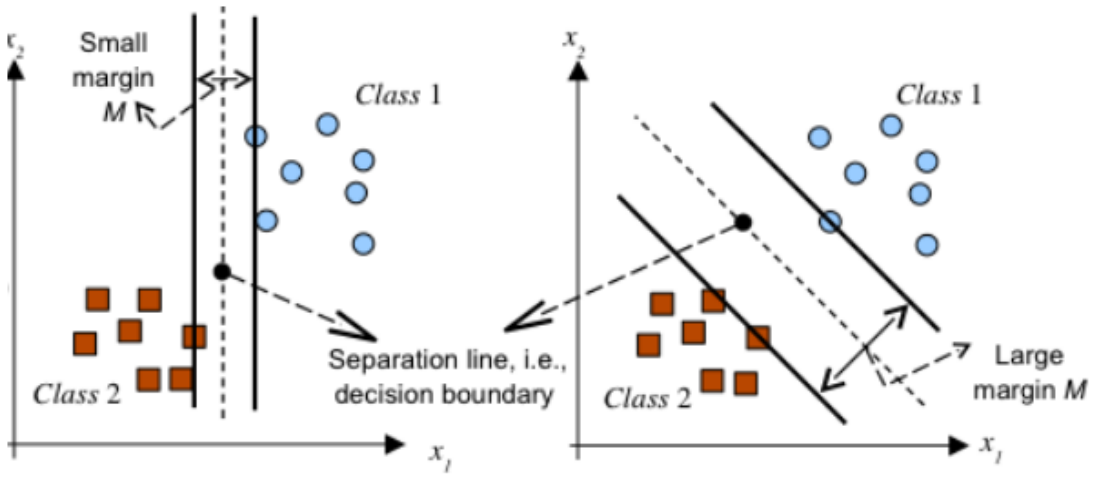

Figure 4.2 Support Vector Machine Hyperplane

Source: yottamine.com/machine-learning-svm (Yottamine Analytics 2013)

There are a couple of advantages of using SVM over other data mining algorithms. First, training process is comparatively easy with even small number of parameters and the final model is not presented with a local optimum, unlike some other techniques. Second, SVM scales to high dimensional data relatively well. Some of the recent and most successful applications of SVM have been in image processing,

particularly hand written letter recognition and face recognition (Kantardzic 2011). One of the aspects of SVM which is often criticized is its opaque methodology (Harris 2015).

4.2.3 Logistic Regressions

As one of the oldest classification methods in statistics, logistic regression is the most widely used classification technique in credit scoring (Huang & Day 2013). Its popularity is not without reason. It can handle dependent variable with two or more values and independent variables could be numerical or categorical. The results of logistic regression are clear and interpretable, that is, results have practical meaning especially in credit scoring as probability of default. Finally, logistic regression is one of the most reliable classification methods (Tufféry 2011).

Within the context of credit scoring the aim of using logistic regression is to detect whether counterparties (whether real people or companies) will pay their debts in time. The event of nonpayment or (depending on the problem setting) restructuring of the debt is denoted by positive (bad) class. Whereas, the event where everything goes normal and the borrower pays the debt promptly is denoted by negative (good) class. The fundamental equations of logistic regression are as follows:

$$\Pr(Y = "bad"|X) = \frac{e^{\beta_0+\sum_i \beta_i * X_i}}{1 + e^{\beta_0+\sum_i \beta_i * X_i}} \tag{4.2}$$

$$\Pr(Y = "good"|X) = \frac{1}{1 + e^{\beta_0+\sum_i \beta_i * X_i}} \tag{4.3}$$

In the first equation, what the logistic regression model produces the conditional probability of a nonpayment, given that some explanatory variables exist and their values are known, which are denoted by X. Figure 4.3 demonstrates geometrical

representation of a logistic regression equation, where the value logit (L) along the horizontal axis is calculated by a linear equation of X values, $L = \beta_0 + \sum_i \beta_i * X_i$. If the logit value is used in the fundamental equation instead of X values, the main equation becomes

$$\Pr(Y = "bad"|X) = \frac{e^L}{1 + e^L} \tag{4.4}$$

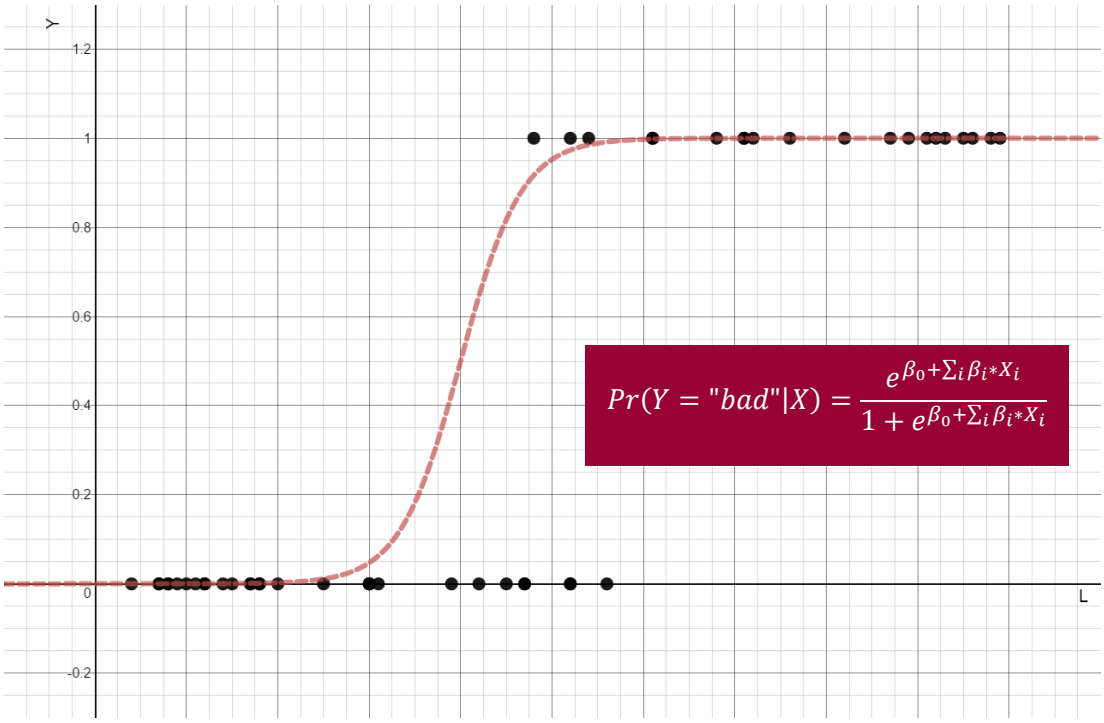

Figure 4.3 Logistic Regression Example

4.2.4 Bayesian Classifiers

Bayesian classification method is also one of the simplest and most widely used probabilistic learning methods. What it learns from the training data is the conditional probability of each attribute A_i where the class label C is known. The strong major assumption is that all attributes are independent. The assumption of conditional independence of variable is very crucial in order for this method to produce robust results (Twala 2010).

4.2.5 Decision Trees

Decision trees are arguably the most intuitive and popular data mining method among others. It provides explicit and easy rules for classification. These rules are easy both to implement on new examples and to present the findings to third parties. Decision trees handle some of the challenging problems of data mining very well also. It can operate on heterogeneous data. It is not affected by missing values and also non-linearity of input variables because it does not rely on the distributional assumptions (Tufféry 2011).

Despite its easiness and popularity decision trees produce relatively poorer results than other data mining methods in credit scoring. This is mainly because of two reasons: (1) The noise and (2) the redundant variables in the data affect the performance of decision trees (Wang et al. 2012).

Decision trees, as the name suggests, a tree-like top-down structure which has root, nodes, branches, and leaves. Each internal node represents a test on one attribute and each branch denotes the outcome of this test. The leaves (terminal nodes) represents either a pure class or a class distribution. The top node in the tree is the root node with the highest information gain. After the root node, among the remaining attributes the one with the highest information gain is chosen and test is performed on this node. The overall process continues until there is no remaining attribute on which example set can be partitioned (Tsai et al. 2014). A decision tree algorithm performed on the German credit data set is presented in Figure 4.4.

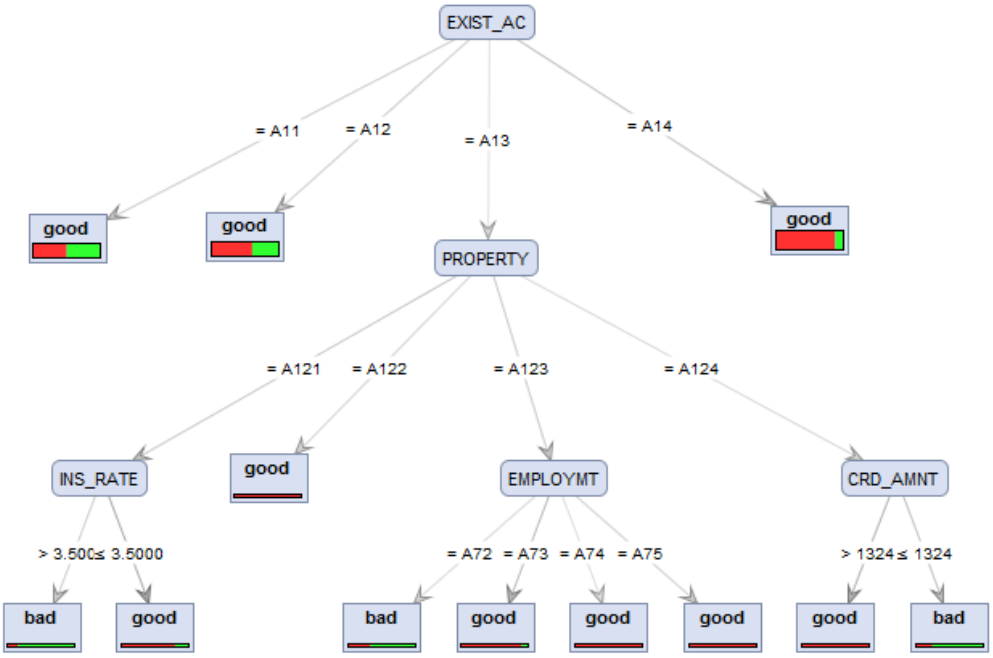

Figure 4.4 Decision Tree on German Credit Data

4.3 Ensemble Models

Ensemble models combine single models into a model which is hoped to be more successful than its constituent models. The idea is that every model does not have the same error for a specific example (observation, data point) and they complement each other. Ensemble models have gained widespread usage since they are deemed the most profound development in data mining field in the last decade (Seni & Elder 2010). Especially in credit scoring, where the predictive capability is more important than interpretability of the model, ensemble models provide significant gains over single models' prediction accuracies.

Ensemble models can be categorized into two broader classes: homogeneous ensemble classifiers and heterogeneous ensemble classifiers. Homogeneous ensemble classifiers combine the outcomes of the same base model. One of the most widely used such classifiers is bagging (bootstrap aggregating) where independent

small samples is drawn from the whole data set with replacement and the base model is trained on each one of them. In the end, bagging method combines all predictions. Unlike bagging, in boosting algorithm, constituent models depend on each other. Every next iteration of the boosted algorithm learns from the mistakes of the previous one.

Heterogeneous ensemble classifiers combine multiple different single models to create the final more successful model. Basically there are two heterogeneous models: stacking and voting. In stacking method outcomes of each base models are collected and together with existing examples (instances) a new data set is created. Usually logistic regression with regularization is learnt on this final data set. In voting scheme, the outcomes of single models for every example are compared and the final decision is made either by majority voting or weighted voting (Wang et al. 2011). The most recent studies which compare accuracies of different ensemble models point to the weighted voting as the most successful ensemble model (Lessmann et al. 2015). In order to keep the focus of the study on finding the most successful models automatically and dynamically, ensemble models other than weighted voting have not been employed in this study.

4.4 Weighted Voting

In binary classification problems, the first outcome of the models is the confidence value which shows how confident the model is in predicting the class of an example. In credit scoring this confidence value is usually interpreted as the probability of default of the examined entity. In other words, the confidence value indicates how likely the counterparty fails to pay its debt on time. Weighted voting ensemble model takes the average of the class predictions of single models using their confidence

values as weights of the averaging. That way, it is expected to reach a more successful model.

Yet, the question of which single models to be selected in building the ensemble model remains unanswered. Since there are 43 built-in models in RapidMiner the number of possible combinations which can be used in the ensemble model is $8.8^{12} = 2^{43}$. The time and computational resources required to finish such a task is beyond the limit ordinary computers can handle. Until now, almost every kind of data transformation has been applied and each built-in model in RapidMiner is learnt on these transformed data sets and resulting accuracies are compared. The steps up to here are automatically conducted and dynamically adjusted without any human interaction. Since one of the main objectives of the study is completing the overall credit scoring system without any human interaction, an objective selection criterion is required to choose single models to be used in weighted voting.

4.5 Objective Selection Criterion

As discussed in previous sections the idea behind ensemble models is that single models have diverse inner functions and different input-output mappings. Therefore, when especially dissimilar but successful models are combined into an ensemble model, it is expected that they will complete each other well and the accuracy gain will be higher. To test whether this hypothesis correct or not the ten most successful models which are delivered by model comparison engine are gathered. Out of this selection, model sets consisting of two models are combined into the weighted voting ensemble model. Eventually there were $45 = \binom{10}{2}$ sets to be tested. All of these sets

are trained on German[3] data set and the outcomes are sorted by confidence correlations of single models.

The assumption was that less correlated models are dissimilar to each other and they complement each other better, thus the improvement in the accuracy over the single models will be higher. Improvement is defined as the difference between the ensemble AUC and maximum of single AUCs. As can be seen in Figure 4.5, the results are not supporting this hypothesis. Expectation was detecting a pattern in the improvement column linked with correlation column. Although model couples are sorted by their correlations there is no detectable pattern along the improvement column. It looks almost the result of a random distribution. Hence, we failed to find an objective selection criterion which will automatically select successful models of MCE and pass those models to the next phase where ensemble models are built. Therefore, credit scoring system will have two successive engines which are bonded through human interaction. The first engine was model comparison engine. The second engine is ensemble model building engine (EMBE).

[3] The search for objective selection criterion has been conducted on Australian and some other data sets as well. The results are almost the same.

Base Model 1	Base Model 2	Correlation	AUC1	AUC2	AUC Final	Improvement
Model_6	Model_9	0,857	0,9360	0,9308	0,9369	0,0009
Model_8	Model_9	0,859	0,9308	0,9308	0,9392	0,0084
Model_9	Model_10	0,859	0,9308	0,9308	0,9307	-0,0001
Model_7	Model_9	0,874	0,9349	0,9308	0,9370	0,0021
Model_1	Model_8	0,890	0,9392	0,9308	0,9390	-0,0002
Model_1	Model_10	0,890	0,9392	0,9308	0,9337	-0,0055
Model_2	Model_8	0,892	0,9321	0,9308	0,9376	0,0055
Model_2	Model_10	0,892	0,9321	0,9308	0,9327	0,0005
Model_4	Model_9	0,892	0,9356	0,9308	0,9363	0,0007
Model_3	Model_9	0,893	0,9357	0,9308	0,9378	0,0021
Model_5	Model_9	0,895	0,9359	0,9308	0,9324	-0,0035
Model_1	Model_6	0,917	0,9392	0,9360	0,9382	-0,0010
Model_6	Model_8	0,918	0,9360	0,9308	0,9381	0,0021
Model_6	Model_10	0,918	0,9360	0,9308	0,9384	0,0024
Model_2	Model_9	0,919	0,9321	0,9308	0,9371	0,0050
Model_1	Model_7	0,921	0,9392	0,9349	0,9334	-0,0058
Model_1	Model_9	0,925	0,9392	0,9308	0,9397	0,0005
Model_4	Model_8	0,929	0,9356	0,9308	0,9357	0,0001
Model_4	Model_10	0,929	0,9356	0,9308	0,9346	-0,0011
Model_3	Model_8	0,932	0,9357	0,9308	0,9377	0,0020
Model_3	Model_10	0,932	0,9357	0,9308	0,9304	-0,0053
Model_1	Model_5	0,937	0,9392	0,9359	0,9340	-0,0052
Model_1	Model_4	0,941	0,9392	0,9356	0,9395	0,0003
Model_1	Model_3	0,942	0,9392	0,9357	0,9393	0,0001
Model_2	Model_7	0,948	0,9321	0,9349	0,9354	0,0005
Model_2	Model_6	0,950	0,9321	0,9360	0,9317	-0,0043
Model_5	Model_8	0,951	0,9359	0,9308	0,9371	0,0012
Model_5	Model_10	0,951	0,9359	0,9308	0,9301	-0,0058
Model_1	Model_2	0,954	0,9392	0,9321	0,9390	-0,0002
Model_7	Model_8	0,956	0,9349	0,9308	0,9362	0,0013
Model_7	Model_10	0,956	0,9349	0,9308	0,9345	-0,0004
Model_2	Model_5	0,968	0,9321	0,9359	0,9310	-0,0049
Model_2	Model_4	0,972	0,9321	0,9356	0,9339	-0,0017
Model_2	Model_3	0,972	0,9321	0,9357	0,9382	0,0025
Model_4	Model_7	0,976	0,9356	0,9349	0,9355	-0,0001
Model_5	Model_6	0,978	0,9359	0,9360	0,9377	0,0016
Model_3	Model_7	0,978	0,9357	0,9349	0,9377	0,0020
Model_4	Model_6	0,980	0,9356	0,9360	0,9347	-0,0013
Model_3	Model_6	0,981	0,9357	0,9360	0,9375	0,0015
Model_6	Model_7	0,984	0,9360	0,9349	0,9353	-0,0007
Model_5	Model_7	0,986	0,9359	0,9349	0,9383	0,0024
Model_4	Model_5	0,994	0,9356	0,9359	0,9332	-0,0027
Model_3	Model_5	0,995	0,9357	0,9359	0,9388	0,0029
Model_3	Model_4	1,000	0,9357	0,9356	0,9314	-0,0043
Model_8	Model_10	1,000	0,9308	0,9308	0,9321	0,0013

Figure 4.5 Improvements of Ensemble Models by Correlation

4.6 Ensemble Model Building Engine

The search for an objective selection criterion did not produce positive results. The reason of this search was massive number of possible combinations (8.8^{12}) in ensemble modelling. To decrease the number of possible trials to manageable size, 6 successful models out of 43 are selected manually by inspection. These 6 models are not selected solely based on their performance values, but rather chosen carefully from different model families to increase the complimentary powers of single models.

Similar to the approach in the model comparison engine 57 ($2^6 - \binom{6}{1} - \binom{6}{0} = 64 - 6 - 1$) different combinations of these 6 successful models have been combined into weighted voting ensemble model and each one of 57 ensemble models is trained again on 64 distinct data sets. Eventually, there were $3648 = (57 * 64)$ trials. Resultant performance measures are logged and compared.

5 RESULTS & CONCLUSION

5.1 Comparison of Single Models

As discussed earlier, model comparison engine produces every possible data-model pairing. For the German credit data, since the data set does not have any missing value, data transformation steps except missing value imputation have been performed. At the end, there were thirty-two (32) distinct transformed data sets. From these data sets forty-three (43) models are learnt. In total, 1,376 pairs have been tried. Among these trials, 621 pairings did not have errors and provided performance metrics. Out of these results the first twenty ones with highest performance results are presented in Table 5.1. The whole result table is presented in Appendix C: Performance Results of Data-Model Pairings on German credit data set.

Although not statistically significant the first thing that stands out in this table is that top places are all occupied by support vector machine models. When we look closely the data sets when SVM models are successful all involve the transformation steps of discretization and numerical conversion. Numerical conversion is already compulsory for SVM models. Since SVM method is based on distance between data points, discretization might have helped to emphasize the different places of examples in space. In addition, normalization also contributes to the models based on distance measurements. Thus, the presence of normalization in the top places is also not a surprise.

Figure 5.1 illustrates average AUC performance of all models across different data transformations. Again, the effect of discretization and numerical conversion is noteworthy. This can be regarded as a rule of thumb in future credit risk analysis. One other remarkable thing is the ranking of no transformation case. This implies

that ignoring data transformation, believing that the data is in good shape, simply does not work.

Figure 5.2 shows maximum attained AUC values across data transformations. It is already clear from the table that these values are achieved by SVM models. Moreover, discretization and numerical conversion are still leading transformation types.

Figure 5.3 illustrates the average AUC values of classifier families. It is clear that functions and Bayesian classifiers are more general tools than others. So, when underlying data structure and attribute types are not fully known one can easily apply linear regression or Naïve Bayes method. Despite their simplistic assumptions, they are still powerful tools.

Figure 5.4 illustrates the maximum AUC values of classifier families. Although SVM model attains the highest performance value, the improvement is not statistically significant, but it is a fact hard to ignore. In second place, functions protect their place and logistic regression family is very close. This graph also shows that the widespread usage and reputation of logistic regressions is not by chance.

Figure 5.5 illustrates average AUC values of every single model (to emphasize the discrepancies between models, figure is focused above 0.75 AUC and thus some models are not shown). Since there are many different data transformations this graph depicts how powerful a single model is in general. Unsurprisingly, logistic regression and generalized linear model[4] are powerful methods in general.

[4] which is pretty much the same thing as logistic regression within this context.

Figure 5.6 illustrates maximum AUC values of every single model (to emphasize the discrepancies between models, figure is focused above 0.795 AUC and thus some models are not shown). Unlike average comparison, here models have the opportunity to reach their full potential since many different data setting is available. Support Vector Machine (LibSVM) model has the highest AUC. Though it is not significant, SVM models seem promising especially when the model criteria and parameters are tuned and optimized to reach higher levels of success. Linear and logistic regression models again prove that their popularity is earned.

Figure 5.7 presents the confusion matrices of the three most successful models on German credit data set. The overall accuracy rates follow the same order as in the AUC performances. Support vector machine seems to be the best model again. However, depending on the cost structure of the organization, linear regression and fast large margin model are also very good candidates since they have higher recall (sensitivities) for "bad" classes. If, for instance, an organization is exposed to more cost by extending credit to bad borrowers than rejecting credit applications of good borrowers, which is usually the case in real business life, than credit analysts should choose fast large margins over the other two models. This situation reveals again the fact that definition of the performance values must be aligned to the problem at hand.

For the Australian credit data, since the data set does not have any missing value and all categorical values are coded in integer form, data transformation steps except missing value imputation and numerical encoding, have been performed. At the end, there were sixteen (16) distinct transformed data sets. From these data sets forty-three (43) models are learnt. In total, 688 pairs have been tried. Among these trials, 316 pairings did not have errors and provided performance metrics. Out of

these results the first twenty ones with highest performance results are presented in Table 5.2.

When the performance values of transformation types and model families are compared on Australian data set outcomes are not notably different than the ones in German data set. However, when it comes to the single model comparison the first rows of Table 5.2 and Figure 5.8 give surprising results about Australian credit data set. Gradient Boosted Trees shows a remarkable performance, significantly higher than the second (deep learning) and third models. This success is probably attributable to the nature of the data set. Australian data set has numerical and categorical features, but its categorical variables are coded in integer values, thus can be used as ordinal values if they are in that form already. Because of the anonymity of Australian data set we cannot know for sure. Gradient Boosted Trees is in very simple terms empowered decision tree algorithms which learns from its mistakes and boosts the model. Decision trees are good at splitting ordinal values.

In summary, two publicly available credit data sets have been evaluated with various data mining methods in this study. To this end, a model comparison engine is developed in RapidMiner environment and this engine is tuned as by-product for future uses as well. For the German data set, although SVM models bring small gains, traditional linear and logistic regressions seem very powerful and generalizable across different data sets. Moreover, in the case of German data set, discretization and numerical conversion appear to have positive effects. However, one should not easily conclude that researchers and/or risk analyst could prefer using already popular logistic regression models. As observed in the case of Australian data set, the nature of the Australian data set brings about an unconventional model,

Gradient Boosted Trees, to be successful by significant margin. In addition, discretization does not bring gain readily in Australian data case. It is maybe because the data is already well discretized or the nature of the data is not suitable for discretization. Credit risk decision is primarily based on the data collected. In the end, researchers or practitioners alike, should be cautious about the nature of the data and try to understand strong or weak aspects of the models. Conducting as many models as possible could help as a deductive approach as long as time or other resource permit.

Table 5.1 First 20 Performance Results of Data-Model Pairings on German credit data set

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc + num + OS	Support Vector Machines	Support Vector Machine (LibSVM)	57	941	0,7978	0,0294
disc + num + OS	Support Vector Machines	Support Vector Machine (LibSVM)	57	951	0,7978	0,0294
norm + disc + num	Support Vector Machines	Support Vector Machine (LibSVM)	58	954	0,7978	0,0297
disc + num	Support Vector Machines	Support Vector Machine (LibSVM)	58	957	0,7978	0,0297
norm + disc + num + PCA + OS	Functions	Linear Regression	28	1391	0,7973	0,0291
disc + num + PCA + OS	Functions	Linear Regression	28	1451	0,7973	0,0291
norm + num + PCA	Support Vector Machines	Fast Large Margin	32	550	0,7963	0,0264
disc + num + PCA + OS	Functions	Generalized Linear Model	28	264	0,7960	0,0263
norm + disc + num + PCA + OS	Functions	Generalized Linear Model	28	270	0,7960	0,0263
norm + disc + num + PCA + OS	Logistic Regression	Logistic Regression	28	270	0,7959	0,0262
disc + num + PCA + OS	Logistic Regression	Logistic Regression	28	272	0,7959	0,0262
norm + disc + num + PCA + OS	Logistic Regression	Logistic Regression (SVM)	28	1081	0,7959	0,0262
disc + num + PCA + OS	Logistic Regression	Logistic Regression (SVM)	28	1083	0,7959	0,0262
norm + disc + num + PCA + OS	Support Vector Machines	Fast Large Margin	28	179	0,7958	0,0284
disc + num + PCA + OS	Support Vector Machines	Fast Large Margin	28	181	0,7958	0,0284
norm + num + PCA	Logistic Regression	Logistic Regression	32	290	0,7957	0,0259
norm + num + PCA	Functions	Generalized Linear Model	32	276	0,7957	0,0260
norm + num + PCA	Logistic Regression	Logistic Regression (SVM)	32	1219	0,7955	0,0261
norm + num + PCA	Functions	Linear Regression	32	1883	0,7944	0,0303
norm + disc	Bayesian	Naive Bayes	15	36	0,7942	0,0220

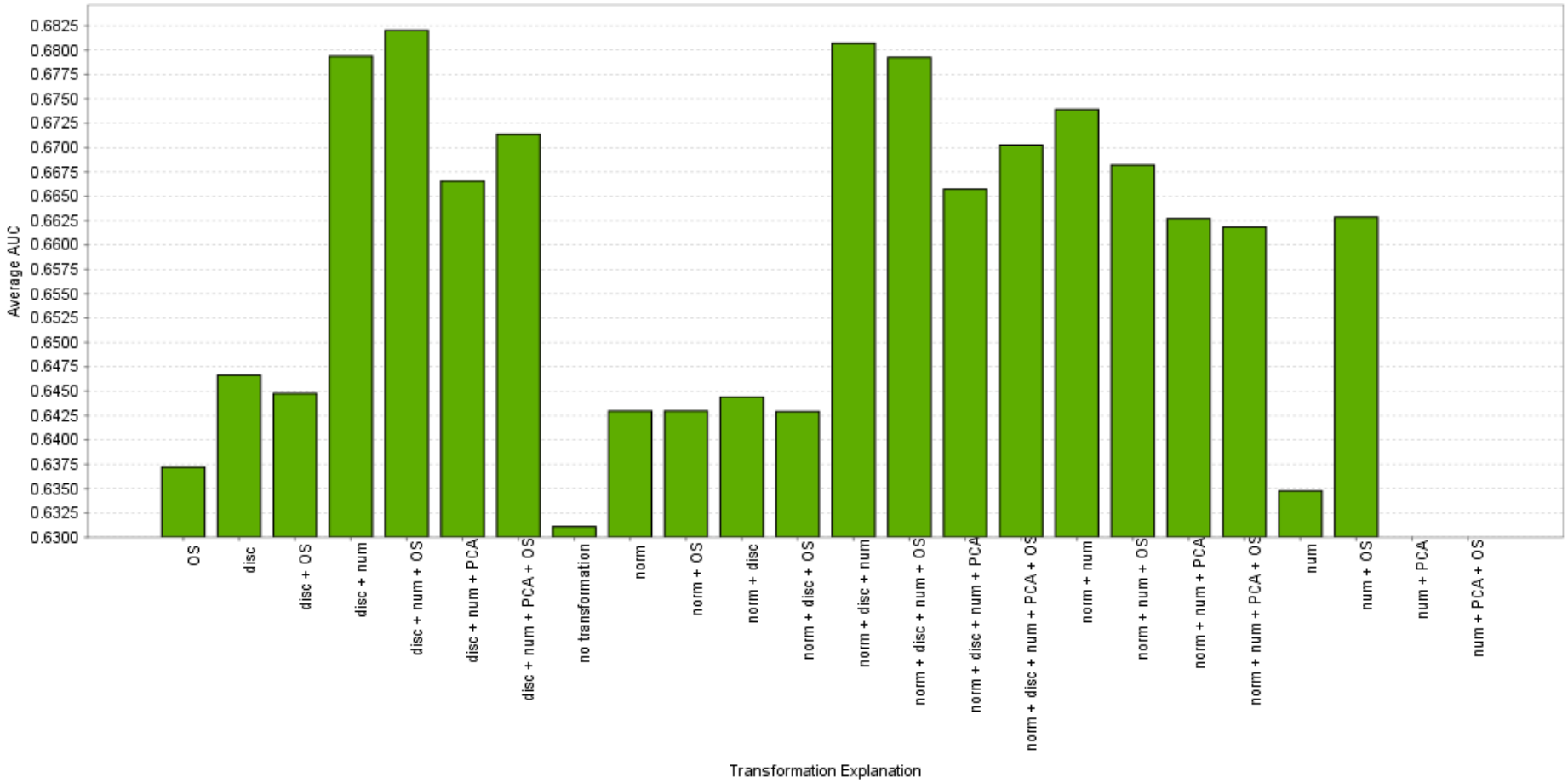

Figure 5.1 Average AUC Values Across Data Transformations on German data

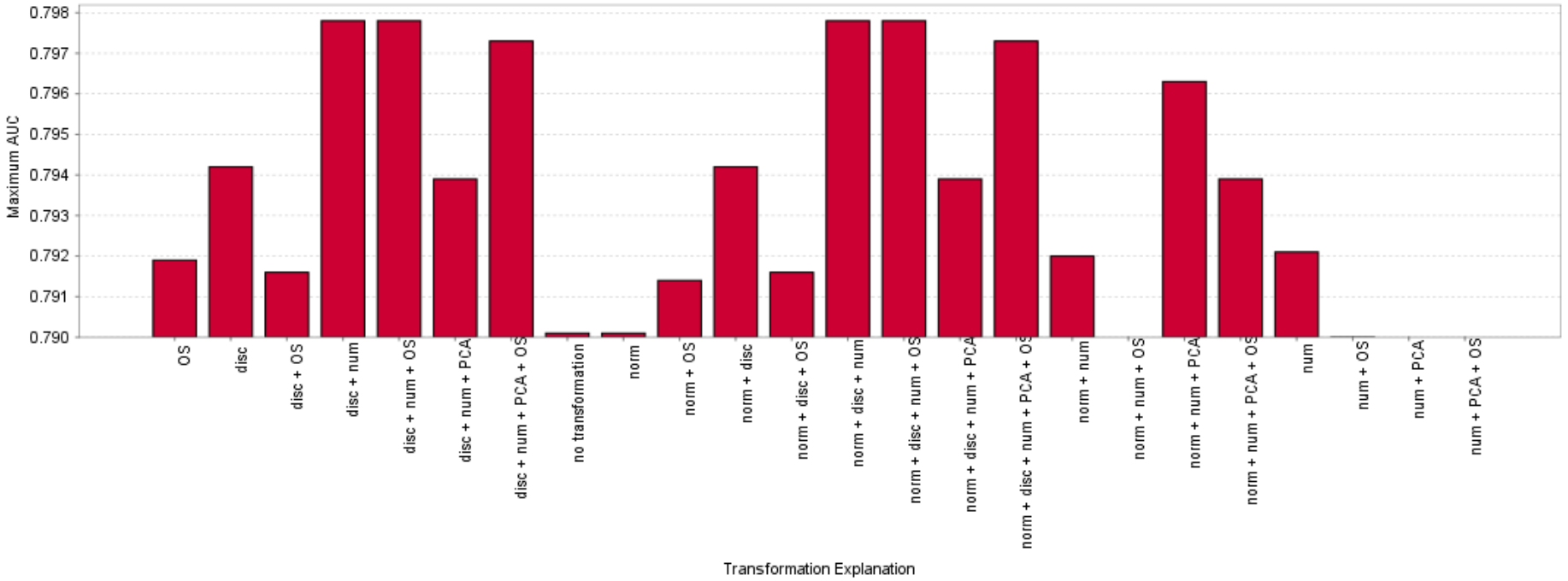

Figure 5.2 Maximum AUC Values Across Data Transformations on German data

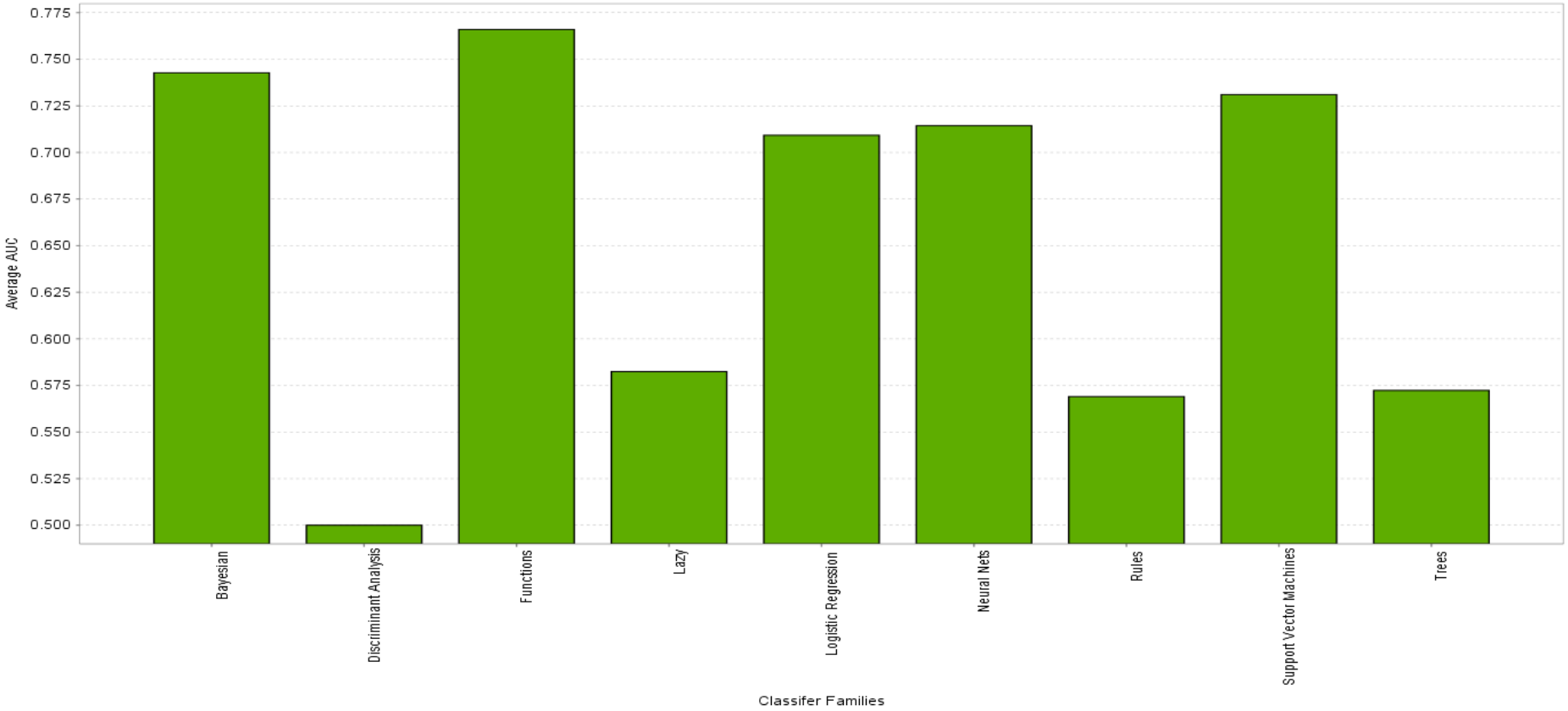

Figure 5.3 Average AUC Values of Classifier Families on German data

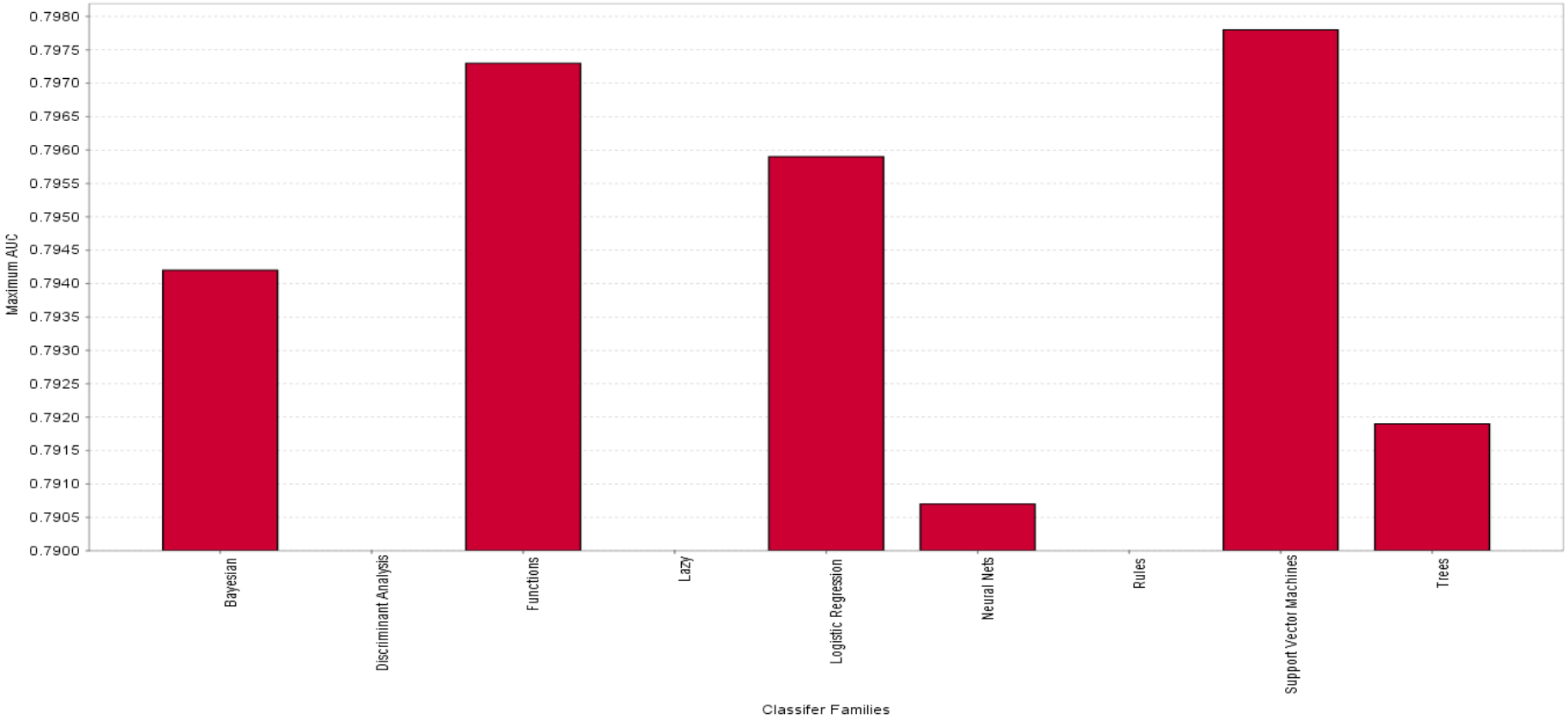

Figure 5.4 Maximum AUC Values of Classifier Families on German data

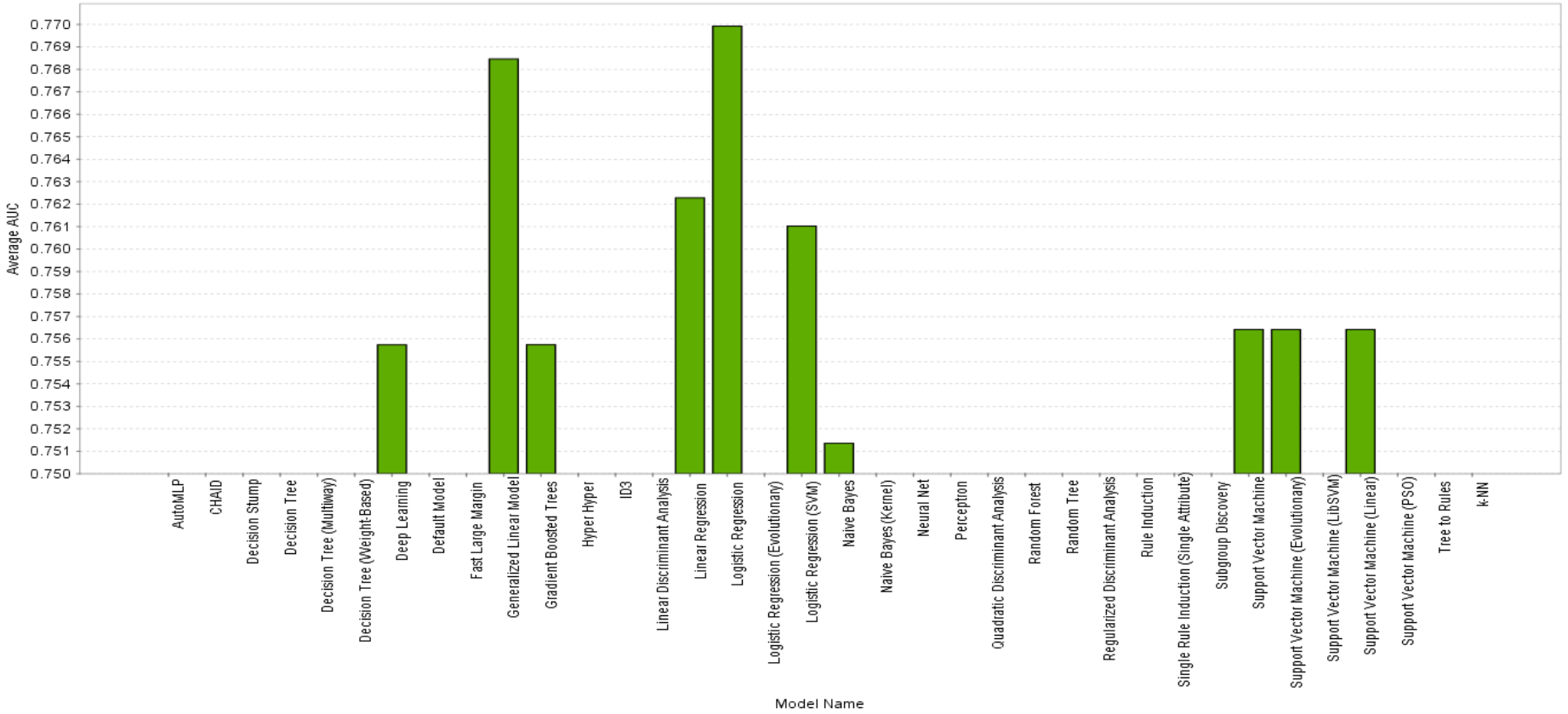

Figure 5.5 Average AUC Values of Models on German data (focused values above 0.75)

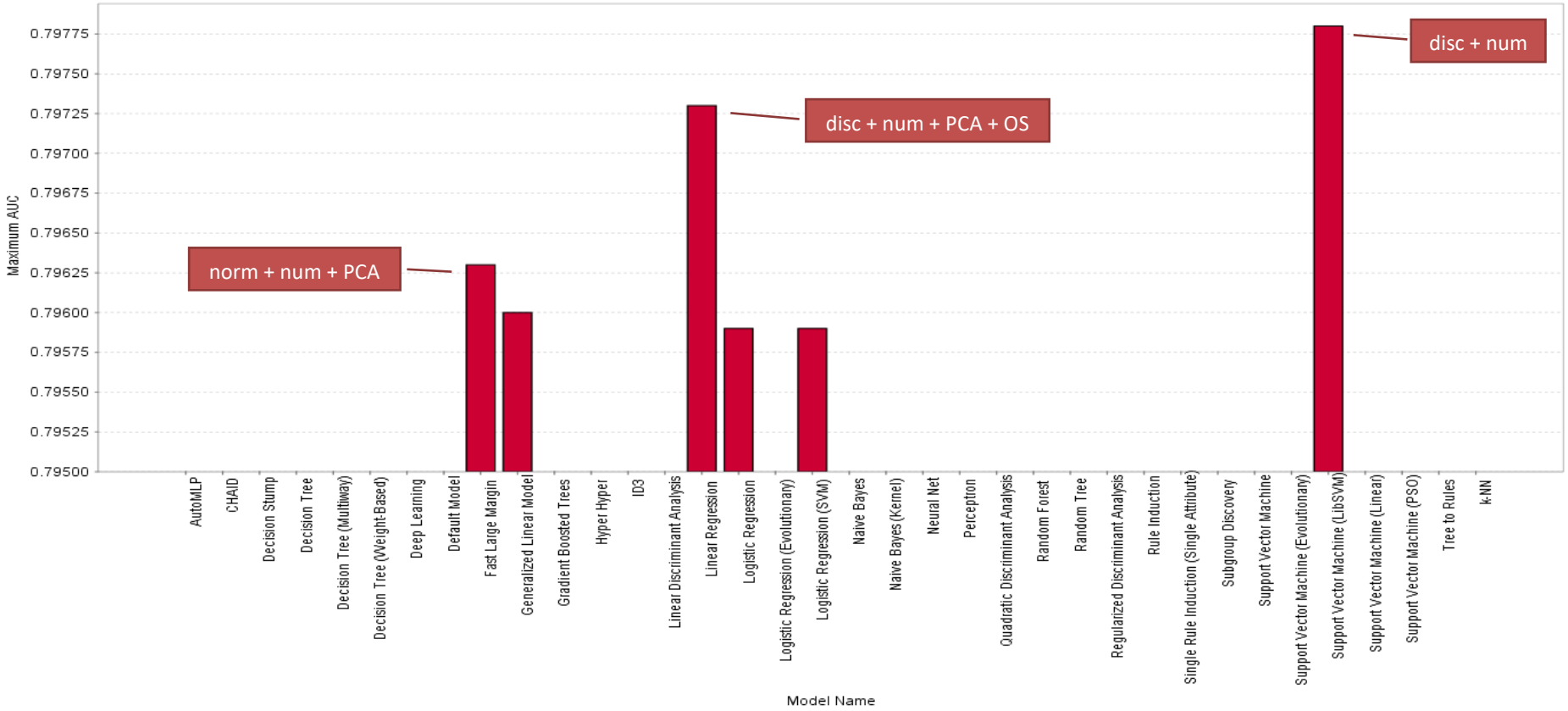

Figure 5.6 Maximum AUC Values of Models on German data (focused values above 0.795)

Support Vector Machine (LibSVM)

accuracy: 76.40%	true good	true bad	class precision
pred. good	544	80	87.18%
pred. bad	156	220	58.51%
class recall	77.71%	73.33%	

Linear Regression

accuracy: 74.00%	true good	true bad	class precision
pred. good	508	68	88.19%
pred. bad	192	232	54.72%
class recall	72.57%	77.33%	

Fast Large Margin

accuracy: 72.60%	true good	true bad	class precision
pred. good	490	64	88.45%
pred. bad	210	236	52.91%
class recall	70.00%	78.67%	

Figure 5.7 Confusion Matrices of the Three Most Successful Models on German Data

Table 5.2 First 20 Performance Results of Data-Model Pairings on Australian credit data set

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + OS	Trees	Gradient Boosted Trees	13	1338	0,9418	0,0341
norm	Trees	Gradient Boosted Trees	14	1325	0,9405	0,0341
no transformation	Trees	Gradient Boosted Trees	14	1364	0,9381	0,0314
norm + OS	Neural Nets	Deep Learning	13	4211	0,9341	0,0196
OS	Logistic Regression	Logistic Regression	13	212	0,9331	0,0241
norm + OS	Logistic Regression	Logistic Regression	13	221	0,9331	0,0241
norm + OS	Support Vector Machines	Fast Large Margin	13	175	0,9331	0,0253
OS	Functions	Generalized Linear Model	13	209	0,9329	0,0243
norm + OS	Functions	Generalized Linear Model	13	214	0,9329	0,0243
OS	Trees	Gradient Boosted Trees	13	1356	0,9329	0,0330
norm + OS	Logistic Regression	Logistic Regression (SVM)	13	325	0,9320	0,0250
norm + disc	Neural Nets	Deep Learning	11	4286	0,9320	0,0288
no transformation	Logistic Regression	Logistic Regression	14	223	0,9314	0,0238
norm	Logistic Regression	Logistic Regression	14	224	0,9314	0,0238
norm	Support Vector Machines	Fast Large Margin	14	201	0,9313	0,0262
norm	Functions	Generalized Linear Model	14	216	0,9309	0,0242
no transformation	Functions	Generalized Linear Model	14	307	0,9309	0,0242
OS	Logistic Regression	Logistic Regression (SVM)	13	375	0,9308	0,0262
norm + disc + OS	Neural Nets	Deep Learning	10	4290	0,9308	0,0296
disc + OS	Trees	Gradient Boosted Trees	10	1008	0,9307	0,0320

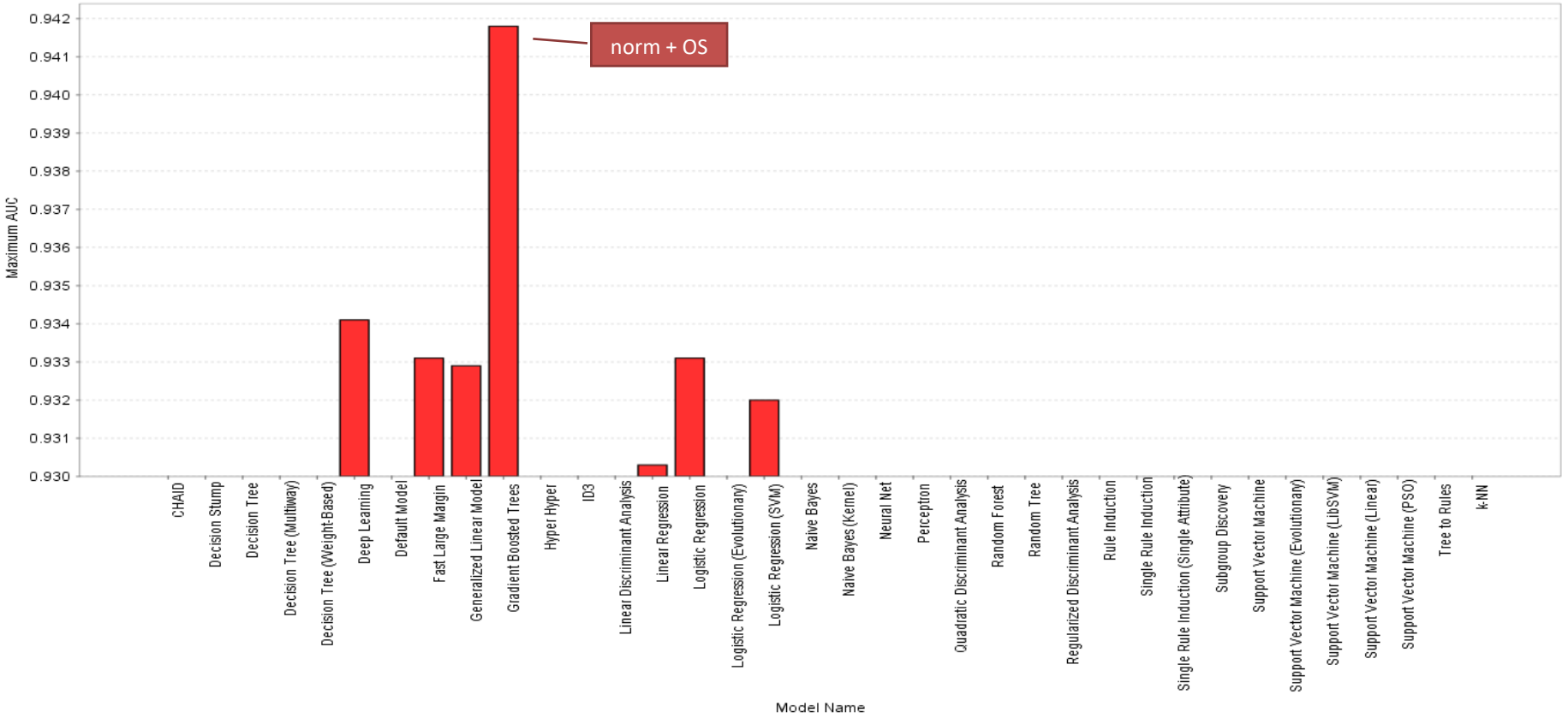

Figure 5.8 Maximum AUC Values of Models on Australian data (focused values above 0.93)

5.2 Finding the Best Ensemble Model

It was shown in recent studies that ensemble models achieve better results than single models in credit scoring. Because of this reason, in this study it is aimed to construct a fully automatic and dynamic system which includes ensemble modelling as well. After failing to find an objective criterion to select successful single models, 6 successful models are selected manually and fed to the ensemble model building engine. EMBE tried all 3648 combinations exhaustively in weighted voting.

For German credit data set, after deducting erroneous trials, the resultant comparison table includes 1032 trials. The first striking result is that ensemble models do not bring significant gain over single model. The first five successful trials are listed in Table 5.3. In the fifth row, there is only one combined model, which means nothing but the single model itself. As can be observed even with bare eye, there is no significant difference in AUC values of weighted voting ensemble models over the single model. Therefore, one should not always expect a gain from ensemble modelling. In such cases where the improvement of AUC values is insignificant keeping the single model will be preferable since it is more interpretable and simple to implement.

For Australian credit data set, the results are different and more promising. Again, after removing erroneous trials, the final comparison table includes 564 trials. The first five successful trials along with the most successful single model is presented in Table 5.4. The most important result here is that weighted voting ensemble model does bring improvement over single model. In the sixth row, the performance of the single gradient boosted trees[5] is presented which was the best predictor among single

[5] Although gradient boosted trees is counted among single models, it is basically boosting type homogeneous ensemble mode by itself which boosts decision tree algorithms by learning its

models. Even the fifth most successful weighted voting has significantly higher AUC than the single gradient boosted trees. From these results it can be confidently concluded that the exhaustive trials of EMBE generate significant improvements over single models and the engine is definitely working.

mistakes. In this study, when ensemble model is referred it means weighted voting which is a heterogeneous ensemble model that can combine different types of single models. Therefore, gradient boosted trees is handled as a single model.

Table 5.3 First 5 Performance Results of Weighted Model Combinations on German credit data set

Combined Model 1	Combined Model 2	Combined Model 3	Combined Model 4	AUC	AUC STDEV
Support Vector Machine (LibSVM)	Generalized Linear Model	Logistic Regression	Logistic Regression (SVM)	0.7996	0.0304
			Logistic Regression (SVM)	0.7995	0.0304
	Generalized Linear Model	Logistic Regression	Logistic Regression (SVM)	0.7995	0.0305
		Logistic Regression	Logistic Regression (SVM)	0.7994	0.0304
	Generalized Linear Model			0.7994	0.0302

Table 5.4 First 5 Performance Results of Weighted Model Combinations on Australian credit data set

Combined Model 1	Combined Model 2	Combined Model 3	Combined Model 4	Combined Model 5	Combined Model 6	AUC	AUC STDEV
Gradient Boosted Trees	Deep Learning			Logistic Regression (SVM)		0.9473	0.0207
Gradient Boosted Trees		Generalized Linear Model				0.9451	0.0236
Gradient Boosted Trees	Deep Learning		Logistic Regression			0.9451	0.0197
Gradient Boosted Trees					Fast Large Margin	0.9449	0.0234
Gradient Boosted Trees	Deep Learning	Generalized Linear Model				0.9448	0.0202
Gradient Boosted Trees						0.9392	0.0305

5.3 Final Words and Future Work

Throughout this study the main objective was constructing totally automatic and dynamic system in RapidMiner which can take any kind of credit data and perform necessary data transformations and train all built-in models of RapidMiner. On top of that, it is also aimed to build automatically weighted voting ensemble models which combine successful single models to achieve better performance results in credit scoring.

Since most of the related studies in this field do not give satisfactory explanations as to which data transformations are applied based on what, a comprehensive approach has been chosen and almost all kind of data transformations are performed and the resultant data sets are kept distinctly. After comparing and evaluating the effects of these transformations on model accuracies, it can be surely concluded that knowing the most optimal and required data transformation beforehand is rather difficult. Thus, trying almost every kind of transformation could give the researcher the ability not to miss the ideal data transformation steps.

In similar studies researchers compare performances of shortlisted candidate models in credit scoring. The underlying rationale of these shortlists is mostly ambiguous. In this study, each 43 built-in model in RapidMiner is considered as s worthwhile candidate for being the best predictor in credit scoring. Accordingly, a model comparison engine is created in RapidMiner environment which is capable of training every model on every transformed data set. This engine proved itself to be beneficial. Because, considered the myriad data transformations, finding the most successful model is a difficult task. Thus, the comprehensive approach of the engine will help researchers reach the most successful model without the fear of omitting likely successful candidates.

As the recent comprehensive studies suggest the best predictors in credit scoring are ensemble models, particularly weighted voting. In this study, it is tried to find an objective selection criterion which will enable the ensemble model building engine to select successful single models and combine them automatically. By this way, the entire system could work without any human intervention. Unfortunately, the consequence was a failure. Thus, a human interaction is required to connect the two engines. After selecting and feeding every combination of these selected models into weighted voting, ensemble model building engine tried every possible pairing and achieved better performance results. Thus, after using model comparison engine, researchers or practitioners could manually select successful models from different model families and find the best combination of ensemble model. Hence, it will not be incorrect to conclude that ensemble model building engine will also help researchers substantially.

If the manual gap between the model comparison engine and the ensemble model building engine could be filled with an objective method, then it would be possible for a credit analyst to feed the credit data into RapidMiner and find out which ensemble model has the best predictive capability on what kind of data transformation. In this study, selection is conducted based on correlation between single models and the outcome was a failure. Future work can be concentrated on this bridge between the two engines. That way, totally automatic and dynamic system working in RapidMiner could be constructed and credit decisions could be made without any human interaction at the best possible accuracy.

REFERENCES

Abellán, J. & Mantas, C.J., 2014. Improving experimental studies about ensembles of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, 41(8), pp.3825–3830. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0957417413009676.

Akkoç, S., 2012. An empirical comparison of conventional techniques, neural networks and the three stage hybrid Adaptive Neuro Fuzzy Inference System (ANFIS) model for credit scoring analysis: The case of Turkish credit card data. *European Journal of Operational Research*, 222(1), pp.168–178.

Baesens, B. et al., 2009. 50 Years of Data Mining and OR: Upcoming Trends and Challenges. *The Journal of the Operational Research Society*, 60, pp.s16–s23. Available at: http://www.jstor.org/stable/40206722.

Brooks, C., 2008. *Introductory Econometrics for Finance* 2nd ed., Cambridge: Cambridge University Press.

Brown, I. & Mues, C., 2012. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3), pp.3446–3453. Available at: http://dx.doi.org/10.1016/j.eswa.2011.09.033.

Brownlee, J., 2016. Supervised and Unsupervised Machine Learning Algorithms - Machine Learning Mastery. Available at: http://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ [Accessed December 16, 2016].

Finlay, S., 2011. Multiple classifier architectures and their application to credit risk

assessment. *European Journal of Operational Research*, 210(2), pp.368–378. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0377221710006272 [Accessed December 12, 2016].

Fortmann-Roe, S., 2012. Understanding the Bias-Variance Tradeoff. Available at: http://scott.fortmann-roe.com/docs/BiasVariance.html [Accessed January 1, 2017].

Giudici, P. & Figini, S., 2009. *Applied Data Mining for Business and Industry* 2nd ed., Chichester, UK: John Wiley & Sons, Ltd. Available at: http://doi.wiley.com/10.1002/9780470745830.

Han, L., Han, L. & Zhao, H., 2013. Orthogonal support vector machine for credit scoring. *Engineering Applications of Artificial Intelligence*, 26(2), pp.848–862. Available at: http://dx.doi.org/10.1016/j.engappai.2012.10.005.

Harris, T., 2015. Credit scoring using the clustered support vector machine. *Expert Systems with Applications*, 42(2), pp.741–750. Available at: http://dx.doi.org/10.1016/j.eswa.2014.08.029.

Hens, A.B. & Tiwari, M.K., 2012. Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method. *Expert Systems with Applications*, 39(8), pp.6774–6781. Available at: http://dx.doi.org/10.1016/j.eswa.2011.12.057.

Hofmann, H., UCI Machine Learning Repository: Statlog (German Credit Data) Data Set. Available at: https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data) [Accessed December 15, 2016].

Huang, S.-C. & Day, M.-Y., 2013. A comparative study of data mining techniques for credit scoring in banking. In *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI)*. IEEE, pp. 684–691. Available at: http://ieeexplore.ieee.org/document/6642534/.

Imanuel, 2016. What is Predictive Modeling ? *Predictive Analytics Today*. Available at: http://www.predictiveanalyticstoday.com/predictive-modeling/ [Accessed December 17, 2016].

Kantardzic, M., 2011. *DATA MINING Concepts, Models, Methods, and Algorithms* 2nd ed., Hoboken, New Jersey: IEEE PRESS A JOHN WILEY & SONS, INC.

Khudnitskaya, A.S., 2010. *Improved Credit Scoring with Multilevel Statistical Modelling*. Technische Universität Dortmund.

Kraus, A., 2014. *Recent Methods from Statistics and Machine Learning for Credit Scoring*. Ludwig-Maximilians-Universität. Available at: http://edoc.ub.uni-muenchen.de/17143/1/Kraus_Anne.pdf.

Kruppa, J. et al., 2013. Consumer credit risk: Individual probability estimates using machine learning. *Expert Systems with Applications*, 40(13), pp.5125–5131. Available at: http://dx.doi.org/10.1016/j.eswa.2013.03.019.

Lessmann, S. et al., 2015. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1), pp.124–136. Available at: http://dx.doi.org/10.1016/j.ejor.2015.05.030.

Li, S., Tsang, I.W. & Chaudhari, N.S., 2012. Relevance vector machine based

infinite decision agent ensemble learning for credit risk analysis. *Expert Systems with Applications*, 39(5), pp.4947–4953. Available at: http://dx.doi.org/10.1016/j.eswa.2011.10.022.

Marqués, A.I., García, V. & Sánchez, J.S., 2012a. Exploring the behaviour of base classifiers in credit scoring ensembles. *Expert Systems with Applications*, 39(11), pp.10244–10250.

Marqués, A.I., García, V. & Sánchez, J.S., 2012b. Two-level classifier ensembles for credit risk assessment. *Expert Systems with Applications*, 39(12), pp.10916–10922.

Mctiernan, K., 2016. *Investigation into the Predictive Capability of Macro-Economic Features in Modelling Credit Risk for Small Medium Enterprises*. Dublin Institute of Technology.

Mitchell-Guthrie, P., 1998. Looking backwards, looking forwards: SAS, data mining, and machine learning - Subconscious Musings. *SAS Institute Inc.* Available at: http://blogs.sas.com/content/subconsciousmusings/2014/08/22/looking-backwards-looking-forwards-sas-data-mining-and-machine-learning/ [Accessed December 16, 2016].

Mues, C. et al., 2004. Decision diagrams in machine learning: an empirical study on real-life credit-risk data. *Expert Systems with Applications*, 27(2), pp.257–264. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0957417404000120.

Refaeilzadeh, P., Tang, L. & Liu, H., 2009. Cross-Validation L. LIU & M. T. ÖZSU, eds. *Encyclopedia of Database Systems*, pp.532–538. Available at: http://dx.doi.org/10.1007/978-0-387-39940-9_565.

Sayad, S., 2016. An Introduction to Data Mining Map. Available at: http://www.saedsayad.com/data_mining_map.htm [Accessed December 16, 2016].

Seni, G. & Elder, J.F., 2010. *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions* R. Grossman, ed., Morgan & Claypool.

Tsai, C.-F., 2009. Feature selection in bankruptcy prediction. *Knowledge-Based Systems*, 22(2), pp.120–127. Available at: http://www.sciencedirect.com/science/article/pii/S0950705108001536.

Tsai, C.-F., Hsu, Y.-F. & Yen, D.C., 2014. A comparative study of classifier ensembles for bankruptcy prediction. *Applied Soft Computing*, 24, pp.977–984. Available at: http://dx.doi.org/10.1016/j.asoc.2014.08.047.

Tsai, C.F., 2014. Combining cluster analysis with classifier ensembles to predict financial distress. *Information Fusion*, 16(1), pp.46–58. Available at: http://dx.doi.org/10.1016/j.inffus.2011.12.001.

Tufféry, S., 2011. *Data Mining and Statistics for Decision Making*, Chichester, UK: John Wiley & Sons, Ltd.

Twala, B., 2010. Multiple classifier application to credit risk assessment. *Expert Systems with Applications*, 37(4), pp.3326–3336. Available at: http://dx.doi.org/10.1016/j.eswa.2009.10.018.

Wang, G. et al., 2011. A comparative assessment of ensemble learning for credit scoring. *Expert Systems with Applications*, 38(1), pp.223–230. Available at:

http://dx.doi.org/10.1016/j.eswa.2010.06.048.

Wang, G. et al., 2012. Two credit scoring models based on dual strategy ensemble trees. *Knowledge-Based Systems*, 26, pp.61–68. Available at: http://dx.doi.org/10.1016/j.knosys.2011.06.020.

Yazdani, Z., Sepehri, M.M. & Teimourpour, B., 2014. Selecting Best Features for Predicting Bank Loan Default. In *Data Mining Applications with R*. Elsevier, pp. 229–245. Available at: http://linkinghub.elsevier.com/retrieve/pii/B9780124115118000086.

Yottamine Analytics, L., 2013. The Machine Learning Advantage. *Yottamine Analytics, LLC*. Available at: https://yottamine.com/machine-learning-svm [Accessed January 1, 2017].

APPENDICES

Appendix A: Description of the German credit dataset

```
Description of the German credit dataset.

1. Title: German Credit data

2. Source Information

Professor Dr. Hans Hofmann
Institut f"ur Statistik und "Okonometrie
Universit"at Hamburg
FB Wirtschaftswissenschaften
Von-Melle-Park 5
2000 Hamburg 13

3. Number of Instances:  1000

Two datasets are provided.  the original dataset, in the form
provided
by Prof. Hofmann, contains categorical/symbolic attributes and
is in the file "german.data".

For algorithms that need numerical attributes, Strathclyde
University
produced the file "german.data-numeric".  This file has been
edited
and several indicator variables added to make it suitable for
algorithms which cannot cope with categorical variables.
Several
attributes that are ordered categorical (such as attribute 17)
have
been coded as integer.    This was the form used by StatLog.


6. Number of Attributes german: 20 (7 numerical, 13 categorical)
   Number of Attributes german.numer: 24 (24 numerical)


7.  Attribute description for german

Attribute 1:  (qualitative)
             Status of existing checking account
                A11 :      ... <    0 DM
             A12 : 0 <= ... <  200 DM
             A13 :      ... >= 200 DM /
                 salary assignments for at least 1 year
               A14 : no checking account

Attribute 2:  (numerical)
            Duration in month

Attribute 3:  (qualitative)
            Credit history
            A30 : no credits taken/
                all credits paid back duly
              A31 : all credits at this bank paid back duly
```

```
            A32 : existing credits paid back duly till now
              A33 : delay in paying off in the past
            A34 : critical account/
                other credits existing (not at this bank)

Attribute 4:  (qualitative)
            Purpose
            A40 : car (new)
            A41 : car (used)
            A42 : furniture/equipment
            A43 : radio/television
            A44 : domestic appliances
            A45 : repairs
            A46 : education
            A47 : (vacation - does not exist?)
            A48 : retraining
            A49 : business
            A410 : others

Attribute 5:  (numerical)
            Credit amount

Attibute 6:  (qualitative)
            Savings account/bonds
            A61 :          ... <  100 DM
            A62 :   100 <= ... <  500 DM
            A63 :   500 <= ... < 1000 DM
            A64 :          .. >= 1000 DM
              A65 :   unknown/ no savings account

Attribute 7:  (qualitative)
            Present employment since
            A71 : unemployed
            A72 :       ... < 1 year
            A73 : 1  <= ... < 4 years
            A74 : 4  <= ... < 7 years
            A75 :       .. >= 7 years

Attribute 8:  (numerical)
            Installment rate in percentage of disposable income

Attribute 9:  (qualitative)
            Personal status and sex
            A91 : male   : divorced/separated
            A92 : female : divorced/separated/married
              A93 : male   : single
            A94 : male   : married/widowed
            A95 : female : single

Attribute 10: (qualitative)
            Other debtors / guarantors
            A101 : none
            A102 : co-applicant
            A103 : guarantor

Attribute 11: (numerical)
            Present residence since
```

Attribute 12: (qualitative)
Property
A121 : real estate
A122 : if not A121 : building society savings agreement/
life insurance
A123 : if not A121/A122 : car or other, not in attribute 6
A124 : unknown / no property

Attribute 13: (numerical)
Age in years

Attribute 14: (qualitative)
Other installment plans
A141 : bank
A142 : stores
A143 : none

Attribute 15: (qualitative)
Housing
A151 : rent
A152 : own
A153 : for free

Attribute 16: (numerical)
Number of existing credits at this bank

Attribute 17: (qualitative)
Job
A171 : unemployed/ unskilled - non-resident
A172 : unskilled - resident
A173 : skilled employee / official
A174 : management/ self-employed/
highly qualified employee/ officer

Attribute 18: (numerical)
Number of people being liable to provide maintenance for

Attribute 19: (qualitative)
Telephone
A191 : none
A192 : yes, registered under the customers name

Attribute 20: (qualitative)
foreign worker
A201 : yes
A202 : no

8. Cost Matrix

This dataset requires use of a cost matrix (see below)

	1	2

1	0	1
2	5	0

(1 = Good, 2 = Bad)

the rows represent the actual classification and the columns the predicted classification.

It is worse to class a customer as good when they are bad (5), than it is to class a customer as bad when they are good (1).

Appendix B: Description of the Australian credit dataset

```
     Description of the Dataset:

THIS CREDIT DATA ORIGINATES FROM QUINLAN (see below).

1. Title: Australian Credit Approval

2. Sources:
    (confidential)
    Submitted by quinlan@cs.su.oz.au

3.  Past Usage:

    See Quinlan,
    * "Simplifying decision trees", Int J Man-Machine Studies 27,
      Dec 1987, pp. 221-234.
    * "C4.5: Programs for Machine Learning", Morgan Kaufmann, Oct
1992

4.  Relevant Information:

    This file concerns credit card applications.  All attribute
names
    and values have been changed to meaningless symbols to
protect
    confidentiality of the data.

    This dataset is interesting because there is a good mix of
    attributes -- continuous, nominal with small numbers of
    values, and nominal with larger numbers of values.  There
    are also a few missing values.

5.  Number of Instances: 690

6.  Number of Attributes: 14 + class attribute

7.  Attribute Information:   THERE ARE 6 NUMERICAL AND 8
CATEGORICAL ATTRIBUTES.

                             THE LABELS HAVE BEEN CHANGED FOR THE
CONVENIENCE
                             OF THE STATISTICAL ALGORITHMS.   FOR
EXAMPLE,
                             ATTRIBUTE 4 ORIGINALLY HAD 3 LABELS
p,g,gg AND
                             THESE HAVE BEEN CHANGED TO LABELS
1,2,3.


    A1:    0,1    CATEGORICAL
        a,b
    A2:    continuous.
    A3:    continuous.
    A4:    1,2,3         CATEGORICAL
        p,g,gg
    A5:  1, 2,3,4,5, 6,7,8,9,10,11,12,13,14    CATEGORICAL
         ff,d,i,k,j,aa,m,c,w, e, q, r,cc, x
```

```
A6:     1, 2,3, 4,5,6,7,8,9    CATEGORICAL
    ff,dd,j,bb,v,n,o,h,z

A7:    continuous.
A8:    1, 0       CATEGORICAL
    t, f.
A9: 1, 0         CATEGORICAL
    t, f.
A10:   continuous.
A11:  1, 0       CATEGORICAL
      t, f.
A12:    1, 2, 3    CATEGORICAL
        s, g, p
A13:   continuous.
A14:   continuous.
A15:   1,2
       +,-         (class attribute)
```

8. Missing Attribute Values:
   37 cases (5%) HAD one or more missing values. The missing values from particular attributes WERE:

```
A1:  12
A2:  12
A4:   6
A5:   6
A6:   9
A7:   9
A14: 13
```

   THESE WERE REPLACED BY THE MODE OF THE ATTRIBUTE (CATEGORICAL)
                                MEAN OF THE ATTRIBUTE (CONTINUOUS)

9. Class Distribution

```
+: 307 (44.5%)    CLASS 2
-: 383 (55.5%)    CLASS 1
```

10. There is no cost matrix.

Appendix C: Performance Results of Data-Model Pairings on German credit data set

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc + num + OS	Support Vector Machines	Support Vector Machine (LibSVM)	57	941	0,7978	0,0294
disc + num + OS	Support Vector Machines	Support Vector Machine (LibSVM)	57	951	0,7978	0,0294
norm + disc + num	Support Vector Machines	Support Vector Machine (LibSVM)	58	954	0,7978	0,0297
disc + num	Support Vector Machines	Support Vector Machine (LibSVM)	58	957	0,7978	0,0297
norm + disc + num + PCA + OS	Functions	Linear Regression	28	1391	0,7973	0,0291
disc + num + PCA + OS	Functions	Linear Regression	28	1451	0,7973	0,0291
norm + num + PCA	Support Vector Machines	Fast Large Margin	32	550	0,7963	0,0264
disc + num + PCA + OS	Functions	Generalized Linear Model	28	264	0,7960	0,0263
norm + disc + num + PCA + OS	Functions	Generalized Linear Model	28	270	0,7960	0,0263
norm + disc + num + PCA + OS	Logistic Regression	Logistic Regression	28	270	0,7959	0,0262
disc + num + PCA + OS	Logistic Regression	Logistic Regression	28	272	0,7959	0,0262
norm + disc + num + PCA + OS	Logistic Regression	Logistic Regression (SVM)	28	1081	0,7959	0,0262
disc + num + PCA + OS	Logistic Regression	Logistic Regression (SVM)	28	1083	0,7959	0,0262
norm + disc + num + PCA + OS	Support Vector Machines	Fast Large Margin	28	179	0,7958	0,0284
disc + num + PCA + OS	Support Vector Machines	Fast Large Margin	28	181	0,7958	0,0284
norm + num + PCA	Logistic Regression	Logistic Regression	32	290	0,7957	0,0259
norm + num + PCA	Functions	Generalized Linear Model	32	276	0,7957	0,0260
norm + num + PCA	Logistic Regression	Logistic Regression (SVM)	32	1219	0,7955	0,0261
norm + num + PCA	Functions	Linear Regression	32	1883	0,7944	0,0303
norm + disc	Bayesian	Naive Bayes	15	36	0,7942	0,0220
disc	Bayesian	Naive Bayes	15	38	0,7942	0,0220
norm + disc	Bayesian	Naive Bayes (Kernel)	15	39	0,7942	0,0220
disc	Bayesian	Naive Bayes (Kernel)	15	40	0,7942	0,0220
norm + disc + num + PCA	Functions	Linear Regression	32	1876	0,7939	0,0293

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
disc + num + PCA	Functions	Linear Regression	32	1894	0,7939	0,0293
norm + num + PCA + OS	Logistic Regression	Logistic Regression	30	284	0,7939	0,0312
norm + disc + num	Support Vector Machines	Support Vector Machine	58	2469	0,7939	0,0361
norm + disc + num	Support Vector Machines	Support Vector Machine (Linear)	58	2469	0,7939	0,0361
norm + disc + num	Support Vector Machines	Support Vector Machine (Evolutionary)	58	2483	0,7939	0,0361
disc + num	Support Vector Machines	Support Vector Machine (Evolutionary)	58	2488	0,7939	0,0361
disc + num	Support Vector Machines	Support Vector Machine	58	2523	0,7939	0,0361
disc + num	Support Vector Machines	Support Vector Machine (Linear)	58	2584	0,7939	0,0361
norm + num + PCA + OS	Functions	Generalized Linear Model	30	277	0,7938	0,0312
norm + num + PCA + OS	Logistic Regression	Logistic Regression (SVM)	30	1152	0,7937	0,0314
norm + num + PCA + OS	Functions	Linear Regression	30	1627	0,7933	0,0340
norm + disc + num + PCA + OS	Support Vector Machines	Support Vector Machine (Linear)	28	2270	0,7932	0,0283
norm + disc + num + PCA + OS	Support Vector Machines	Support Vector Machine (Evolutionary)	28	2282	0,7932	0,0283
disc + num + PCA + OS	Support Vector Machines	Support Vector Machine	28	2290	0,7932	0,0283
norm + disc + num + PCA + OS	Support Vector Machines	Support Vector Machine	28	2308	0,7932	0,0283
disc + num + PCA + OS	Support Vector Machines	Support Vector Machine (Linear)	28	2367	0,7932	0,0283
disc + num + PCA + OS	Support Vector Machines	Support Vector Machine (Evolutionary)	28	2908	0,7932	0,0283
norm + num + PCA + OS	Support Vector Machines	Fast Large Margin	30	490	0,7931	0,0324
norm + disc + num	Functions	Linear Regression	58	7776	0,7930	0,0311
disc + num	Functions	Linear Regression	58	7851	0,7930	0,0311
norm + disc + num + OS	Support Vector Machines	Support Vector Machine (Linear)	57	2321	0,7926	0,0360
norm + disc + num + OS	Support Vector Machines	Support Vector Machine (Evolutionary)	57	2340	0,7926	0,0360
norm + disc + num + OS	Support Vector Machines	Support Vector Machine	57	2354	0,7926	0,0360
disc + num + OS	Support Vector Machines	Support Vector Machine (Linear)	57	2368	0,7926	0,0360
disc + num + OS	Support Vector Machines	Support Vector Machine	57	2376	0,7926	0,0360
disc + num + OS	Support Vector Machines	Support Vector Machine (Evolutionary)	57	2378	0,7926	0,0360

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc + num + OS	Functions	Generalized Linear Model	57	350	0,7924	0,0231
disc + num + OS	Functions	Generalized Linear Model	57	354	0,7924	0,0231
norm + disc + num	Functions	Generalized Linear Model	58	352	0,7923	0,0231
disc + num	Functions	Generalized Linear Model	58	366	0,7923	0,0231
norm + disc + num + PCA	Logistic Regression	Logistic Regression (SVM)	32	1223	0,7921	0,0278
disc + num + PCA	Logistic Regression	Logistic Regression (SVM)	32	1258	0,7921	0,0278
norm + disc + num + PCA	Functions	Generalized Linear Model	32	275	0,7921	0,0280
disc + num + PCA	Functions	Generalized Linear Model	32	286	0,7921	0,0280
num	Support Vector Machines	Support Vector Machine (Evolutionary)	61	3238	0,7921	0,0315
num	Support Vector Machines	Support Vector Machine (Linear)	61	3269	0,7921	0,0315
num	Support Vector Machines	Support Vector Machine	61	3350	0,7921	0,0315
norm + disc + num + PCA	Logistic Regression	Logistic Regression	32	286	0,7920	0,0277
disc + num + PCA	Logistic Regression	Logistic Regression	32	291	0,7920	0,0277
disc + num + PCA	Support Vector Machines	Fast Large Margin	32	213	0,7920	0,0294
norm + disc + num + PCA	Support Vector Machines	Fast Large Margin	32	215	0,7920	0,0294
norm + num	Support Vector Machines	Fast Large Margin	61	1018	0,7920	0,0300
OS	Trees	Gradient Boosted Trees	17	1737	0,7919	0,0376
disc + OS	Bayesian	Naive Bayes	14	36	0,7916	0,0246
norm + disc + OS	Bayesian	Naive Bayes	14	36	0,7916	0,0246
disc + OS	Bayesian	Naive Bayes (Kernel)	14	37	0,7916	0,0246
norm + disc + OS	Bayesian	Naive Bayes (Kernel)	14	37	0,7916	0,0246
norm + num	Functions	Linear Regression	61	9048	0,7916	0,0308
disc + num + PCA + OS	Support Vector Machines	Support Vector Machine (LibSVM)	28	680	0,7915	0,0273
norm + disc + num + PCA + OS	Support Vector Machines	Support Vector Machine (LibSVM)	28	681	0,7915	0,0273
num	Functions	Linear Regression	61	9836	0,7915	0,0289
norm + OS	Trees	Gradient Boosted Trees	17	1820	0,7914	0,0340

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + num	Functions	Generalized Linear Model	61	374	0,7913	0,0223
num	Functions	Generalized Linear Model	61	414	0,7913	0,0223
norm + disc + num + OS	Functions	Linear Regression	57	7495	0,7913	0,0314
disc + num + OS	Functions	Linear Regression	57	7582	0,7913	0,0314
norm + disc + num	Bayesian	Naive Bayes (Kernel)	58	86	0,7911	0,0190
disc + num	Bayesian	Naive Bayes (Kernel)	58	89	0,7911	0,0190
norm + disc + num + OS	Bayesian	Naive Bayes (Kernel)	57	88	0,7911	0,0192
disc + num + OS	Bayesian	Naive Bayes (Kernel)	57	95	0,7911	0,0192
norm + num	Support Vector Machines	Support Vector Machine (Linear)	61	4293	0,7909	0,0322
norm + num	Support Vector Machines	Support Vector Machine	61	4329	0,7909	0,0322
norm + num	Support Vector Machines	Support Vector Machine (Evolutionary)	61	4335	0,7909	0,0322
norm + disc + num + OS	Logistic Regression	Logistic Regression (SVM)	57	1392	0,7908	0,0280
disc + num + OS	Logistic Regression	Logistic Regression (SVM)	57	1416	0,7908	0,0280
norm + disc + num + PCA	Support Vector Machines	Support Vector Machine (LibSVM)	32	724	0,7907	0,0299
disc + num + PCA	Support Vector Machines	Support Vector Machine (LibSVM)	32	744	0,7907	0,0299
norm + num	Neural Nets	Deep Learning	61	8486	0,7907	0,0378
norm + disc + num	Logistic Regression	Logistic Regression (SVM)	58	1396	0,7904	0,0287
disc + num	Logistic Regression	Logistic Regression (SVM)	58	1402	0,7904	0,0287
norm + OS	Logistic Regression	Logistic Regression	17	251	0,7902	0,0281
OS	Logistic Regression	Logistic Regression	17	256	0,7902	0,0281
norm	Logistic Regression	Logistic Regression	20	287	0,7901	0,0297
no transformation	Logistic Regression	Logistic Regression	20	359	0,7901	0,0297
norm + num	Logistic Regression	Logistic Regression	61	379	0,7901	0,0297
num	Logistic Regression	Logistic Regression	61	409	0,7901	0,0297
num + OS	Support Vector Machines	Support Vector Machine (Linear)	59	3658	0,7900	0,0282
num + OS	Support Vector Machines	Support Vector Machine (Evolutionary)	59	3677	0,7900	0,0282

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
num + OS	Support Vector Machines	Support Vector Machine	59	3724	0,7900	0,0282
norm + disc + num + OS	Support Vector Machines	Fast Large Margin	57	465	0,7900	0,0346
disc + num + OS	Support Vector Machines	Fast Large Margin	57	483	0,7900	0,0346
num + OS	Functions	Generalized Linear Model	59	337	0,7899	0,0225
norm + num + OS	Functions	Generalized Linear Model	59	348	0,7899	0,0225
norm + disc + num	Support Vector Machines	Fast Large Margin	58	486	0,7899	0,0346
disc + num	Support Vector Machines	Fast Large Margin	58	487	0,7899	0,0346
norm + num + OS	Support Vector Machines	Support Vector Machine	59	5481	0,7898	0,0284
norm + num + OS	Support Vector Machines	Support Vector Machine (Linear)	59	5497	0,7898	0,0284
norm + num + OS	Support Vector Machines	Support Vector Machine (Evolutionary)	59	5512	0,7898	0,0284
norm + num	Support Vector Machines	Support Vector Machine (LibSVM)	61	1006	0,7888	0,0210
norm + OS	Bayesian	Naive Bayes	17	38	0,7887	0,0142
OS	Bayesian	Naive Bayes	17	45	0,7887	0,0142
norm + disc	Neural Nets	Deep Learning	15	5294	0,7883	0,0276
norm + OS	Neural Nets	Deep Learning	17	5572	0,7883	0,0320
norm + num	Logistic Regression	Logistic Regression (SVM)	61	1554	0,7882	0,0311
num	Logistic Regression	Logistic Regression (SVM)	61	1738	0,7882	0,0311
no transformation	Trees	Gradient Boosted Trees	20	1919	0,7882	0,0334
norm + disc	Logistic Regression	Logistic Regression	15	251	0,7881	0,0334
disc	Logistic Regression	Logistic Regression	15	261	0,7881	0,0334
norm + disc + num	Logistic Regression	Logistic Regression	58	330	0,7881	0,0334
norm + disc + num + OS	Logistic Regression	Logistic Regression	57	341	0,7881	0,0334
disc + num	Logistic Regression	Logistic Regression	58	354	0,7881	0,0334
disc + num + OS	Logistic Regression	Logistic Regression	57	357	0,7881	0,0334
norm	Bayesian	Naive Bayes	20	39	0,7859	0,0087
no transformation	Bayesian	Naive Bayes	20	54	0,7859	0,0087

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + num + PCA	Support Vector Machines	Support Vector Machine (LibSVM)	32	736	0,7854	0,0245
num + OS	Functions	Linear Regression	59	8324	0,7850	0,0279
norm	Trees	Gradient Boosted Trees	20	1854	0,7850	0,0302
norm + num + PCA	Support Vector Machines	Support Vector Machine (Linear)	32	2111	0,7850	0,0302
norm + num + PCA	Support Vector Machines	Support Vector Machine (Evolutionary)	32	2118	0,7850	0,0302
norm + num + PCA	Support Vector Machines	Support Vector Machine	32	2122	0,7850	0,0302
disc + OS	Logistic Regression	Logistic Regression	14	245	0,7849	0,0344
norm + disc + OS	Logistic Regression	Logistic Regression	14	247	0,7849	0,0344
norm + num + OS	Support Vector Machines	Support Vector Machine (LibSVM)	59	975	0,7841	0,0251
norm + num + PCA + OS	Support Vector Machines	Support Vector Machine (Evolutionary)	30	1536	0,7841	0,0340
norm + num + PCA + OS	Support Vector Machines	Support Vector Machine (Linear)	30	1544	0,7841	0,0340
norm + num + PCA + OS	Support Vector Machines	Support Vector Machine	30	1568	0,7841	0,0340
norm + num + OS	Functions	Linear Regression	59	8213	0,7839	0,0283
num + OS	Logistic Regression	Logistic Regression	59	354	0,7839	0,0289
norm + num + OS	Logistic Regression	Logistic Regression	59	373	0,7839	0,0289
norm + num + OS	Support Vector Machines	Fast Large Margin	59	796	0,7837	0,0292
norm + num + OS	Logistic Regression	Logistic Regression (SVM)	59	1457	0,7833	0,0289
num + OS	Logistic Regression	Logistic Regression (SVM)	59	1481	0,7833	0,0289
norm + disc + OS	Neural Nets	Deep Learning	14	5385	0,7833	0,0469
norm + disc + num + PCA	Support Vector Machines	Support Vector Machine	32	1977	0,7828	0,0303
norm + disc + num + PCA	Support Vector Machines	Support Vector Machine (Evolutionary)	32	1981	0,7828	0,0303
norm + disc + num + PCA	Support Vector Machines	Support Vector Machine (Linear)	32	1991	0,7828	0,0303
disc + num + PCA	Support Vector Machines	Support Vector Machine	32	2023	0,7828	0,0303
disc + num + PCA	Support Vector Machines	Support Vector Machine (Linear)	32	2024	0,7828	0,0303
disc + num + PCA	Support Vector Machines	Support Vector Machine (Evolutionary)	32	2051	0,7828	0,0303
OS	Neural Nets	Deep Learning	17	5749	0,7827	0,0308

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc + OS	Functions	Generalized Linear Model	14	246	0,7816	0,0245
norm + disc	Functions	Generalized Linear Model	15	250	0,7816	0,0245
disc	Functions	Generalized Linear Model	15	251	0,7816	0,0245
disc + OS	Functions	Generalized Linear Model	14	286	0,7816	0,0245
norm	Neural Nets	Deep Learning	20	5646	0,7811	0,0361
disc + OS	Neural Nets	Deep Learning	14	5376	0,7805	0,0324
disc + num + PCA + OS	Trees	Gradient Boosted Trees	28	2497	0,7804	0,0202
norm + disc + num + PCA + OS	Trees	Gradient Boosted Trees	28	2542	0,7804	0,0202
norm + num + PCA + OS	Support Vector Machines	Support Vector Machine (LibSVM)	30	701	0,7800	0,0276
norm	Functions	Generalized Linear Model	20	263	0,7790	0,0309
no transformation	Functions	Generalized Linear Model	20	473	0,7790	0,0309
no transformation	Neural Nets	Deep Learning	20	6423	0,7779	0,0341
norm + OS	Functions	Generalized Linear Model	17	245	0,7777	0,0315
OS	Functions	Generalized Linear Model	17	268	0,7777	0,0315
disc	Neural Nets	Deep Learning	15	5374	0,7771	0,0261
norm + disc + num + PCA	Trees	Gradient Boosted Trees	32	2614	0,7758	0,0187
disc + num + PCA	Trees	Gradient Boosted Trees	32	2659	0,7758	0,0187
norm + num + PCA + OS	Bayesian	Naive Bayes	30	42	0,7758	0,0360
norm + num + PCA	Bayesian	Naive Bayes	32	42	0,7754	0,0359
norm + num + PCA + OS	Neural Nets	Deep Learning	30	5855	0,7749	0,0432
norm + num	Trees	Gradient Boosted Trees	61	2579	0,7742	0,0298
norm + disc + num + PCA	Neural Nets	Deep Learning	32	5838	0,7733	0,0278
disc + num + PCA	Neural Nets	Deep Learning	32	5974	0,7730	0,0435
num	Neural Nets	Deep Learning	61	8277	0,7716	0,0266
num	Trees	Gradient Boosted Trees	61	2532	0,7711	0,0431
norm + num + OS	Trees	Gradient Boosted Trees	59	2445	0,7710	0,0335

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
disc + num + PCA + OS	Bayesian	Naive Bayes	28	39	0,7704	0,0279
norm + disc + num + PCA + OS	Bayesian	Naive Bayes	28	41	0,7704	0,0279
norm + num + PCA	Neural Nets	Deep Learning	32	5926	0,7701	0,0314
disc + OS	Trees	Gradient Boosted Trees	14	1524	0,7700	0,0354
norm + disc + OS	Trees	Gradient Boosted Trees	14	1659	0,7700	0,0354
disc + num	Trees	Gradient Boosted Trees	58	2280	0,7698	0,0301
norm + disc + num	Trees	Gradient Boosted Trees	58	2397	0,7698	0,0301
disc + num + OS	Trees	Gradient Boosted Trees	57	2256	0,7695	0,0299
norm + disc + num + OS	Trees	Gradient Boosted Trees	57	2256	0,7695	0,0299
disc	Trees	Gradient Boosted Trees	15	1548	0,7695	0,0376
norm + disc	Trees	Gradient Boosted Trees	15	1568	0,7695	0,0376
norm + disc + num	Neural Nets	Deep Learning	58	7845	0,7694	0,0385
disc + num	Neural Nets	Deep Learning	58	7954	0,7689	0,0398
num	Bayesian	Naive Bayes (Kernel)	61	107	0,7687	0,0194
norm + disc + num + PCA + OS	Neural Nets	Deep Learning	28	5740	0,7681	0,0250
disc + num + PCA + OS	Neural Nets	Deep Learning	28	5695	0,7660	0,0375
disc + num + PCA	Bayesian	Naive Bayes	32	40	0,7654	0,0402
norm + disc + num + PCA	Bayesian	Naive Bayes	32	42	0,7654	0,0402
norm + num + OS	Neural Nets	Deep Learning	59	8000	0,7648	0,0371
OS	Bayesian	Naive Bayes (Kernel)	17	64	0,7647	0,0225
num + OS	Neural Nets	Deep Learning	59	8103	0,7642	0,0317
num + OS	Trees	Gradient Boosted Trees	59	2365	0,7640	0,0335
num + OS	Support Vector Machines	Fast Large Margin	59	1976	0,7623	0,0423
norm + disc + num + OS	Neural Nets	Deep Learning	57	7726	0,7619	0,0386
disc + num + OS	Neural Nets	AutoMLP	57	293554	0,7596	0,0334
num + OS	Bayesian	Naive Bayes (Kernel)	59	105	0,7595	0,0281

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc + num	Neural Nets	Neural Net	58	115464	0,7593	0,0510
no transformation	Bayesian	Naive Bayes (Kernel)	20	87	0,7588	0,0195
num	Neural Nets	Neural Net	61	128963	0,7586	0,0551
disc + num + OS	Neural Nets	Deep Learning	57	7807	0,7580	0,0254
norm + disc + num + OS	Neural Nets	AutoMLP	57	321560	0,7546	0,0462
norm + num + PCA + OS	Trees	Gradient Boosted Trees	30	2591	0,7545	0,0360
num + OS	Neural Nets	Perceptron	59	118	0,7544	0,0324
norm + disc + num	Neural Nets	AutoMLP	58	316215	0,7537	0,0388
norm + num + PCA	Trees	Gradient Boosted Trees	32	2670	0,7526	0,0468
norm + num + PCA + OS	Neural Nets	AutoMLP	30	264507	0,7508	0,0467
norm + num + PCA	Neural Nets	Neural Net	32	42423	0,7503	0,0351
num + OS	Neural Nets	Neural Net	59	121775	0,7497	0,0389
norm + num + PCA	Neural Nets	AutoMLP	32	254176	0,7476	0,0210
disc + num + PCA + OS	Support Vector Machines	Support Vector Machine (PSO)	28	7932	0,7475	0,0277
disc + num + OS	Neural Nets	Neural Net	57	114673	0,7474	0,0569
num	Neural Nets	AutoMLP	61	282337	0,7468	0,0285
norm + num	Neural Nets	AutoMLP	61	323459	0,7464	0,0388
norm + disc + num + OS	Bayesian	Naive Bayes	57	48	0,7458	0,0524
disc + num + OS	Bayesian	Naive Bayes	57	49	0,7458	0,0524
norm + disc + num + PCA + OS	Support Vector Machines	Support Vector Machine (PSO)	28	7798	0,7448	0,0265
disc + num + PCA + OS	Neural Nets	AutoMLP	28	254954	0,7446	0,0306
disc + num + PCA	Support Vector Machines	Support Vector Machine (PSO)	32	8501	0,7442	0,0310
disc + num	Bayesian	Naive Bayes	58	47	0,7436	0,0550
norm + disc + num	Bayesian	Naive Bayes	58	47	0,7436	0,0550
disc	Rules	Rule Induction	15	311	0,7432	0,0385
disc + num + PCA + OS	Neural Nets	Neural Net	28	34446	0,7430	0,0450

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + num	Bayesian	Naive Bayes	61	49	0,7425	0,0516
num	Bayesian	Naive Bayes	61	59	0,7425	0,0516
norm + num + PCA + OS	Neural Nets	Neural Net	30	39238	0,7415	0,0516
disc + num + OS	Support Vector Machines	Support Vector Machine (PSO)	57	12146	0,7401	0,0267
norm + disc + num + PCA + OS	Neural Nets	AutoMLP	28	273768	0,7400	0,0439
num + OS	Neural Nets	AutoMLP	59	340390	0,7397	0,0229
disc + num	Neural Nets	AutoMLP	58	342807	0,7387	0,0428
norm + num	Neural Nets	Neural Net	61	130891	0,7382	0,0416
norm + num + OS	Neural Nets	Neural Net	59	122051	0,7364	0,0479
norm + num + OS	Bayesian	Naive Bayes	59	50	0,7361	0,0513
num + OS	Bayesian	Naive Bayes	59	53	0,7361	0,0513
norm + disc + num + OS	Neural Nets	Neural Net	57	114117	0,7355	0,0361
norm + disc + num	Support Vector Machines	Support Vector Machine (PSO)	58	12134	0,7337	0,0318
norm + disc + num + PCA	Support Vector Machines	Support Vector Machine (PSO)	32	8478	0,7335	0,0476
disc + num	Support Vector Machines	Support Vector Machine (PSO)	58	12135	0,7329	0,0295
norm + disc + num + PCA	Neural Nets	Neural Net	32	43619	0,7328	0,0448
norm + disc + num + OS	Support Vector Machines	Support Vector Machine (PSO)	57	11950	0,7313	0,0351
disc + num	Neural Nets	Neural Net	58	116487	0,7304	0,0299
disc + num + PCA	Neural Nets	Neural Net	32	41873	0,7298	0,0361
norm + num	Bayesian	Naive Bayes (Kernel)	61	103	0,7294	0,0371
num + OS	Support Vector Machines	Support Vector Machine (LibSVM)	59	1088	0,7276	0,0286
norm + disc	Rules	Rule Induction	15	340	0,7234	0,0498
disc + num + PCA	Neural Nets	AutoMLP	32	252456	0,7231	0,0463
norm + OS	Bayesian	Naive Bayes (Kernel)	17	48	0,7221	0,0361
norm + num + OS	Bayesian	Naive Bayes (Kernel)	59	101	0,7217	0,0438
norm + disc + num + PCA	Neural Nets	AutoMLP	32	276012	0,7216	0,0501

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + num + OS	Neural Nets	AutoMLP	59	345033	0,7212	0,0443
disc + num + PCA	Lazy	k-NN	32	119	0,7191	0,0373
norm + disc + num + PCA	Lazy	k-NN	32	120	0,7191	0,0373
norm + disc + num + OS	Neural Nets	Perceptron	57	81	0,7185	0,0434
disc + num + OS	Neural Nets	Perceptron	57	93	0,7185	0,0434
disc + OS	Rules	Rule Induction	14	349	0,7167	0,0476
norm + disc + num + PCA + OS	Neural Nets	Neural Net	28	34875	0,7162	0,0348
norm + num	Neural Nets	Perceptron	61	93	0,7162	0,0388
disc + num + OS	Rules	Rule Induction	57	2890	0,7141	0,0329
norm + disc + num	Neural Nets	Perceptron	58	98	0,7139	0,0365
disc + num	Neural Nets	Perceptron	58	99	0,7139	0,0365
disc + num	Rules	Rule Induction	58	3002	0,7130	0,0544
norm + disc + OS	Rules	Rule Induction	14	349	0,7115	0,0343
disc + OS	Lazy	k-NN	14	139	0,7105	0,0375
norm + disc + OS	Lazy	k-NN	14	269	0,7105	0,0375
disc	Lazy	k-NN	15	152	0,7102	0,0408
norm + disc + num	Lazy	k-NN	58	183	0,7102	0,0408
disc + num	Lazy	k-NN	58	188	0,7102	0,0408
norm + disc	Lazy	k-NN	15	198	0,7102	0,0408
norm + num + OS	Neural Nets	Perceptron	59	89	0,7100	0,0413
disc + num + PCA + OS	Bayesian	Naive Bayes (Kernel)	28	127	0,7099	0,0554
norm + disc + num + PCA + OS	Bayesian	Naive Bayes (Kernel)	28	127	0,7099	0,0554
norm	Bayesian	Naive Bayes (Kernel)	20	74	0,7086	0,0380
norm	Rules	Rule Induction	20	671	0,7081	0,0373
disc + num + PCA + OS	Lazy	k-NN	28	111	0,7079	0,0316
norm + disc + num + PCA + OS	Lazy	k-NN	28	118	0,7079	0,0316

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc	Trees	Decision Stump	15	38	0,7079	0,0440
norm + disc + OS	Trees	Decision Stump	14	39	0,7079	0,0440
disc	Trees	Decision Stump	15	40	0,7079	0,0440
disc + OS	Trees	Decision Stump	14	41	0,7079	0,0440
norm + disc + num + OS	Lazy	k-NN	57	181	0,7077	0,0371
disc + num + OS	Lazy	k-NN	57	185	0,7077	0,0371
OS	Rules	Rule Induction	17	562	0,7049	0,0336
norm + OS	Rules	Rule Induction	17	518	0,7037	0,0262
norm + disc + num + PCA	Bayesian	Naive Bayes (Kernel)	32	140	0,7026	0,0622
disc + num + PCA	Bayesian	Naive Bayes (Kernel)	32	146	0,7026	0,0622
norm + num + OS	Support Vector Machines	Support Vector Machine (PSO)	59	12418	0,7019	0,0410
norm + num	Lazy	k-NN	61	191	0,7005	0,0391
norm + num + OS	Lazy	k-NN	59	188	0,6995	0,0378
norm + num	Support Vector Machines	Support Vector Machine (PSO)	61	12729	0,6987	0,0365
norm + disc + num	Rules	Rule Induction	58	2873	0,6973	0,0448
num + OS	Rules	Rule Induction	59	4882	0,6970	0,0458
norm + num + OS	Rules	Rule Induction	59	4106	0,6950	0,0378
norm + num + PCA	Support Vector Machines	Support Vector Machine (PSO)	32	8462	0,6937	0,0393
norm + num + PCA + OS	Support Vector Machines	Support Vector Machine (PSO)	30	8028	0,6924	0,0439
norm + disc + num + OS	Rules	Rule Induction	57	2592	0,6921	0,0593
no transformation	Rules	Rule Induction	20	824	0,6915	0,0625
disc + num + PCA	Neural Nets	Perceptron	32	67	0,6908	0,0407
norm + disc + num + PCA	Neural Nets	Perceptron	32	75	0,6908	0,0407
disc + num + PCA + OS	Neural Nets	Perceptron	28	63	0,6905	0,0616
norm + disc + num + PCA + OS	Neural Nets	Perceptron	28	71	0,6905	0,0616
norm + num + PCA	Bayesian	Naive Bayes (Kernel)	32	142	0,6876	0,0688

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + num + PCA + OS	Lazy	k-NN	30	115	0,6863	0,0280
norm + OS	Lazy	k-NN	17	136	0,6860	0,0511
norm + num + PCA	Lazy	k-NN	32	223	0,6851	0,0482
norm + num + PCA + OS	Bayesian	Naive Bayes (Kernel)	30	143	0,6821	0,0720
norm + num + PCA + OS	Neural Nets	Perceptron	30	71	0,6810	0,0447
norm + num + PCA	Neural Nets	Perceptron	32	68	0,6805	0,0584
norm + disc + num	Trees	Decision Stump	58	163	0,6719	0,0484
norm + disc + num + OS	Trees	Decision Stump	57	165	0,6719	0,0484
disc + num	Trees	Decision Stump	58	166	0,6719	0,0484
disc + num + OS	Trees	Decision Stump	57	175	0,6719	0,0484
norm	Lazy	k-NN	20	152	0,6699	0,0459
disc + num + PCA + OS	Rules	Rule Induction	28	1281	0,6623	0,0670
norm + disc + OS	Trees	Decision Tree (Weight-Based)	14	2381	0,6552	0,0646
disc + OS	Trees	Decision Tree (Weight-Based)	14	2555	0,6552	0,0646
norm + disc + OS	Trees	CHAID	14	2065	0,6543	0,0652
disc + OS	Trees	CHAID	14	2083	0,6543	0,0652
norm + disc	Trees	Decision Tree (Weight-Based)	15	2728	0,6518	0,0613
disc	Trees	Decision Tree (Weight-Based)	15	2824	0,6518	0,0613
num	Rules	Rule Induction	61	4705	0,6509	0,0507
norm + disc	Trees	CHAID	15	2277	0,6509	0,0619
disc	Trees	CHAID	15	2521	0,6509	0,0619
norm + num	Rules	Rule Induction	61	4182	0,6455	0,0469
norm + disc + num + PCA + OS	Rules	Rule Induction	28	1303	0,6445	0,0543
norm + disc + num + PCA	Rules	Rule Induction	32	1492	0,6417	0,0725
disc + num + PCA	Rules	Rule Induction	32	1418	0,6406	0,0498
num + OS	Support Vector Machines	Support Vector Machine (PSO)	59	13838	0,6340	0,0707

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
disc + num + PCA + OS	Logistic Regression	Logistic Regression (Evolutionary)	28	25798	0,6328	0,0454
norm + disc + num + PCA + OS	Logistic Regression	Logistic Regression (Evolutionary)	28	23737	0,6245	0,0401
norm + num + PCA	Rules	Rule Induction	32	1715	0,6219	0,0554
norm + disc + num + PCA	Logistic Regression	Logistic Regression (Evolutionary)	32	30154	0,6190	0,0602
norm + num + PCA + OS	Rules	Rule Induction	30	1382	0,6190	0,0733
disc + num + PCA	Logistic Regression	Logistic Regression (Evolutionary)	32	24860	0,6173	0,0540
num + OS	Lazy	k-NN	59	192	0,6023	0,0586
disc + OS	Rules	Single Rule Induction (Single Attribute)	14	62	0,5980	0,0522
norm + disc + OS	Rules	Single Rule Induction (Single Attribute)	14	62	0,5980	0,0522
disc	Rules	Single Rule Induction (Single Attribute)	15	65	0,5980	0,0522
norm + disc	Rules	Single Rule Induction (Single Attribute)	15	65	0,5980	0,0522
disc + OS	Trees	ID3	14	182	0,5947	0,0476
norm + disc + OS	Trees	ID3	14	185	0,5947	0,0476
disc	Trees	ID3	15	187	0,5923	0,0483
norm + disc	Trees	ID3	15	195	0,5923	0,0483
num + OS	Rules	Single Rule Induction (Single Attribute)	59	316	0,5881	0,0658
norm + num + OS	Rules	Single Rule Induction (Single Attribute)	59	350	0,5881	0,0658
disc + num + OS	Logistic Regression	Logistic Regression (Evolutionary)	57	31271	0,5840	0,0497
norm	Trees	Random Tree	20	43	0,5829	0,1040
norm + disc + num + OS	Logistic Regression	Logistic Regression (Evolutionary)	57	29820	0,5826	0,0524
num	Support Vector Machines	Fast Large Margin	61	2240	0,5806	0,0786
norm + disc + num + PCA + OS	Trees	Random Forest	28	354	0,5785	0,0775
norm + disc + num + OS	Rules	Single Rule Induction (Single Attribute)	57	289	0,5781	0,0408
disc + num	Rules	Single Rule Induction (Single Attribute)	58	290	0,5781	0,0408
norm + disc + num	Rules	Single Rule Induction (Single Attribute)	58	290	0,5781	0,0408
disc + num + OS	Rules	Single Rule Induction (Single Attribute)	57	291	0,5781	0,0408

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
num + PCA	Bayesian	Naive Bayes (Kernel)	1	37	0,5770	0,0511
num + PCA + OS	Bayesian	Naive Bayes (Kernel)	1	40	0,5770	0,0511
disc + num	Logistic Regression	Logistic Regression (Evolutionary)	58	28989	0,5768	0,0514
num + PCA	Bayesian	Naive Bayes	1	34	0,5765	0,0380
num + PCA + OS	Bayesian	Naive Bayes	1	35	0,5765	0,0380
norm + disc + num	Logistic Regression	Logistic Regression (Evolutionary)	58	30095	0,5753	0,0549
disc	Trees	Random Tree	15	38	0,5737	0,0960
num	Trees	Random Forest	61	300	0,5655	0,0754
norm + num + PCA + OS	Logistic Regression	Logistic Regression (Evolutionary)	30	24470	0,5642	0,0443
num + PCA + OS	Trees	Gradient Boosted Trees	1	1026	0,5620	0,0633
num + PCA	Trees	Gradient Boosted Trees	1	1095	0,5620	0,0633
norm + num + PCA	Logistic Regression	Logistic Regression (Evolutionary)	32	24585	0,5617	0,0357
norm + num + OS	Logistic Regression	Logistic Regression (Evolutionary)	59	28655	0,5610	0,0384
norm + num	Logistic Regression	Logistic Regression (Evolutionary)	61	32345	0,5605	0,0358
num + PCA + OS	Neural Nets	Deep Learning	1	4240	0,5584	0,0433
disc + OS	Trees	Random Tree	14	39	0,5577	0,0940
num	Support Vector Machines	Support Vector Machine (LibSVM)	61	2484	0,5574	0,0569
num + OS	Logistic Regression	Logistic Regression (Evolutionary)	59	28040	0,5571	0,0485
OS	Trees	Random Tree	17	52	0,5565	0,0919
num + PCA + OS	Rules	Rule Induction	1	216	0,5550	0,0616
num + PCA + OS	Support Vector Machines	Support Vector Machine (PSO)	1	7938	0,5535	0,0553
disc + num + PCA + OS	Trees	Random Forest	28	363	0,5530	0,0742
num + PCA + OS	Functions	Linear Regression	1	45	0,5529	0,0478
num + PCA	Functions	Linear Regression	1	53	0,5529	0,0478
num + PCA	Logistic Regression	Logistic Regression (SVM)	1	157	0,5529	0,0478
num + PCA + OS	Logistic Regression	Logistic Regression (SVM)	1	157	0,5529	0,0478

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
num + PCA + OS	Logistic Regression	Logistic Regression	1	171	0,5529	0,0478
num + PCA + OS	Functions	Generalized Linear Model	1	179	0,5529	0,0478
num + PCA	Functions	Generalized Linear Model	1	196	0,5529	0,0478
num + PCA	Logistic Regression	Logistic Regression	1	196	0,5529	0,0478
num + PCA	Neural Nets	Neural Net	1	2776	0,5529	0,0478
num + PCA + OS	Neural Nets	Neural Net	1	2778	0,5529	0,0478
num + PCA + OS	Neural Nets	AutoMLP	1	212976	0,5529	0,0478
num + PCA	Neural Nets	AutoMLP	1	215229	0,5529	0,0478
num	Lazy	k-NN	61	196	0,5506	0,0499
no transformation	Lazy	k-NN	20	200	0,5486	0,0514
OS	Lazy	k-NN	17	146	0,5486	0,0566
num + PCA	Rules	Rule Induction	1	312	0,5475	0,0807
disc + num + OS	Trees	Random Forest	57	274	0,5469	0,0727
num + PCA	Support Vector Machines	Support Vector Machine (PSO)	1	6759	0,5464	0,0361
num + PCA	Neural Nets	Deep Learning	1	4190	0,5453	0,0410
norm + num	Trees	Random Forest	61	283	0,5446	0,0684
norm + OS	Trees	Random Tree	17	39	0,5446	0,0893
norm + OS	Rules	Single Rule Induction (Single Attribute)	17	250	0,5445	0,0671
OS	Rules	Single Rule Induction (Single Attribute)	17	268	0,5445	0,0671
norm	Rules	Single Rule Induction (Single Attribute)	20	275	0,5445	0,0671
no transformation	Rules	Single Rule Induction (Single Attribute)	20	352	0,5445	0,0671
num	Rules	Single Rule Induction (Single Attribute)	61	550	0,5445	0,0671
norm + num	Rules	Single Rule Induction (Single Attribute)	61	552	0,5445	0,0671
norm + num + OS	Trees	Random Forest	59	286	0,5393	0,0789
disc + num + PCA	Trees	Random Forest	32	405	0,5374	0,0417
norm + disc	Trees	Random Tree	15	34	0,5373	0,0810

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + num + PCA	Trees	Random Forest	32	403	0,5365	0,0387
norm + num + PCA + OS	Trees	Random Forest	30	377	0,5324	0,0412
num + PCA + OS	Rules	Single Rule Induction (Single Attribute)	1	780	0,5269	0,0492
num + PCA	Rules	Single Rule Induction (Single Attribute)	1	884	0,5269	0,0492
num + PCA	Trees	Random Forest	1	119	0,5268	0,0182
num + PCA + OS	Support Vector Machines	Support Vector Machine (LibSVM)	1	1591	0,5265	0,0497
num + PCA	Support Vector Machines	Support Vector Machine (LibSVM)	1	1634	0,5265	0,0497
num + PCA	Lazy	k-NN	1	52	0,5241	0,0605
num + PCA + OS	Lazy	k-NN	1	54	0,5241	0,0605
num + PCA + OS	Trees	Random Forest	1	116	0,5235	0,0204
norm + disc + OS	Trees	Random Tree	14	34	0,5230	0,0690
num + PCA + OS	Support Vector Machines	Support Vector Machine (Linear)	1	244	0,5229	0,0664
num + PCA	Support Vector Machines	Support Vector Machine (Evolutionary)	1	247	0,5229	0,0664
num + PCA + OS	Support Vector Machines	Support Vector Machine (Evolutionary)	1	247	0,5229	0,0664
num + PCA	Support Vector Machines	Support Vector Machine (Linear)	1	248	0,5229	0,0664
num + PCA + OS	Support Vector Machines	Support Vector Machine	1	248	0,5229	0,0664
num + PCA	Support Vector Machines	Support Vector Machine	1	258	0,5229	0,0664
norm + num + OS	Trees	Decision Stump	59	171	0,5221	0,0664
num + OS	Trees	Decision Stump	59	186	0,5221	0,0664
norm + disc + num	Trees	Random Forest	58	269	0,5202	0,0576
norm + disc + num + PCA	Trees	Random Forest	32	372	0,5185	0,0424
norm	Trees	Random Forest	20	160	0,5124	0,0178
OS	Trees	Random Forest	17	162	0,5108	0,0206
norm + OS	Trees	Random Forest	17	140	0,5104	0,0105
num + OS	Trees	Random Tree	59	52	0,5102	0,0307
num	Trees	Random Tree	61	55	0,5102	0,0307

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc + num + OS	Trees	Random Forest	57	271	0,5088	0,0331
disc + num	Trees	Random Forest	58	279	0,5086	0,0227
num	Neural Nets	Perceptron	61	163	0,5074	0,0222
no transformation	Trees	Random Forest	20	177	0,5052	0,0138
num + OS	Trees	Random Forest	59	285	0,5052	0,0158
num + PCA	Rules	Tree to Rules	1	39	0,5002	0,0007
num + PCA + OS	Rules	Tree to Rules	1	40	0,5002	0,0007
num + PCA	Trees	Decision Tree	1	41	0,5002	0,0007
num + PCA + OS	Trees	Decision Tree	1	41	0,5002	0,0007
norm + disc + OS	Lazy	Default Model	14	32	0,5000	0,0000
norm + disc + num + PCA	Lazy	Default Model	32	32	0,5000	0,0000
disc + num + PCA + OS	Lazy	Default Model	28	33	0,5000	0,0000
norm + OS	Lazy	Default Model	17	33	0,5000	0,0000
num + PCA + OS	Lazy	Default Model	1	34	0,5000	0,0000
norm	Lazy	Default Model	20	34	0,5000	0,0000
norm + num + OS	Lazy	Default Model	59	34	0,5000	0,0000
norm + disc	Lazy	Default Model	15	34	0,5000	0,0000
norm + disc + OS	Rules	Tree to Rules	14	34	0,5000	0,0000
norm + disc + num + PCA + OS	Lazy	Default Model	28	34	0,5000	0,0000
disc + OS	Trees	Decision Tree	14	35	0,5000	0,0000
disc + num	Lazy	Default Model	58	35	0,5000	0,0000
norm + num	Lazy	Default Model	61	35	0,5000	0,0000
norm + disc	Trees	Decision Tree	15	35	0,5000	0,0000
norm + disc	Rules	Tree to Rules	15	35	0,5000	0,0000
norm + disc + OS	Trees	Decision Tree	14	35	0,5000	0,0000
norm + disc + num	Lazy	Default Model	58	35	0,5000	0,0000

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
num + PCA	Lazy	Default Model	1	36	0,5000	0,0000
disc	Trees	Decision Tree	15	36	0,5000	0,0000
disc	Rules	Tree to Rules	15	36	0,5000	0,0000
disc + OS	Lazy	Default Model	14	36	0,5000	0,0000
disc + num + OS	Lazy	Default Model	57	36	0,5000	0,0000
norm + num + PCA	Lazy	Default Model	32	36	0,5000	0,0000
num + PCA + OS	Trees	Random Tree	1	37	0,5000	0,0000
disc + OS	Rules	Tree to Rules	14	37	0,5000	0,0000
disc + num + PCA	Lazy	Default Model	32	37	0,5000	0,0000
norm + disc + OS	Trees	Decision Tree (Multiway)	14	37	0,5000	0,0000
norm + disc + num + OS	Lazy	Default Model	57	37	0,5000	0,0000
num + PCA	Discriminant Analysis	Quadratic Discriminant Analysis	1	38	0,5000	0,0000
norm + num + PCA + OS	Lazy	Default Model	30	38	0,5000	0,0000
norm + disc	Trees	Decision Tree (Multiway)	15	38	0,5000	0,0000
num + OS	Lazy	Default Model	59	39	0,5000	0,0000
num + PCA	Trees	Random Tree	1	39	0,5000	0,0000
num + PCA + OS	Support Vector Machines	Hyper Hyper	1	39	0,5000	0,0000
num + PCA + OS	Discriminant Analysis	Linear Discriminant Analysis	1	39	0,5000	0,0000
disc	Lazy	Default Model	15	39	0,5000	0,0000
disc	Trees	Decision Tree (Multiway)	15	39	0,5000	0,0000
disc + OS	Trees	Decision Tree (Multiway)	14	39	0,5000	0,0000
num	Lazy	Default Model	61	40	0,5000	0,0000
num + PCA + OS	Discriminant Analysis	Quadratic Discriminant Analysis	1	40	0,5000	0,0000
OS	Lazy	Default Model	17	41	0,5000	0,0000
num + PCA	Trees	Decision Stump	1	41	0,5000	0,0000
num + PCA	Support Vector Machines	Hyper Hyper	1	41	0,5000	0,0000

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
num + PCA	Discriminant Analysis	Linear Discriminant Analysis	1	41	0,5000	0,0000
num + PCA + OS	Trees	Decision Stump	1	41	0,5000	0,0000
num + PCA + OS	Discriminant Analysis	Regularized Discriminant Analysis	1	41	0,5000	0,0000
num + PCA	Discriminant Analysis	Regularized Discriminant Analysis	1	43	0,5000	0,0000
norm + OS	Trees	Decision Stump	17	48	0,5000	0,0000
norm + disc + num	Trees	Random Tree	58	48	0,5000	0,0000
no transformation	Trees	Random Tree	20	49	0,5000	0,0000
norm + num + OS	Trees	Random Tree	59	49	0,5000	0,0000
norm + num	Trees	Random Tree	61	50	0,5000	0,0000
no transformation	Lazy	Default Model	20	51	0,5000	0,0000
disc + num	Trees	Random Tree	58	51	0,5000	0,0000
norm + disc + num + OS	Trees	Random Tree	57	52	0,5000	0,0000
disc + num + OS	Trees	Random Tree	57	53	0,5000	0,0000
norm + num + PCA	Trees	Random Tree	32	58	0,5000	0,0000
norm + num + PCA + OS	Trees	Random Tree	30	60	0,5000	0,0000
norm + disc + num + PCA	Trees	Random Tree	32	61	0,5000	0,0000
OS	Trees	Decision Stump	17	62	0,5000	0,0000
num	Support Vector Machines	Hyper Hyper	61	65	0,5000	0,0000
disc + num + PCA	Trees	Random Tree	32	65	0,5000	0,0000
disc + num + PCA + OS	Discriminant Analysis	Linear Discriminant Analysis	28	66	0,5000	0,0000
disc + num + PCA + OS	Discriminant Analysis	Quadratic Discriminant Analysis	28	67	0,5000	0,0000
norm + num + PCA + OS	Discriminant Analysis	Linear Discriminant Analysis	30	69	0,5000	0,0000
norm + disc + num + PCA + OS	Trees	Random Tree	28	69	0,5000	0,0000
norm + num + PCA + OS	Discriminant Analysis	Quadratic Discriminant Analysis	30	71	0,5000	0,0000
norm + disc + num + PCA + OS	Discriminant Analysis	Quadratic Discriminant Analysis	28	71	0,5000	0,0000
disc + num + PCA	Discriminant Analysis	Quadratic Discriminant Analysis	32	72	0,5000	0,0000

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc + num + PCA	Discriminant Analysis	Quadratic Discriminant Analysis	32	72	0,5000	0,0000
disc + num + PCA	Discriminant Analysis	Linear Discriminant Analysis	32	73	0,5000	0,0000
norm + num + PCA	Discriminant Analysis	Linear Discriminant Analysis	32	74	0,5000	0,0000
norm + disc + num + PCA + OS	Discriminant Analysis	Linear Discriminant Analysis	28	74	0,5000	0,0000
norm	Trees	Decision Stump	20	75	0,5000	0,0000
norm + num + PCA	Discriminant Analysis	Quadratic Discriminant Analysis	32	75	0,5000	0,0000
norm + disc + num + PCA	Discriminant Analysis	Linear Discriminant Analysis	32	76	0,5000	0,0000
disc + num + PCA + OS	Discriminant Analysis	Regularized Discriminant Analysis	28	77	0,5000	0,0000
norm + disc + num + OS	Rules	Tree to Rules	57	77	0,5000	0,0000
disc + num	Rules	Tree to Rules	58	78	0,5000	0,0000
norm + disc + num + OS	Trees	Decision Tree	57	78	0,5000	0,0000
norm + disc + num	Rules	Tree to Rules	58	79	0,5000	0,0000
disc + num + OS	Trees	Decision Tree	57	80	0,5000	0,0000
norm + disc + num	Trees	Decision Tree	58	80	0,5000	0,0000
disc + num	Trees	Decision Tree	58	81	0,5000	0,0000
norm + num + OS	Rules	Tree to Rules	59	81	0,5000	0,0000
norm + num + PCA + OS	Discriminant Analysis	Regularized Discriminant Analysis	30	82	0,5000	0,0000
norm + num + OS	Trees	Decision Tree	59	83	0,5000	0,0000
num + OS	Trees	Decision Tree	59	85	0,5000	0,0000
num + OS	Rules	Tree to Rules	59	85	0,5000	0,0000
no transformation	Trees	Decision Stump	20	86	0,5000	0,0000
norm + disc + num + PCA	Discriminant Analysis	Regularized Discriminant Analysis	32	86	0,5000	0,0000
norm + disc + num + PCA + OS	Discriminant Analysis	Regularized Discriminant Analysis	28	86	0,5000	0,0000
disc + num + PCA	Discriminant Analysis	Regularized Discriminant Analysis	32	88	0,5000	0,0000
norm + num + PCA	Discriminant Analysis	Regularized Discriminant Analysis	32	89	0,5000	0,0000
disc + num + OS	Rules	Tree to Rules	57	95	0,5000	0,0000

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + disc + OS	Trees	Random Forest	14	109	0,5000	0,0000
disc + OS	Trees	Random Forest	14	110	0,5000	0,0000
disc + num + PCA + OS	Trees	Random Tree	28	111	0,5000	0,0000
disc	Trees	Random Forest	15	112	0,5000	0,0000
norm + disc	Trees	Random Forest	15	114	0,5000	0,0000
norm + disc + num	Trees	Decision Tree (Multiway)	58	121	0,5000	0,0000
disc + num + OS	Trees	Decision Tree (Multiway)	57	122	0,5000	0,0000
disc + num	Trees	Decision Tree (Multiway)	58	123	0,5000	0,0000
norm + disc + num + OS	Trees	Decision Tree (Multiway)	57	123	0,5000	0,0000
norm + disc + num + OS	Discriminant Analysis	Linear Discriminant Analysis	57	131	0,5000	0,0000
norm + num + OS	Trees	Decision Tree (Multiway)	59	133	0,5000	0,0000
disc + num + OS	Discriminant Analysis	Linear Discriminant Analysis	57	138	0,5000	0,0000
norm + disc + num	Discriminant Analysis	Linear Discriminant Analysis	58	139	0,5000	0,0000
norm + num + OS	Discriminant Analysis	Linear Discriminant Analysis	59	140	0,5000	0,0000
disc + num	Discriminant Analysis	Linear Discriminant Analysis	58	141	0,5000	0,0000
num + OS	Trees	Decision Tree (Multiway)	59	142	0,5000	0,0000
num + OS	Discriminant Analysis	Linear Discriminant Analysis	59	145	0,5000	0,0000
norm + num	Discriminant Analysis	Linear Discriminant Analysis	61	155	0,5000	0,0000
disc + num + OS	Discriminant Analysis	Regularized Discriminant Analysis	57	172	0,5000	0,0000
norm + disc + num + OS	Discriminant Analysis	Regularized Discriminant Analysis	57	174	0,5000	0,0000
disc + num + PCA + OS	Trees	Decision Stump	28	177	0,5000	0,0000
norm + disc + num	Discriminant Analysis	Regularized Discriminant Analysis	58	178	0,5000	0,0000
disc + num	Discriminant Analysis	Regularized Discriminant Analysis	58	182	0,5000	0,0000
norm + num + OS	Discriminant Analysis	Regularized Discriminant Analysis	59	182	0,5000	0,0000
num + OS	Discriminant Analysis	Regularized Discriminant Analysis	59	183	0,5000	0,0000
num	Discriminant Analysis	Linear Discriminant Analysis	61	184	0,5000	0,0000

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + num	Trees	Decision Stump	61	184	0,5000	0,0000
norm + disc + num + PCA + OS	Trees	Decision Stump	28	186	0,5000	0,0000
norm + num + PCA + OS	Trees	Decision Stump	30	188	0,5000	0,0000
num	Trees	Decision Stump	61	195	0,5000	0,0000
norm + disc + num + PCA	Trees	Decision Stump	32	196	0,5000	0,0000
norm + num + PCA	Trees	Decision Stump	32	197	0,5000	0,0000
norm + num	Discriminant Analysis	Regularized Discriminant Analysis	61	207	0,5000	0,0000
num	Discriminant Analysis	Regularized Discriminant Analysis	61	209	0,5000	0,0000
disc + num + PCA	Trees	Decision Stump	32	238	0,5000	0,0000
disc + num + PCA + OS	Trees	Decision Tree	28	419	0,5000	0,0000
norm + disc + num + PCA + OS	Rules	Tree to Rules	28	420	0,5000	0,0000
disc + num + PCA + OS	Rules	Tree to Rules	28	427	0,5000	0,0000
norm + disc + num + PCA + OS	Trees	Decision Tree	28	428	0,5000	0,0000
norm + disc + num + PCA	Rules	Tree to Rules	32	590	0,5000	0,0000
norm + disc + num + PCA	Trees	Decision Tree	32	594	0,5000	0,0000
disc + num + PCA	Trees	Decision Tree	32	599	0,5000	0,0000
disc + num + PCA	Rules	Tree to Rules	32	602	0,5000	0,0000
norm + num + PCA + OS	Trees	Decision Tree	30	856	0,5000	0,0000
norm + num + PCA + OS	Rules	Tree to Rules	30	861	0,5000	0,0000
norm + num + PCA	Trees	Decision Tree	32	983	0,5000	0,0000
norm + num + PCA	Rules	Tree to Rules	32	988	0,5000	0,0000
disc + OS	Rules	Subgroup Discovery	14	6534	0,5000	0,0000
norm + disc + OS	Rules	Subgroup Discovery	14	6694	0,5000	0,0000
norm + disc	Rules	Subgroup Discovery	15	7164	0,5000	0,0000
disc	Rules	Subgroup Discovery	15	8472	0,5000	0,0000
norm + num + PCA + OS	Rules	Single Rule Induction (Single Attribute)	30	17728	0,5000	0,0000

Transformation Explanation	Family	Model Name	# Attributes	Time (ms)	AUC	AUC STDEV
norm + num + PCA	Rules	Single Rule Induction (Single Attribute)	32	18851	0,5000	0,0000
norm + OS	Trees	Decision Tree	17	44	0,4984	0,0049
norm + OS	Rules	Tree to Rules	17	45	0,4984	0,0049
OS	Rules	Tree to Rules	17	51	0,4984	0,0049
norm	Trees	Decision Tree	20	51	0,4984	0,0049
OS	Trees	Decision Tree	17	54	0,4984	0,0049
no transformation	Rules	Tree to Rules	20	65	0,4984	0,0049
norm	Rules	Tree to Rules	20	67	0,4984	0,0049
no transformation	Trees	Decision Tree	20	74	0,4984	0,0049
norm + num	Trees	Decision Tree	61	95	0,4984	0,0049
num	Trees	Decision Tree	61	100	0,4984	0,0049
num	Rules	Tree to Rules	61	100	0,4984	0,0049
norm + num	Rules	Tree to Rules	61	100	0,4984	0,0049
num	Logistic Regression	Logistic Regression (Evolutionary)	61	30425	0,4970	0,0104
num + PCA	Support Vector Machines	Fast Large Margin	1	344	0,4880	0,0702
num + PCA + OS	Support Vector Machines	Fast Large Margin	1	347	0,4880	0,0702
num	Support Vector Machines	Support Vector Machine (PSO)	61	15733	0,4857	0,0223
disc + num + PCA + OS	Rules	Single Rule Induction (Single Attribute)	28	9185	0,4802	0,0751
norm + disc + num + PCA + OS	Rules	Single Rule Induction (Single Attribute)	28	9614	0,4802	0,0751
norm + disc + num + PCA	Rules	Single Rule Induction (Single Attribute)	32	11007	0,4765	0,0733
disc + num + PCA	Rules	Single Rule Induction (Single Attribute)	32	11191	0,4765	0,0733
num + PCA	Logistic Regression	Logistic Regression (Evolutionary)	1	24891	0,4750	0,0551
num + PCA + OS	Logistic Regression	Logistic Regression (Evolutionary)	1	24391	0,4735	0,0654
num + PCA	Neural Nets	Perceptron	1	41	0,3245	0,1170
num + PCA + OS	Neural Nets	Perceptron	1	41	0,3245	0,1170