Saarland University

Bachelor Thesis

Explainable Classification for an Application System

Submitted by:

Paul Nikolaus Krieger

Submitted on:

September 3, 2020

Reviewers:

Univ.-Prof. Dr. Verena Wolf

Univ.-Prof. Dr. Jan Reineke

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,_____ _____

(Datum/Date) (Unterschrift/Signature)

Abstract

The objective of this thesis was the examination of the use of artificial intelligence for the admission of students to the master programme in computer science of Saarland University. Given a set of student applications, the aim was to develop a classification approach based on decision trees and random forests to provide a recommendation of acceptance or rejection of a student. In addition to the decision engine a variety of data cleaning methods was developed to make the input data set compatible with the machine learning models' requirements. Also, features such as spelling of the applicant and university rankings were added to the data as part of feature engineering. The data was also labelled. This was a particular challenge, as only a few applicants in the data set were accepted at Saarland University and the labelling was based on their performance. An important part of this thesis was the explainability of the trained models. For that approaches were developed to make the reasoning of the models understandable for humans. These approaches are based on visualisations of the decision trees and the analysis of the most important features. The available application data were labelled in several ways and models could be trained with a high accuracy. Furthermore, the performance of the decision trees and random forests was compared to gradient boosting models. The final part of the thesis provides a critical analysis of the practical suitability of the trained models, which revealed that the recommendation quality is questionable for several reasons. For example, the performance of the models is very poor, if a student is from an university which the models have not seen before.

Acknowledgments

This Bachelor thesis was written at the Faculty of Mathematics and Computer Science at Saarland University. At this point I would like to thank my supervisors Prof. Dr. Verena Wolf and Timo Gros for their friendly and dedicated support as well as for helpful suggestions and ideas for my thesis. I would also like to thank the study coordination for providing the data and explaining it.

Contents

1. Introduction

1.1. Focus of this Work

Artificial intelligence is an increasingly important area of computer science and is used in many areas of our lives. In recent years, numerous successes have been achieved, especially in the field of machine learning. These successes are partly due to scientific progress, which uses complex algorithms to find patterns in data that no human being would discover. On the other hand, they are also due to the enormous increase in the amount of data that is collected every day [40, 73]. Meanwhile there are agents based on artificial intelligence which are able to win highly complex games like chess or go against world leading players [1]. But impressive results have also been achieved in the areas of autonomous driving, education, manufacturing, recruitment and many other areas. This variety of possible applications has already led to artificial intelligence-based applications being used in many areas of our lives, and this trend is likely to increase in the future [20]. For example, systems that make decisions automatically have an ever-increasing impact on our lives. It is therefore of enormous importance that these systems function very well. However, the performance not only plays an important role here, but also whether one can trust these systems and understand why they behave the way they do. For example, artificial intelligence can be used to support doctors in diagnosing diseases [40, 73]. When such a tool diagnoses a disease, it is desirable that the tool also provides reasons why it believes the patient has the disease. The doctor can then check whether the diagnosis is plausible. The same applies to tools that support recruitment. If an artificial intelligence wants to accept an applicant, it is important to know what the decisive reasons were. The need to understand why a decision is made is also shown by the example of amazon, which has developed a tool to optimize the application process. The tool, which is based on artificial intelligence, sorted out CVs that contained the word women's [65]. Such practices should obviously be avoided and the goal should be to ensure that the artificial intelligence is not only performance-driven, but also explainable. In this context, one also speaks of explainable AI, which should ensure the explainability, comprehensibility and traceability of artificial intelligence. Explainable AI also has the opportunity to create more trust among users. Apart from that, there are also legal regulations that require an explanation of artificial intelligence. This is defined as the right of explanation in the new European data protection regulation (DSGVO, see also with ISO/IEC 27001) [40, 73]. However, it often relatively difficult to provide explainability. For example, words are mapped to high-dimensional vectors and are therefore no longer understandable for humans. Also the explainability depends on the machine learning algorithm used. The decisions of models like neural networks or support vector machines are not comprehensible for humans, they are so-called black box models. But there are also models, such as decision trees, which are easy to understand. Decision trees have the advantage that the decision can be easily understood on the basis of simple rules, which are ordered hierarchically.

Furthermore, decision trees show a good performance in practice, even if it is not as good as that of neural networks [40, 57, 73].

This thesis is about the explainable classification of applicant data. The underlying data set is the applicant data set for the master's programme in computer science at Saarland University. At Saarland University there is a large number of international applicants with different backgrounds. The applicants differ in personal and academic characteristics. There are differences in age, gender, origin or course of study as well as grades. For professors and the admissions office, this multitude of profiles is difficult to keep track of and complicates the admissions process. There are also groups of students who do not achieve internal goals of the faculty in terms of grades and duration of study. In order to solve these problems, the application of artificial intelligence in the admissions process will now be investigated. However, we will mainly work with decision trees, as they are easy to understand and only support the selection committee in its decision making. For comparison, other machine learning methods such as random forests and gradient boosting are also considered. For these models, possibilities are also presented that allow a better understanding of their decisions. The main steps of a machine learning project will be worked out.

1.2. Process Model

In applied machine learning and data science, various process models have been established for projects. However, most of them differ only in nuances. In principle, the essential steps which are listed in the data science lifecycle of Microsoft Azure are processed here [54]. The lifecycle is depicted in figure 1.1. An important step in any machine learning project is business understanding. This involves developing a sound understanding of the underlying problem. In this project, the problem is that the admission of students to the master programme at Saarland University is not optimal. Some students do not achieve the internal goals of the computer science faculty. Machine learning methods will now be used to solve this problem. In order to solve this problem by means of machine learning, it is important to become familiar with the application process of the university and to clearly define internal requirements. Only in this way such a project can succeed. It is also determined what data is needed to carry out such a project. This also leads to the next elementary step in the lifecycle of any data science and machine learning project, the data acquisition and understanding. In this project, anonymised data were obtained by the study coordination. This includes all data collected during the application process and during the study. An essential step is now to thoroughly understand these data. This includes the essential steps such as data cleaning and data analysis. The data analysis especially gives us the possibility to understand the data, to discover special features like outliers and null values and to prepare the data set for the actual machine learning. In general, there are also other things, such as pipelining the data, which must be considered in this step [54, 62]. However, these are not necessary in this project, because there is only one CSV file with a manageable size. After the data is available, cleaned and analysed, the actual modelling can begin in a machine learning project. The elementary steps in modelling

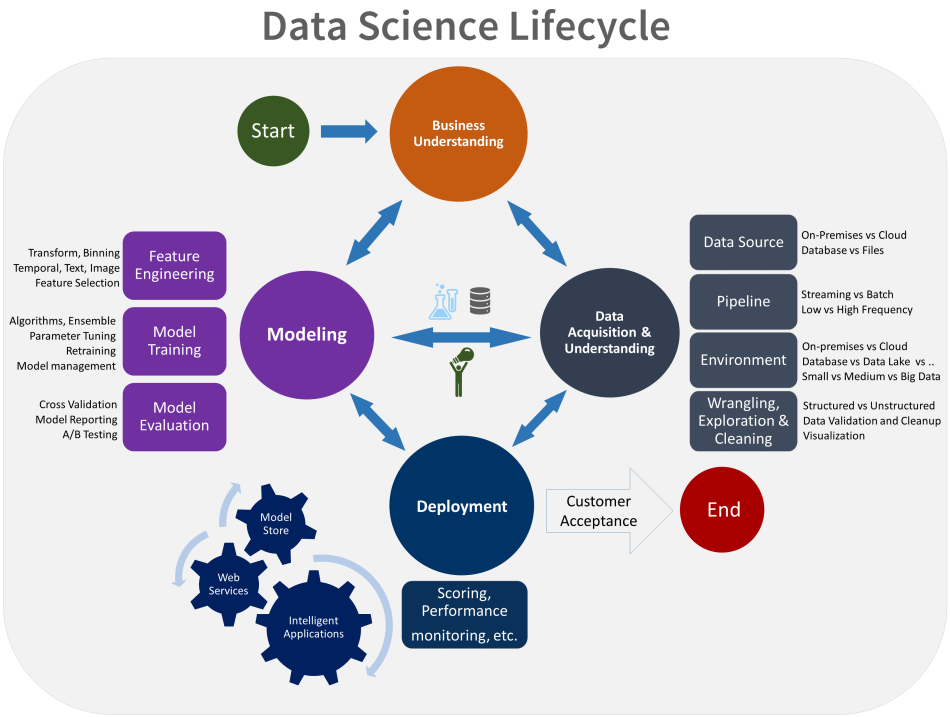


Figure 1.1.: Data science lifecycle from Microsoft Azure. Reprinted from [54].

are feature engineering, model selection and the evaluation of the selected models. In feature engineering, new features are extracted from the existing data. These can lead to a considerable improvement in performance [54]. In model training, the selected models are trained on the data. Depending on the data, this can be unsupervised, supervised or based on reinforcement learning. In this project we only work with methods that are based on decision trees (also random forests). Afterwards the trained model must also be evaluated. Metrics such as accuracy, precision and recall are suitable for this. In order to ensure a high level of security with regard to these values, the results can be cross-validated. A visualisation of the results is also part of this step. Confusion matrices, for example, can be used to visualise the performance well [26, 54]. Ultimately, this is how the best model is selected, i.e. the model to be used in the future. The last step is the deployment. The modelling is finished and the model is ready for deployment. The performance of the model is monitored to ensure correctness. If it works well, the tool is left to the customer as much as possible and the project is finished [54].

1.3. Organisation of the Thesis

This thesis examines whether decision trees and random forests can be used to optimise the application process for the master program in computer science at Saarland University. Chapter 2 introduces and defines basic concepts and provides algorithms for decision trees and random forests. Also, techniques to improve explainability are

presented. In addition, techniques for working with unbalanced data and metrics for evaluating the models are defined. In the following chapters, all relevant steps of a practical machine learning project are covered. In chapter 3 the data set is presented, cleaned and analysed. Chapter 4 deals with modelling. New features are created and the data is labelled. In the subsequent chapter models are trained, optimised and evaluated. In chapter 6 approaches to explain the decision of the model are presented and in chapter 7 the results of the thesis are summarised. Furthermore, recommendations for the improvement of the application tool are given and an outlook is given.

2. Preliminaries

This chapter introduces the basics needed to understand the thesis. First, general terms about machine learning and explainable artificial intelligence are defined. Afterwards, decision trees and the techniques based on them are introduced. These constructs are mainly used in the modelling part. Additionally, techniques for tuning decision trees, feature selection and evaluation are introduced.

2.1. Artificial Intelligence and Machine Learning

Artificial intelligence is an increasingly important field of research in computer science that deals with the automation of intelligent behaviour and machine learning. In machine learning we try to find similarities and patterns in the available data. Machine learning is divided into the areas of supervised learning, unsupervised learning and reinforcement learning. In supervised learning, learning takes place from a data set for which the results, i.e. the output to be learned, are already known. This includes regressions and classification problems. In unsupervised learning, however, no target values are known which could be used for learning. In particular, this includes clustering algorithms. Typical problems in machine learning are the recognition of handwritten numbers or spam detection [8, 13, 78].

2.2. Explainable AI

Explainable AI (XAI) includes machine learning models that enable people to understand the decisions that are made. In general, an XAI tool is designed to ask how a certain outcome was achieved and why no other outcome was achieved. It must be clear which parameters influence the output most. Also, it is important to know in which cases the result is trustworthy and in which not, and what can be done to correct a possible error. These are all questions that are answered by an XAI model. Figure 2.1 illustrates the difference between a normal machine learning tool and an XAI tool [15, 40, 73].

2.3. Classification

Classification is an area from machine learning and belongs to the supervised learning methods. It is about the assignment of a new observation to a class. This is done by an algorithm which has been trained on data for which the classes are already known. A classification algorithm detects patterns in the data during this training, on the basis of which it tries to make assignments for observations with unknown classes. Mathematically a classifier can be described as function f, which receives an observation x as an input and outputs the corresponding class y, i.e., $f(x) = y$. A

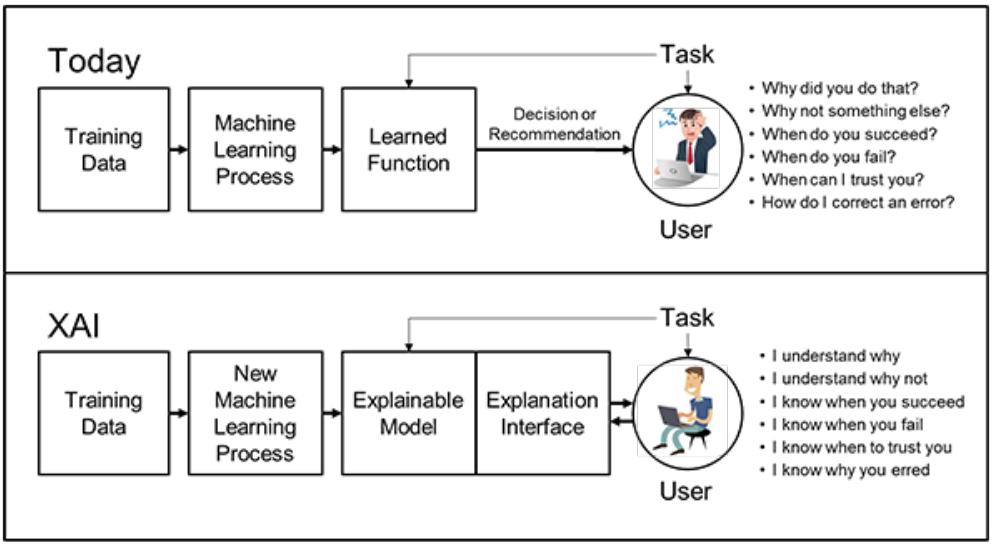


Figure 2.1.: Concept of XAI. The states with and without explanatory components for AI technologies are presented. Reprinted from [15].

typical classification problem is spam email filtering in which we try to classify new emails into the categories spam or not spam [13, 46, 78].

2.4. Decision Trees

Decision trees belong to the supervised learning methods and are used for classification and regression [66]. In this project, however, decision trees are only used as classifiers, since applicants are either accepted or rejected. Decision trees are based on the idea of extracting simple rules inferred from a data set and its features to predict a target variable. These rules are arranged hierarchically in a tree structure and are the internal nodes of the tree. The leaf nodes of the tree stand for different classes, to which we want to assign new observations. A basic decision tree algorithm is listed in algorithm 1. The algorithm initially receives the training data. Based on the training data, it selects the best attribute to divide the data set into subsets that have a higher purity with respect to their labels. For these subsets, the above procedure is repeated until the subsets are pure, no attributes are left or another condition, such as maximum depth, is met. There are different methods for the decision process regarding the attribute. For example, information gain and entropy can be used to select the best attribute [66, 77]. Figure 2.2 shows a decision tree. It shows an example of how students are classified in terms of acceptance or rejection at a university. Students who have previously studied at Oxford are always accepted, whereas students from Aachen need a very good grade point average to be accepted.

There are various algorithms for generating such a decision tree. Some of the best known

Algorithm 1 Basic Decision Tree Algorithm (ID3). Reprinted and adapted from [47, 77].

```
Split(Node,Data)
A = best attribute for splitting
Decision attribute for node = A
for values in A do
  Create new child node
  Split Data to child nodes
  for each child node/subset do
    if subset is pure then
      Stop
    else
      Split(child node,subset)
    end if
  end for
end for
```

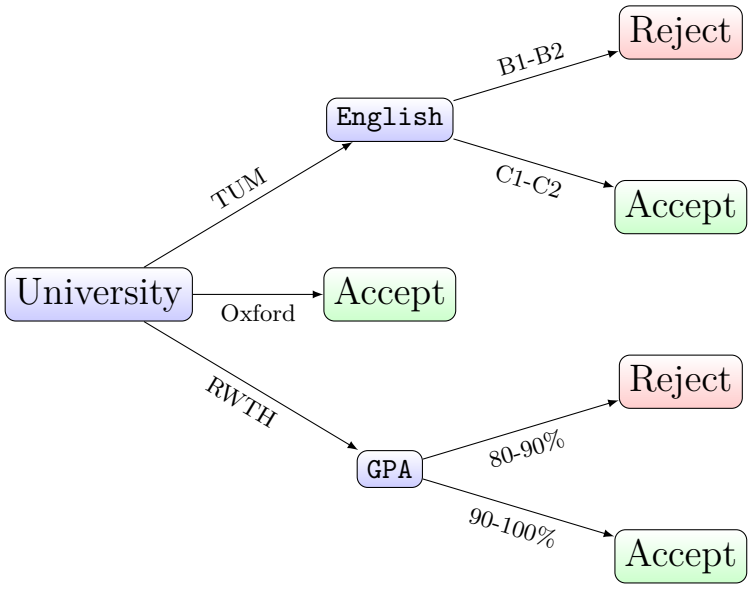


Figure 2.2.: Example of a decision tree. The decision tree visualises the decision process for the admission of students with different qualifications.

algorithms are called ID3, C4.5, C5.0 or CART. The ID3 algorithm is a basic greedy algorithm, which has been further developed to C4.5 or C5. The CART algorithm is similar to C4.5. Also, it generates only binary trees and uses information gain to determine the nodes. In this project decision trees are built by the CART algorithm. In general, decision trees have advantages but also disadvantages. One of the main advantages of decision trees is that they are very understandable and they are interpretable as long as they are not too deep. Furthermore, one can visualise decision trees. Another advantage is that decision trees require very little data preparation compared to other techniques, for example, the data does not need to be normalised. In addition, decision trees can handle both numeric and categorical attributes well. This is particularly important because both are used in the data set used. Also, the cost of using the tree is low. The costs are logarithmic in the number of data points used to train the decision tree. Moreover, decision trees generally perform very well. However, there are some disadvantages too. Decision trees can become too complex and not generalise the data, resulting in overfitting. Indeed, there are techniques like pruning to solve this problem. In addition, the decision trees can be unstable, since small variations in the data set can lead to very different trees. This problem can also be solved by using the trees in an ensemble. This technique can also help to fix the problem that a decision tree is greedy and therefore not necessarily optimal. Furthermore, decision trees cannot handle unbalanced data sets so well, i.e., when there are significantly more of one class than of another class in a data set. In such cases, the decision tree could be biased and decide in favour of the class that occurs most often. Balancing the data set by methods like undersampling or oversampling can mitigate this problem [28, 48, 57, 66].

When a decision tree is created, different features are used differently. In addition, some features help more to preserve pure subsets than others. It is therefore interesting to know which features are particularly important, as this also plays a role in the interpretation of the tree. One measure used for this is the feature importance. This feature importance is calculated as a normalized, total reduction of the used measure (such as gini impurity) which is brought by the feature [48, 71].

One way to optimise the performance of decision trees is pruning. If a decision tree is created based on data, it may be overfitted, as mentioned above. One method, for example, is cost-complexity pruning. The decision tree is first allowed to grow completely and is then pruned. Pruning is performed by recursively searching for nodes with a weak link. These are defined using a cost-complexity criterion. Finally, a balance is sought between the goodness of fit and the complexity of the decision tree [33, 69].

2.5. Ensemble Methods

Another possibility to get better results with decision trees is the use of so-called ensemble methods. In these methods several models are combined with each other. In general one distinguishes between ensemble methods that combine similar models, such as decision trees, and methods that combine different machine learning methods. When

combining models that are very similar, one can further distinguish between bagging and boosting [79].

2.5.1. Bagging

In bagging or bootstrap aggregation, different versions of a predictor are combined to an aggregated predictor. If this new predictor is to classify an observation, all the predictors which it consists of classify the observation. The predictor then returns the class that was classified most frequently. The versions of a model are created by training the models on bootstrap samples of the data set. Bagging can reduce the variance of an estimated prediction function and is very useful for decision trees. Furthermore, the accuracy is often improved [10, 79].

Random Forests

Random forests belong to the ensemble learning techniques. Random forests are a combination of multiple decision trees which are generated on bootstrap samples of the data set. In principle, for each tree a sample of the data and a subset of attributes are randomly selected. The individual decision trees are created based on this data and are not pruned. To classify a new datapoint, each decision tree predicts the class. The class that occurs most frequently is then returned. The class of the object to be classified is therefore determined with a majority vote. In general, random forests are a powerful machine learning technique, which have a better performance than decision trees. Random forests are much more stable than decision trees. Adding a new data point to the data set may affect individual trees, but not the whole set. Also, overfitting is avoided by the combination of a large number of different decision trees. A disadvantage compared to decision trees is that random forests with an increasing number of decision trees have a higher complexity and it takes longer to train the classifier [9, 34]. The random forest pseudocode is provided in algorithm 2.

Algorithm 2 Random Forests for Classification. Reprinted from [35].

1: **Random Forests**(Training Data)
2: **for** $b = 1$ to B **do**
3: Draw a bootstrap sample $\mathbf{Z}$ of size N from the training data.
4: Grow a random forest tree T_b to the bootstrapped data, by recursively repeating
5: the following steps for each terminal node of the tree, until the maximum node size n_{min} is reached.
6: i) Select m variables at random from the p variables.
7: ii) Pick the best variable/split-point among the m.
8: iii) Split the node into two daughter nodes.
9: **end for**
10: Output the ensemble of trees $\{T_b\}_1^B$

2.5.2. Boosting

Boosting is one of the most important learning ideas in recent years and is based like bagging on the use of several models. However, there are significant differences between the two methods. Boosting is one of the most powerful methods to optimise weak learners and achieves a high performance. These weak classifiers often take into account only a few features of the objects and do not provide good predictions regarding the classification of observations. In boosting, these classifiers are now combined and each one is given a weight. These weights are optimised to take greater account of classifiers who can better contribute to the correct classification of observations. Classifiers who can contribute little to the classification will be given a very low weight. To classify an observation, all classifiers predict the class and these predictions are combined into a final class based on the weights. AdaBoost, one of the best known boosting methods, is presented in detail below [36].

Algorithm 3 AdaBoost Algorithm. Reprinted from [37].

1: **AdaBoost**(Training Data)
2: Initialise the observation weights $w_i = \frac{1}{N}, i = 1, 2, ..., N$
3: **for** $m = 1$ to M **do**
4: Fit a classifier $G_m(x)$ to the training data using weights w_i
5: Compute $err_m = \frac{\sum_{i=1}^{N} w_i \cdot I(y_i \neq G_m(x_i))}{\sum_{t=1}^{N} w_i}$
6: Compute $\alpha_m = log(\frac{1-err_m}{err_m})$
7: Set $w_i \leftarrow w_i \cdot exp[\alpha_m \cdot I(y_i \neq G_m(x_i))], i = 1, 2, ...N$
8: **end for**
9: **return** $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

AdaBoost

A well-known boosting algorithm is AdaBoost which is the abbreviation for adaptive boosting. This works as follows: Suppose we have a weak classifier - a classifier that is only slightly better than random. Now this classifier sequential is applied to slightly modified versions of the existing data. A sequence of classifiers is generated. These classifiers classify a new data point and with a weighted majority vote of all predictions the corresponding class is returned. Classifiers which were more accurate in training, obtain more influence. This is made possible by a factor α, which is multiplied by the predicted class. If this factor is higher, the influence on the average weighted class is obviously higher. There are also observation weights. These weights exist for the training data and are all initially worth $\frac{1}{N}$, where N is the size of the data set. These weights are adjusted in an iterative process. The models classify the data as described above. Training observations that have been incorrectly classified receive a greater observation weight, while those that have been correctly classified are reduced. Permanently wrongly classified features get more and more influence after some time. Each classifier focuses on the corresponding data to classify it correctly. Algorithm 3

specifies the pseudo code for AdaBoost. M is the number of iterations in the algorithm, I is the loss function and (x_i, y_i) are the training observations. In addition to AdaBoost, other boosting techniques have also become widespread [36].

Gradient Boosting

Another well-known boosting method is gradient boosting. As with the techniques presented above, several weak models, usually decision trees, are combined to form a strong model. It belongs to the ensemble learning methods and is based on additive modelling. In gradient boosting an objective function is defined, which consists of a training loss and regularisation part. This indicates how well the model is fitted to the training data. The regularisation part serves to manage the complexity of the model and to avoid overfitting. In an additive process, decision trees are added to the model to optimise the objective function. This creates a model that delivers very good results in practice [14, 38]. In this thesis gradient boosting is used to compare the performance of decision trees and random forests with a stronger model.

2.6. Feature Selection

The amount of data collected has increased considerably in recent years. These data sets often have a variety of attributes. In these high-dimensional data sets, however, not all attributes are equally important. Some are irrelevant and affect the performance of machine learning algorithms. Feature selection solves the problem of high-dimensional data by selecting the most important attributes, that is, those that provide the most information. Features that hardly contain any information, or are even misleading, are removed by feature selection. In addition, feature selection also protects against overfitting if there are relatively many features to few observations. Instead of working with the whole data set, only the most important attributes are used. In the following paragraphs different feature selection methods are introduced. Another way to reduce the dimensions of a data set is feature extraction. This possibility is not considered in this thesis, because it would mean the loss of attributes which would therefore have a significant negative impact on the explainability of the tool [31, 63].

2.6.1. Recursive Feature Elimination

Recursive feature elimination is one of the backward feature elimination methods. Here a classifier is trained on the training data. Afterwards the features are ranked according to their importance to successfully classify the data. The least important feature is removed and the process is repeated with the remaining features. If there is a large amount of features and data, it may make sense to remove several features at once. In these cases the subsets of the features are ranked and the subset that has the least importance is removed. In the end, the combination of features is selected from which the model has performed best. The pseudocode for recursive feature elimination is

provided in algorithm 4 [31]. In the algorithm n is the number of features of the data.

Algorithm 4 Recursive Feature Elimination [31]

RFE(Data)
for $i = n$ to 1 **do**
  Train the classifier on the given features
  Report the accuracy of the classifier
  Rank the features according to their importance
  Remove least important feature, continue with the new subset
end for
return best subset

2.6.2. Feature Selection with Chi-Square Tests

Another way to select relevant features is to use the chi-square test. In general the chi-square test is a statistical test to determine the independence between 2 events. In the feature selection features should be found which have a dependency to the target variable. Here the chi-square test is very practical. A high dependence is also associated with a high chi-suare value. With this method features with a high dependency on the target variable can be selected for a better learning process. However, it should be noted that the chi-square test makes less sense for categorical features [29].

2.7. Data Sampling Methods

Data sets in which one class occurs significantly more frequently than the other class are called unbalanced data sets. The class that occurs most often is called majority class, while the other class is called minority class. For classification problems, equally distributed classes in a data set are optimal for the learning process. However, it often happens that the data is unbalanced. Unbalanced data can have a considerable impact on the model, since the minority class could be considered as noisy data. In this case the machine learning model could discriminate against the minority class and in the worst case only predict the majority class. To solve this problem, the record must be balanced, i.e. the minority class must be given a higher weight. There are several methods to correct this imbalance between classes. The following section introduces methods that undersample or oversample the data in order to balance the data set [28, 76].

2.7.1. Undersampling

In undersampling, large parts of the majority class in the data set are deleted so that both classes occur equally often. One problem associated with this method is that information is lost when data is deleted. An advantage is that the data set is balanced

and the time required to train a model is reduced because the data set is smaller [28, 76].

Random Undersampling

In random undersampling observations are randomly removed from the majority class until the ratios of the two classes are equal. A drawback of this sampling method is that the data is removed randomly, and important information may be removed while redundant information remains in the data set. There are approaches such as condensed nearest neighbours that solve this problem [28, 76].

Condensed Nearest Neighbours

Instead of randomly selecting the observations from the majority class to be removed, instances for which several very similar observations exist in the data set can be removed first. Such an approach is implemented by the method condensed nearest neighbours. First, all observations belonging to the minority class are determined. Then, elements with similar properties from the majority class are added step by step to the training set, which initially consists only of the elements from the minority class. In general, this is a sampling method intended for k-nearest neighbours [28, 32].

2.7.2. Oversampling

In oversampling, new instances belonging to the minority class are added to the data set to balance it. In contrast to undersampling, an advantage of oversampling is that no information is lost. The disadvantage is that no new information is added to the data set regarding the minority class. This can lead to an overfit of the model. Also, it takes longer to train the model [28, 76].

Random Oversampling

In random oversampling, observations are randomly selected from the minority class and added to the data set. This removes the imbalance between the classes. However, the model can overfit because no new information is added to the data set by this type of oversampling [28, 76].

Synthetic Minority Oversampling Technique

Another well-known oversampling method is the Synthetic Minority Oversampling Technique (SMOTE). In this method, synthetic data of the minority class is generated and added to the data set in order to balance it. These are generated using the clustering algorithm k-nearest neighbours. For a randomly selected observation from the minority class, the k-nearest neighbours are determined and a new observation is generated, which lies between the selected observation and one of its neighbours. In practice, SMOTE is a widely used method, which is usually better than random oversampling [28, 76].

2.8. Evaluation

In general, if a machine learning model is to be trained, the data set available for this is divided into training data and test data. The model is trained on the training data and evaluated on the previously unknown test data. Splitting the data into training and test data is done to ensure that the model actually performs well and not because it has seen the data before. Often a validation set is also created to have another instance to test the performance. This is especially useful when tuning parameters, as one might choose some that work well on the test data by chance. The test data should ensure that if the model performed well on the validation set, this is not purely random. If the model is too adapted to the training data, it performs poorly on the test data and we speak of overfitting. However, there are methods with which performance can be estimated with even greater certainty [39, 67].

2.8.1. Cross-Validation

In the procedure described above, the data is broken down into up to three data sets - training data, validation data and test data. However, this leads to the problem that considerably less data is available for the actual learning process, which in turn can have a negative effect on the performance of the model. This problem can be solved with the cross-validation procedure. Here the data is split into training data and test data, while the validation data is no longer needed. The training data is now divided into k equal subsets. The model is then trained on k-1 of these subsets and evaluated on one. This is repeated k times, so each subset was once test data and k-1 times part of the training data. Finally, the average performance over all so-called folds is considered. the test data can then be used for the final test. The schematic process of cross-validation for k=5 is shown in figure 2.3. However, there are other cross-validation methods. For example, it can be determined that each class occurs equally often in each subset [39, 67].

2.8.2. Confusion Matrix

A confusion matrix is a matrix that visualises the performance of a classifier. One axis is labelled with the actual classes and the other axis with the predicted classes. This results in four fields to which the observations are assigned corresponding to the classification [26].

- True positive: The actual and the predicted class are positive.
- False positive: The actual class is negative but positive was predicted.
- False negatives: The actual class is positive but negative was predicted.
- True negative: The actual and the predicted class are negative.

An example of a confusion matrix of a classifier to accept and reject applicants is given in figure 2.4 [26].

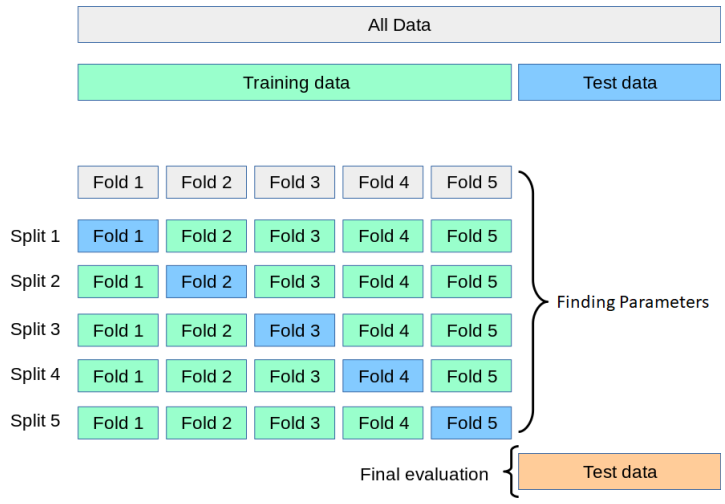


Figure 2.3.: Visualisation of cross-validation. Reprinted from [67].

		True Admission	
		Accept	Reject
Admission Classifier	Accept	True Positive	False Positive
	Reject	False Negative	True Negative

Figure 2.4.: Example Confusion Matrix

2.8.3. Accuracy, Precision, Recall and F1 Score

In the previous paragraphs we have often talked about the performance of a machine learning model. However, this has yet to be defined. To evaluate the performance of a machine learning model, there are different metrics. The most used metrics are accuracy, precision and recall. The accuracy is the fraction of correct predictions of a model, i.e., the proportion of true positives and true negatives. A high accuracy is therefore desirable. However, accuracy is not always a useful metric for evaluating the performance of a model. For highly unbalanced data sets, high accuracy does not say much about the model. A simple model that always predicts the class that occurs most often could also achieve high accuracy in this way. Therefore, in practice, precision-recall metric is often used to evaluate classifiers. Precision is the number of true positives over the number of true positives and false negatives, and the recall is the number of true positives over the number of true positives and false negatives. The precision indicates how exactly the model predicts the positively labelled class. One can easily see if many of the other classes are predicted incorrectly. In practice this can be relevant, for example when predicting spam in emails. If many emails that are not spam are classified as spam, important emails could be lost. Therefore, a high precision is desirable. The recall, on the other hand, indicates how many of the positively labelled ones have actually been predicted as positive. The recall is therefore particularly important if false negative values are to be avoided. Often we also want to have a balance between precision and recall. To achieve a balance between the two

values, the F1 score is used. The F1 score is the double product of precision and recall divided by the sum of precision and recall. The optimal F1 score is 1 and is also called harmonic mean. Also, the precision corresponds to the positive predicted value (PPV) and the recall to the true positive rate (TPR). The negative predicted value (NPV) and the true negative rate (TNR) are basically precision and recall for the negative class. These metrics help us to analyse and evaluate the performance of the models. Definition 2.8.1 gives the formulas of the metrics introduced above [26, 75, 70].

Definition 2.8.1. Accuracy, Precision and Recall: Let TP, FP, TN and FN be the terms for true positive, false positive, true negative and false negative.

i) $Accuracy := \frac{TP+FP}{TP+TN+FP+FN}$
ii) $Precision\ (PPV) := \frac{TP}{TP+FP}$
iii) $Recall\ (TPR) := \frac{TP}{TP+FN}$
iv) $NPV := \frac{TN}{TN+FN}$
v) $TNR := \frac{TN}{TN+FP}$
vi) $F1 := 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

[26]

2.8.4. Wilson Score

In the previous section, various machine learning models based on decision trees were introduced. Metrics to evaluate their performance were also introduced. However, it would also be interesting to know the confidence intervals for these. A possible confidence interval can be given by the so-called Wilson score interval. This assumes a binomial distribution. Thus, only two possible outcomes are expected, such as a coin toss or correct classification of a student [53, 55, 84].

Definition 2.8.2. Wilson Score: The interval below is called Wilson score interval. Here, p is the quotient of the number of positive observations that occurred and the total number of observations, which are called n. $z_{\alpha/2}$ describes the quantile of the standard normal distribution. Also, the sum is called upper (w_-) bound and the difference lower bound (w_+). If p is equals to 0 then $w_- = 0$ and if p is equals to 1 then $w_+ = 1$.

$$w_+ = (p + \frac{z^2_{\alpha/2}}{2n} + z_{\alpha/2}\sqrt{[p(1-p) + z^2_{\alpha/2}/4n]/n})/(1 + z^2_{\alpha/2}/n)$$

$$w_- = (p + \frac{z^2_{\alpha/2}}{2n} - z_{\alpha/2}\sqrt{[p(1-p) + z^2_{\alpha/2}/4n]/n})/(1 + z^2_{\alpha/2}/n)$$

For example, the Wilson lower bound score and the upper bound score can be used to specify a confidence interval that indicates accuracy with 95% probability. Thus, the Wilson score provides further information to evaluate the performance and reliability of our model [53, 55, 84].

2.9. Explainability Methods

In this section the Shapley values are introduced. These allow a better explainability for complex models.

2.9.1. Shapley Values

Not all models are as easy to interpret as a decision tree. Bagging and boosting methods are much more difficult to interpret. The possibility to calculate the feature importances of the different models has already been mentioned. However, there are ways to better analyse the importance of the features, for example using the Shapley values, a well-known concept in game theory [23]. In game theory Shapley values are used to solve distribution problems that result from the cooperation of unequal players and additional profits. This problem can be transferred to the importance of a variable as follows. Individual variables, such as the GPA or the nationality of an applicant, can have a certain impact on the prediction of the model. However, several features combined can have a greater influence on the prediction than if the influence of the features is considered individually. The Shapley values assign the features their actual effect on a prediction [6, 23]. Formally, the problem can be described as follows. Given is a model $f()$ and p ordered variables $\{1, ..., p\}$, which are called permutation J. In the following formulas, we denote the set of variables which are placed before the j-th variable in J as $\pi(J, j)$. Furthermore x_* is a concrete instance of interest. The Shapley value is defined as follows:

$$\varphi(x_*, j) = \frac{1}{p!} \sum_{J} \Delta^{j|\pi(J,j)}(x_*),$$

where $\Delta^{j|\pi(J,j)}(x_*)$ is the variable importance and $p!$ all possible permutations, i.e. orderings, of the explanatory variables. The importance of a variable can depend on the order of the previously considered features. The variables are considered one after the other to determine their impact on the final prediction. Therefore it makes sense to consider all permutations. Basically, $\varphi(x_*, j)$ is the average of the variable-importance across all possible variable orderings. The Shapley values preserve some important features. For example, local accuracy is guaranteed. This means that the sum of the Shapley values is equal to the model prediction. Another important property is consistency, which means that if one feature is more important in one model than another, the importance is higher regardless of the other features present. In addition, variables that do not contribute to the prediction are assigned Shapley value 0. Shapley values belong to the model-agnostic methods and are useful to determine the influence of different features of a prediction [4, 5, 6, 23, 58]. Shapley values have the advantage of fairly dividing the contribution of features to the prediction. In scikit-learn, feature importances are based on the fraction of samples a feature contributes to and the decrease in impurity of the splits. For example, features with many different values are considered more important than those with less different values. Furthermore, the feature importances in scikit-learn are based on the training data. The feature

importances do not have to apply to unknown data. This makes the Shapley values more interesting for determining feature importance [68].

3. Data Acquisition and Understanding

This section briefly introduces the data set we are working with. This makes it clear which problems with the data are to be solved in the further course. Furthermore, data cleaning is performed and the data is analysed. For the analysis of the data the python packages in table A.1 were used.

3.1. Data Acquisition and Data Set

The available applicant data was collected by Saarland University based on a web-based information system. The data contains information from the application form which students have to fill in to apply for the master programme in computer science at Saarland University. A section of this application form is shown in figure 3.1. In addition, some information about the performance of the students who finally studied at Saarland University is available. In total, the data set contains 11853 applications and 25 different features. The features are listed in table 3.1. Also new feature names are introduced and a short description is provided. The features that were recorded about the performance at Saarland University are marked with the prefix UDS. Overall, there are 16 features from the application form and 9 features which are recorded by Saarland University. The existing data has been made as anonymous as possible in order not to violate data protection regulations. Therefore, some features have been coded and the names removed, which make it impossible to identify an applicant. In addition, the age of applicants is given in age brackets, so the exact age at the time of application is unknown. This is a further anonymisation.

The following features are encoded:

- GENDER
- MARITAL _STATUS
- STUDY _PROG _LAST _DEG_1
- STUDY _PROG _LAST _DEG_2
- NATIONALITY

Figure 3.2 shows the completeness of the data. Black means that the data are available, white means that the corresponding data are not available. It is noticeable that no student has all the data. There are several reasons for this. On the one hand, many of the fields that provide information about the performers at Saarland University are not filled in, which suggests that most of them have not been accepted. However, there are also gaps in some columns which are listed in the application form. For example, some students do not indicate their GPA. However, this value is an essential part of the application process. The combinations of missing values in the data set are visualised in more detail in the figure 3.3. The order of the features in the figure is the same as

in table 3.1. For example, this shows that most applicants have specified everything except DEGREE and STUDY_PROG_LAST_DEG_2.

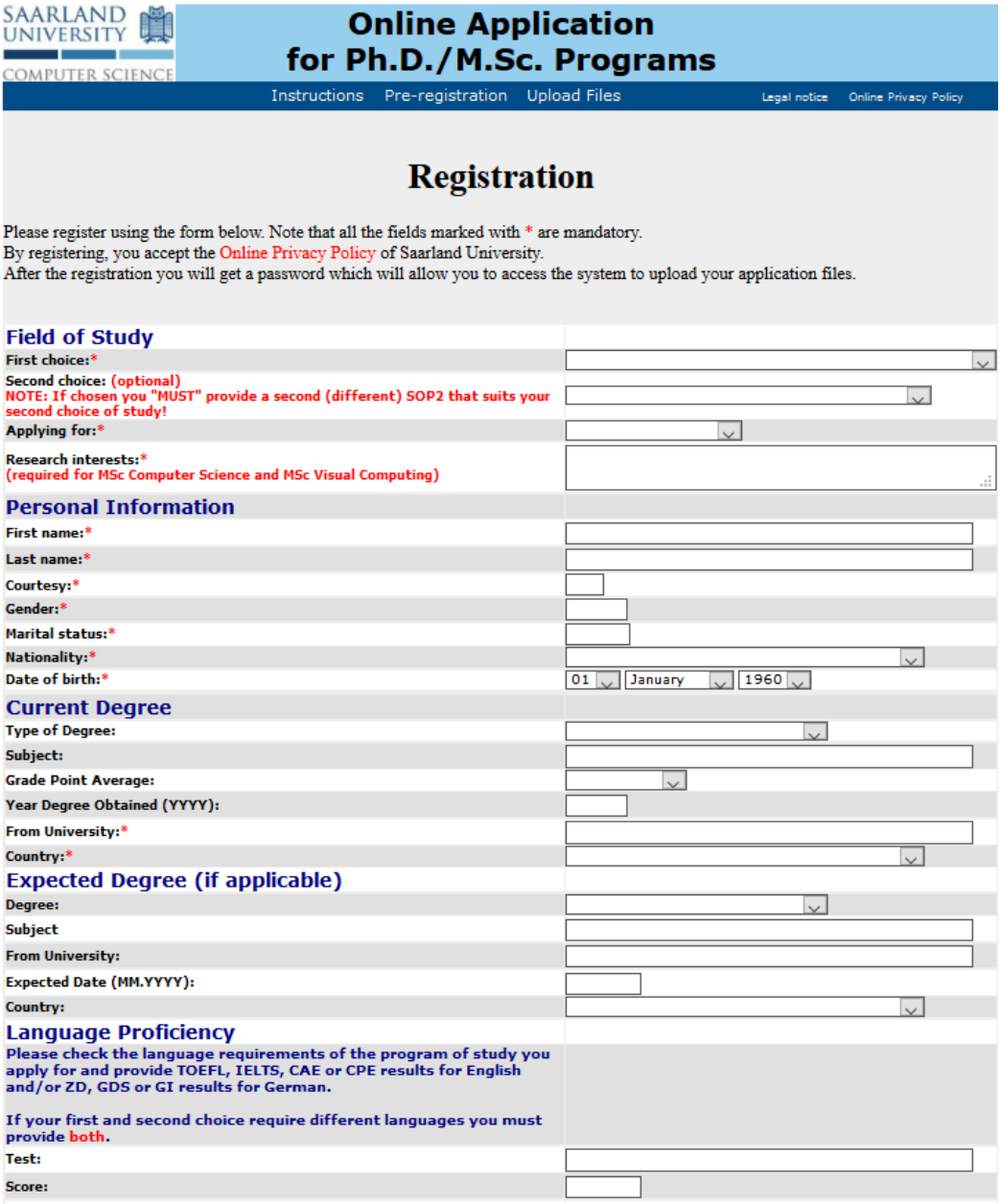
The next step is to check if the available data is clean and how to proceed with the null values.

Old Feature Name	**New Feature Name**	**Short Description**
Age	AGE	Age of the applicant
Gender	GENDER	Gender of the applicant
MaritalStatus	MARITAL_STATUS	Marital status of the applicant
Nat_coded	NATIONALITY	Nationality of the applicant
Uni_coded	INST_LAST_DEG_1	Last university attended
Uni2_coded	INST_LAST_DEG_1	Other university attended
Degree	DEGREE	Degree, e.g. BSc
LastDegree	LAST_DEG	Last Degree, e.g. BSc
YearDegreeObtained	YEAR_OF_LAST_DEG	Year of graduation
Subject	STUDY_PROG_LAST_DEG_1	Study programme
Subject.1	STUDY_PROG_LAST_DEG_2	Minor or second study programme
GradePointAverage	GPA	Grade Point Average of DEGREE
Test	TEST	Language Test, e.g. TOEFL
Score	TEST_SCORE	Score in TEST
Choice1	STUDY_PROG_CHOICE_1	First choice study programme
Choice2	STUDY_PROG_CHOICE_2	Second choice study programme
OI	UDS_OVERALL_IMPRESSION	Impression of the candidates at UdS
Abgesagt	UDS_OFFER_ACCEPTED	Accepted offer to study at UdS
Note_Stamm	UDS_GPA_CORE	Average grade in core lectures
CP_Stamm	UDS_CP_CORE	Credit points in core lectures
Note	UDS_GPA	Actual average grade at UdS
CP	UDS_CP	Achieved credit points
vollend_Semester	UDS_SEM_COMPLETED	Number of semesters completed
Abschluss	UDS_GPA_FINAL	Final average grade at UdS
vorzeitig_Exmat	UDS_PRE_EXMA	Dropped the study programme

Table 3.1.: Original and new names of the features with short description.

3.2. Data Cleaning

One of the most important steps in a machine learning project is data cleaning, or data preparation. Data cleaning is so important because the success of learning depends enormously on the quality of the data. It is much easier to learn from good quality data than from poor quality data. In this step, the data is cleaned and prepared for analysis and modelling. This includes correcting erroneous values, finding outliers and defining a strategy for dealing with null values [54, 62, 80]. However, data cleaning is also one of the most time consuming steps in practical machine learning projects. Often up to 60% of the time in a data science project is spent on data cleaning [12]. After a first inspection and an initial analysis of the data, some problems were found in the data set. Some features require data cleaning. In this project no special preprocessing of the data is required, because decision trees are used. The following lists the corresponding


SAARLAND UNIVERSITY
COMPUTER SCIENCE

Online Application for Ph.D./M.Sc. Programs

Instructions Pre-registration Upload Files Legal notice Online Privacy Policy

Registration

Please register using the form below. Note that all the fields marked with * are mandatory.
By registering, you accept the Online Privacy Policy of Saarland University.
After the registration you will get a password which will allow you to access the system to upload your application files.

Field of Study
First choice:*
Second choice: (optional)
NOTE: If chosen you "MUST" provide a second (different) SOP2 that suits your second choice of study!
Applying for:*
Research interests:*
(required for MSc Computer Science and MSc Visual Computing)

Personal Information
First name:*
Last name:*
Courtesy:*
Gender:*
Marital status:*
Nationality:*
Date of birth:* 01 January 1960

Current Degree
Type of Degree:
Subject:
Grade Point Average:
Year Degree Obtained (YYYY):
From University:*
Country:*

Expected Degree (if applicable)
Degree:
Subject
From University:
Expected Date (MM.YYYY):
Country:

Language Proficiency
Please check the language requirements of the program of study you apply for and provide TOEFL, IELTS, CAE or CPE results for English and/or ZD, GDS or GI results for German.

If your first and second choice require different languages you must provide both.
Test:
Score:

Figure 3.1.: Extract from the application form for masters in computer science at Saarland University. Reprinted and adapted from [82].

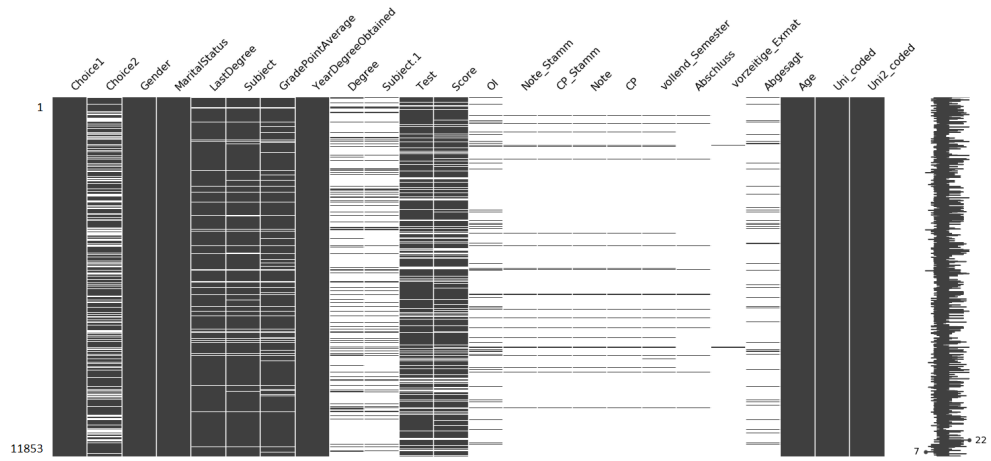


Figure 3.2.: Completeness of the data set provided by the study coordination.

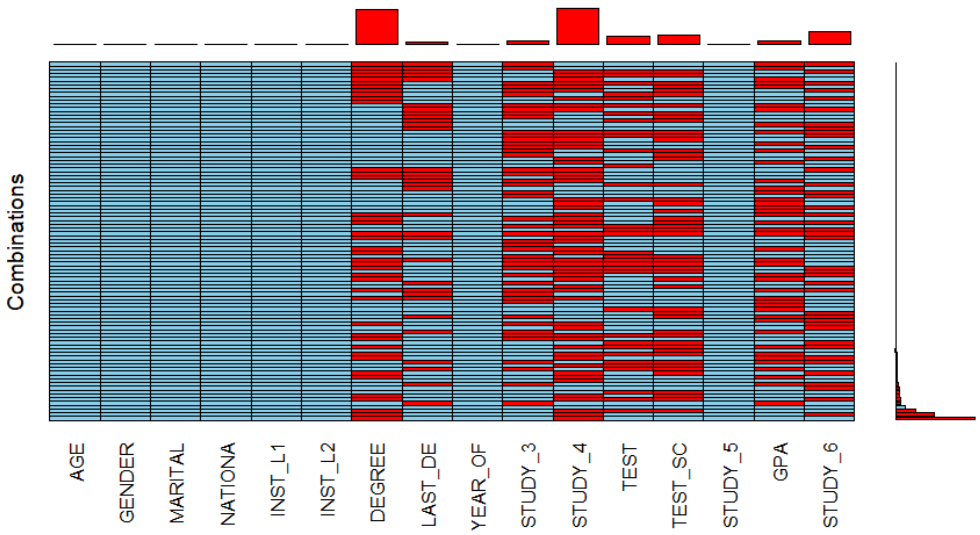


Figure 3.3.: Combinations of null values in the data set provided by the study coordination. Due to technical reasons feature names are abbreviated. This plot was created with the R package VIM [45].

Input University Name	**True University Name**
Anna University	Anna University
ANNA University	Anna University
Anna University, Chennai, India	Anna University
anna university	Anna University
anna University	Anna University
ANNA UNIVERSITY	Anna University
Aligarh Muslim University	Aligarh Muslim University
ALIGARH MUSLIM UNIVERSITY	Aligarh Muslim University
Aligarh Muslim University , Aligarh	Aligarh Muslim University
AIOU	Allama Iqbal Open University
AIOU ISLAMABAD PAKISTAN	Allama Iqbal Open University
GUC	German University in Cairo

Table 3.2.: Examples for university which were entered by applicants.

features and describes the data cleaning for them. In addition, the handling of null values is defined.

3.2.1. Universities

A feature that requires data cleaning is the universities. In the data set 4810 different university names are listed. In fact, there are significantly fewer universities, because applicants have to enter their university name manually and make spelling mistakes. In addition, some universities have names in several languages, which also appear in the data set. Often in these cases the English and local spelling can be found. For example, Saarland University is also a valid name for Universität des Saarlandes. Furthermore, some applicants only enter the abbreviation of their university in the application form (etc. UdS). There are also applicants who enter the faculty or similar information in addition to their university. It is extremely important to standardise the university names, i.e. to change the student's entry in the official name if it differs from the official name. If the data were not cleaned up, this would interfere with the learning process, as the algorithm might think that students are from different universities, although this is not the case. The data cleaning process for the universities should be automated to avoid manual cleaning in the future. Therefore two different approaches were tested.

String Comparison

One way to correct the spelling mistakes is to compare the entered university names with correct university names. The university that is most similar to the university which was entered should then replace it. To evaluate the similarity there are different metrics, which for example find the longest contiguous matching sub-sequence in a string or check to what extent similar sub-strings occur in the string. To be able to make such comparisons, we need a comprehensive database with all universities. There

are numerous databases on the internet, but most of them are not complete. Therefore a database was constructed for this project using Wikidata [43]. With SPARQL queries, all universities listed in Wikipedia were extracted and stored in a csv-file. This database was then used to perform the string comparisons mentioned above. To perform the string comparisons the SequenceMatcher from the python package difflib was used, which contains a modified "gestalt pattern matching" by Ratcliff and Obershelp [27]. This method worked relatively well for university entries which contained only minor errors and little information. However, some students have given abbreviations for the university or additional information such as a city, faculty or subject. In such cases, methods based on string comparison could almost never provide the correct university. The same applies to spelling mistakes. Overall, this was very time-consuming, but only partially successful. The number of incorrect entries could only be reduced slightly. In the next paragraph an approach is presented which solves the problems mentioned above as far as possible.

Microsoft Azure Cognitive Services Bing API and Wikipedia API

Since the data cleaning of the universities using the string comparisons did not work well, a completely different approach was tried. Microsoft Azure Cognitive Services provides users with various services. These include a Bing API that allows users to search for words, images, etc. [2]. In this approach, using this API, Bing is used to automatically search for student input. If the input does not contain the word university, it will be added to the input. This is to ensure that if an applicant has only entered an abbreviation as university, it will be found. In addition, the word Wikipedia is also added to the search term to get the Wikipedia page of the corresponding university. The links of the top five Wikipedia pages for the search term will be considered for further processing. All links are checked to see if they contain the word "uni". The first link that matches this is selected for further processing. If none of the links contain the word university, the first link returned will be continued. Using a Wikipedia API [30], the title of the Wikipedia entry to which the link refers is extracted. This title is written as the correct university name in the record. In addition, the corresponding link is added to the Wikipedia entry, which can provide the selection committee with a quick overview of a university. Overall, this method worked very well. With this procedure it is possible to assign the official name of the university to entries with spelling mistakes, in other languages, with additional information and abbreviations. However, there are also problems. For example, there are entries from applicants that contain far too much information, unknown abbreviations or spelling mistakes that make it impossible to assign the correct name. A similar problem exists if the university has no English or German Wikipedia page. With the problems listed above, sometimes a wrong university is assigned, which is sometimes much better known. In other cases nothing is assigned. Through other small experiments, such as adding the country, we tried to minimise these problems, but they had a negative effect on the final result. Overall, however, this approach is convincing, especially because it works much better than the one above. Table 3.2 shows some examples of the inputs and the corrected university name.

3.2.2. Study Programmes

In our database are 2340 distinct values for study programmes. Examples for the input of the study programmes are in the table 3.3. Many students made spelling mistakes or did not respect upper and lower case. As a consequence one has to correct them in order to work as well as possible with our data. Also, many students attended very specialised programmes like "Management and Processing of Informations". These programmes were matched with their nearest common study programme. Data cleaning was done manually with Excel since many inputs were not appropriate for an automatic solution. After data cleaning 124 programmes are left.

3.2.3. Language Test

One of the prerequisites for studying Computer Science at Saarland University is proof of English language skills at level C1. Students have to submit one of the common language certificates such as TOEFL or IELTS, which test reading, writing, speaking and listening comprehension skills and evaluate them with a score. There are other certificates which are also recognised. However, there are several problems with the available data concerning the language test.

Problems:

1. Spelling mistakes or too much information
2. Many different tests and therefore different scales
3. Invalid language certificates
4. NA values
5. Database not up-to-date

There are 946 different values for language tests in the database. An extract from the database is shown in Table 3.4. As with universities, one of the main problems is that applicants make spelling mistakes when entering or provide additional information. This requires data cleaning, as otherwise the algorithm of the 946 entries would interpret them as different tests, although there are significantly fewer, as most of them are just variations of common language tests. However, there are also a large number of entries that refer to a reference, experience or university certificate that is supposed to confirm English proficiency. Such certificates are not recognised. In addition, there are many students who do not have an English certificate or have written that they will submit a certificate later. Whether or not this has happened has never been updated in the database. All this makes it extremely difficult to work with the data. Therefore, it was decided not to consider the language certificates for learning.

Input Study Programme
Bachelor en informatique (professionnel)
Bachelor of Computer Science (hons)
Bachelor Of Technology(Computer Science)
Bachelors in Computer Science
C,C ++,English,Math,CN,SE,IT
Compter Science
computer
Computer Ccience
Computer Sciencce
Computer Science
COMPUTER SCIENCE
Computer Science (Informatik)
Computer Science A.I
COMPUTER SCIENCE OR RELATED SUBJECT
Computer Science, AI Specialization
Computer Science, BSc
Computer Science, Physics
Computer Science1
Computer Sciences
Computer sciense
Computer Secince
Conputer Science
European Computer Science
Fundamentals of Computer Science
Informatics
Informatik
MSc Computer Science
#38; MSc Bioinformatics
MSc informatique
MSc.Computer Science
Ph.D. in Computer Science

Table 3.3.: Examples for study programmes which were entered by applicants.

Input Language Certificate
Abitur
Deutsches Abitur
Englisch Grundkurs
Leistungskurs Englisch
CAE
Cambridge Advanced English
Certificate in Advanced English (CAE)
CEFR
APTIS
B1
Bogazici University Proficiency Exam
Cambridge
Bonafide College Certificate
Certficate of English Proficiency from my university
ENGLISH CIRTIFICATE
English cirtificate from university
English Proficiency Letter
English Speaking Country
English Spocken Language
ENGLISH THOUGHT
Mock IELTS
university
University Entrance Test
I can take letter from my university to show my English
IGCSE
CET
in Schottland studiert
CPE
None
i didnt have but all education i took is in english langue
GRE
Academic IELTS
I.E.L.T.S
IELTs
TOEFL(ibT)-IELTS
TOEFL
native
Not Required
PTE
I will submit IELTS score as soon as I get it.
Sorry , i don’t have any test but i can speak English clear

Table 3.4.: Examples for language certificates which were entered by applicants.

3.2.4. Replacement of NA Equivalents

In the data set were some minor issues which had to be removed. For example, some students had in UDS_GPA values like ' —' or '0'. These values are replaced by NA values. Also, values like 'v', ' ' and '?' were in the database, which were removed as well in order to obtain a clean data set.

3.2.5. Outliers

In the data set there were unrealistic values only for the points of the language certificates. This is due to the fact that the applicants had to enter their score there manually. There were no outliers for the other features. This is due to the fact that other numerical features like the GPA were selected from a drop down menu. The few unrealistic scores for the language certificates were not considered any further because the feature was removed.

3.2.6. Missing Values

Dealing with missing values is one of the major challenges in machine learning projects. In general there are different strategies to deal with them [19, 81].

1. Delete observations
2. Replace with the summary, e.g. mean or median
3. Random replace
4. Use a predictive model, e.g. k nearest neighbours or regression
5. Replace with a specific value, e.g. 0 for all missing values

The beginning of this provides a first overview of the data. There, figure 3.2 also visualises the completeness of the data. Obviously there are many features which have null values for various reasons. The features that are recorded at Saarland University obviously only have values if the applicants have studied there for at least one semester. Since these features are obviously not needed for learning, no changes are made to these features. Option 5 is implemented for the further procedure for the other features with missing data. If there is no second choice for a subject, this can also be a feature. Perhaps such students have thought about the choice of subject particularly well. In these cases, the missing value is simply replaced by a value that is coded later. This is done analogously for the subject, the second subject or minor, the degree, etc. It is also possible that accidental replacement of these values, or replacement of the values according to a distribution, could distort the data set. Similar problems arise if the GPA is not available. Therefore, this is also replaced by a fixed value. However, option 5 allows to test the other options during the learning process. In particular, deleting the missing values will be tried out in the modelling [19, 81].

3.2.7. Results

The data could be cleaned by the measures described above. In particular, the university names and courses of study could be given standardised names. As a result, there are significantly fewer different values for these features. Figure 3.4 shows the number of distinct values before and after data cleaning. The number of different values for universities was reduced by 53.93% and the number of different study programmes by 94.7%. This is a significant improvement, although it must be assumed that there are still errors in both features. Unfortunately, the language test and its score had to be removed since there are too many distinct tests and the database is not updated. Also, some random values (like '?') which were entered manually in the database were removed. Furthermore, no outliers were found that could negatively influence the learning process. A strategy for dealing with the missing values was also defined. Overall, a significantly better data quality was achieved, which enables the data to be analysed and used for machine learning.

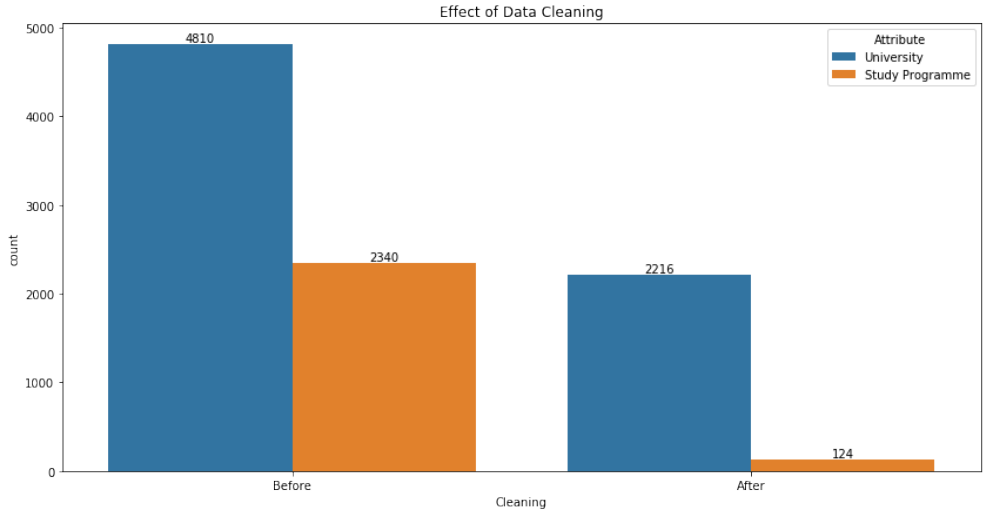


Figure 3.4.: Visualisation of the impact of data cleaning ($n = 11853$).

3.3. Data Exploration

In this section, the previously introduced data are further analysed. First statistics on the data set and individual features and relations to other features are discussed. This allows a good understanding and helps to optimise the modelling and interpret the results.

The data set contains 11853 applications from which 1192 applicants got an offer and 459 accepted it. Table 3.5 shows the applications broken down by gender and marital status. Also, the table provides exact numbers for offers, accepted offers, declined offers and rejections. 80% of the applicants have the gender 0 and 20% have the gender 1. About 10% of the applicants got an offer and 90% were rejected. Interestingly, only 8.84% of the applicants with gender 0 get an offer, but 14.83% of applicants with gender 1 get an offer. As result 70% of the applicants who got an offer have the gender 0 and 30% the gender 1. From the 10% who got an offer only 38.51% of the applicants accepted their offer. 37.81% of the applicants with gender 1 and 38.8% of applicants with gender 0 accepted their offer and are therefore equally likely to accept their offer. Another noticeable aspect is that married applicants have much lower chances to get an offer. Only 5.92% of married applicants get an offer. This number is especially low since 73.37% of the applicants with gender 0 and only 4.3% of married applicants with gender 0 were accepted while the 26.63% married applicants with gender 1 had with an offer rate of 10.37%. In conclusion, we can see that the admission process at Saarland University in computer science is very selective. Applicants with gender 0 and especially married applicants with gender 0 are accepted much less often at Saarland University than applicants with gender 1. Only 3.87% of the applicants finally study at Saarland University.

Another interesting feature is the age of the applicants. Table 3.6 shows the age distribution of the applicants. 77.32% of the applicants are between 22 and 27 years old.

Criteria	Applications	Offer	Offer Accepted	Offer Declined	Rejection
Overall	11853	1192	459	733	10661
Gender 0	9446	835	324	511	8611
Gender 1	2407	357	135	222	2050
Not Married	10839	1132	440	692	9707
Married	1014	60	19	41	954
Married, Gen. 0	744	32	10	22	722
Married, Gen. 1	270	28	9	19	242

Table 3.5.: Applicant statistics on gender and marital status.

Age	Applications	Offer	Offer Accepted	Offer Declined	Rejection
<20	12	0	0	0	12
20/21	532	101	44	57	431
22/23	3209	533	207	326	2676
24/25	3809	397	150	247	3412
26/27	2147	103	38	65	2044
28/29	972	34	9	25	948
30/31	524	14	4	10	510
32/33	287	5	3	2	282
34/35	163	1	1	0	162
36/37	79	1	0	1	78
38/39	50	1	1	0	49
>40	70	2	2	0	68

Table 3.6.: Applicant statistics on age.

Applicants younger than 20 and older than 35 barely exist. It is noticeable that the older applicants are extremely underrepresented in the group of applicants who got an offer. Only 2.05% of the applicants who are 30 years or older get an offer. In contrast to those applicants, younger applicants have much higher chances to get an offer. The 20 to 21 year old applicants have an acceptance rate of 18.98% at Saarland University, which is the highest value for all age groups.

Detailed statistics on the applicants are provided above. Table 3.7 provides a detailed statistic about the number of universities and nationalities from which the applicants are from. Students from 136 countries and 2116 universities applied at Saarland University. Applicants from 86 nations and 553 universities got an offer and finally applicants from 61 countries and 291 universities arrived at Saarland University. The accepted applicants are highly diverse since only 459 applicants accepted their offer and there is a high number of different universities and nationalities. On average every university occurs less than two times and every nation less than 5 times in the data set of accepted applicants. However, there is not a uniform distribution of universities and nationalities in the data set. Only 25 universities occur more than two times and only 5 universities 10 times ore more. University 1365 occurs 19 times which is the highest value and about 4.1% of all accepted applicants. 217 Universities are only one time in the data set of

Criteria	Applications	Offer	Offer Accepted	Offer Declined	Rejection
Nationalities	136	86	61	70	129
Universities	2116	553	291	384	1939

Table 3.7.: Number of universities and nationalities in the data set.

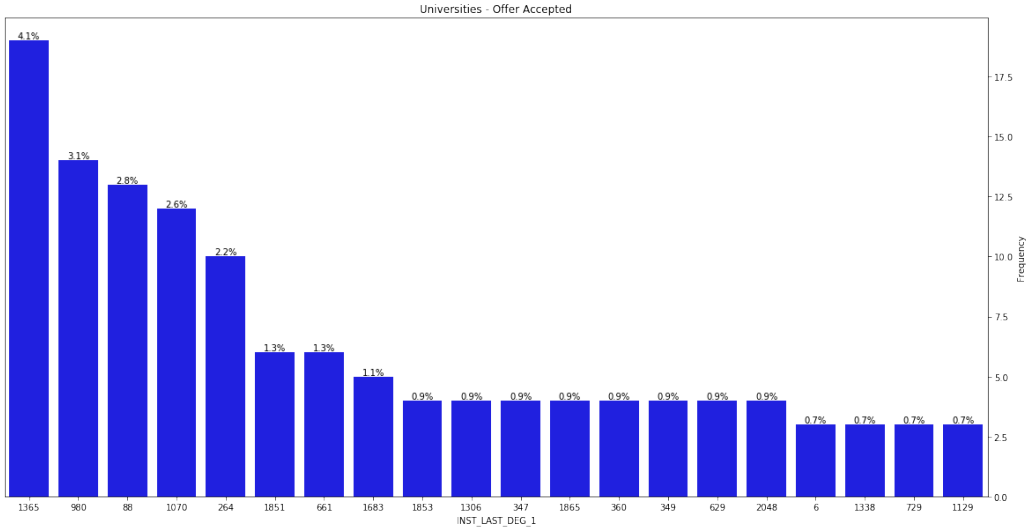


Figure 3.5.: Most common universities from which applicants who accepted an offer come from.

applicants who accepted their offer. Figure 3.5 shows the 20 universities which occur the most. The most common nationalities of applicants with an offer are depicted in figure 3.6. 25.8% are from country 91, 11.3% from country 102 and 8.7% from country 67 which makes them to the three most frequent countries. 45 of the 61 nationalities occur only three times or even less. According to the study coordination, one of the most important features for the selection process is the applicants grade point average (GPA). In general students should have achieved an average of at least 75% in order to obtain an offer. Table 3.8 provides a detailed statistic about students grades and their application details. Applicants with a very good performance have much higher chances to obtain an offer compared to students with lower grades. 31.81% of the applicants with a GPA of $91\% - 100\%$ obtain an offer which are more than double so many as the ones with an GPA of 81%. The chance of being admitted decreases continuously with the GPA. Applicants with a GPA of less than or equal to 70% have an acceptance rate which is less than 1%. Figure 3.7 shows the GPA distribution of the applicants which were rejected and those which were admitted. It is noticeable that applicants with a GPA of greater than 80% are the absolute majority of those who hold an offer. Applicants with a lower GPA barely obtain an offer. Besides the GPA, the last degree, especially the study programme is an important feature. Applicants have various backgrounds. The majority of applicants have a bachelor or master degree in computer science or a computer science-related course of study. This is followed by mathematics, engineering and physics. However, there are also applicants from study

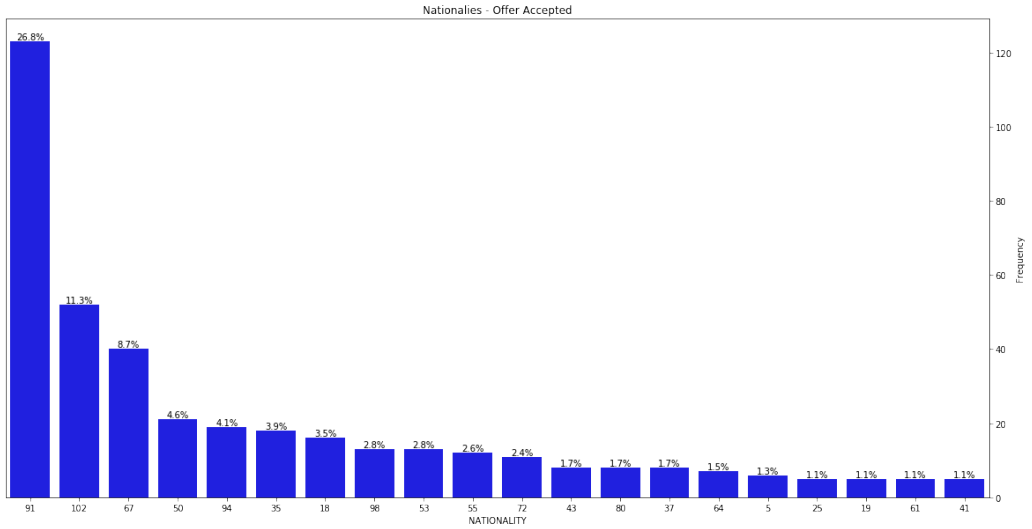


Figure 3.6.: Most common nationalities from which applicants who accepted an offer come from.

GPA	Applications	Offer	Offer Accepted	Offer Declined	Rejection
91%-100%	962	306	110	196	656
81%-90%	2699	423	165	258	2276
75%-80%	2853	130	53	77	2723
71%-75%	1579	29	13	16	1550
61%-70%	2257	12	4	8	2245
0%-60%	392	1	1	0	391
NA	1111	291	113	178	820

Table 3.8.: Admission statistics of all applicants grouped by the GPA of their last degree.

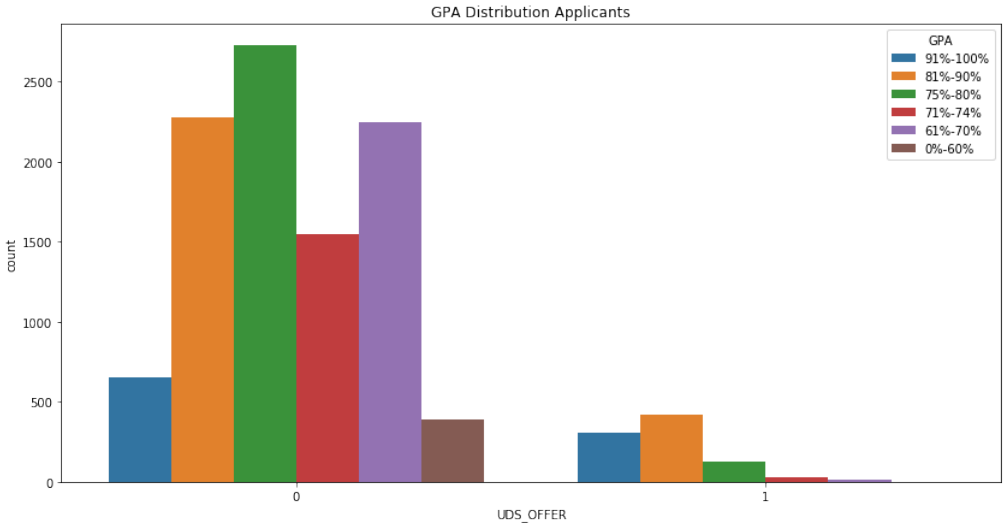


Figure 3.7.: GPA of applicants with and without offer.

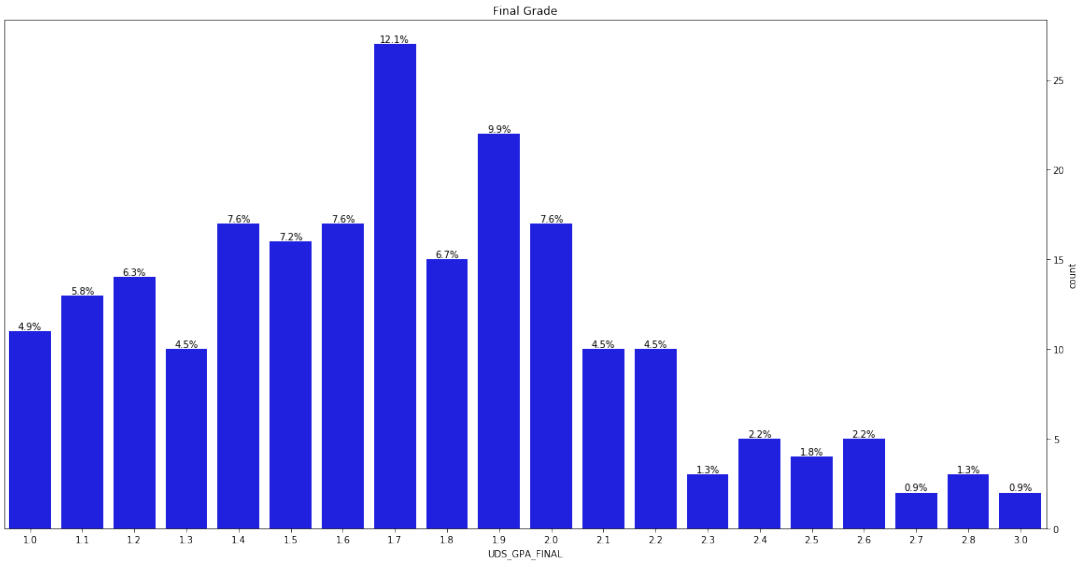


Figure 3.8.: Final grades of applicants who accepted their offer at Saarland University.

programmes such as business studies or humanities.
Above many admission statistics and applicants statistics are provided. But it is also interesting and important to see how applicants performed at Saarland University. Figure 3.8 shows the final grade distribution of those who got an offer and finished their degree. Most students graduated with an average better than 2.0 and only 7.1% had an average worse than 2.4. However, there are big differences regarding the time taken and the grades in core lectures. Also, 50 students were exmatriculated before graduation. At Saarland University about 30 credit points are scheduled per semester. Figure 3.9 shows the amount of credit points which the 390 students who completed at least two semesters achieved. Only 13.7% of the students were able to achieve at least 30 CP. Some of them achieved even more than 40 CP. 12.05% achieved $25 - 30$ CP per semester which means that 74.25% of the students achieved less than 25 CP. Also, 22.31% achieved less than the half of the recommended points per semester. Since the majority of students do not achieve 30 CP the study duration is much longer than the scheduled duration of four semesters. Figure 3.10 shows that 20.6% graduate on time and 48.9% graduate within a delay of one to two semesters. However, 30.4% of the students have a significant delay of three semesters or more. Some students even require up to eleven or twelve semesters to finish their master degree. An important part of a master degree at Saarland University are core lectures. Regular lectures provide students with a profound knowledge of the relevant topic and are theoretically extremely challenging. Figure 3.11 shows the GPA distribution in core lectures. Remarkably the core lecture grades are significantly worse than the final GPA. 11.1% of the students achieve a grade better than or equals to 1.5 and 11.1% have a grade worse than 1.5 but better than 2.0. 17.9% of the students have a GPA of 2.0 to 2.4, 24.5% a GPA of 2.5 to 2.9, 13.8% a GPA of 3.0 to 3.4 and 11.6% a GPA between 3.5 and 4.0. On average, 9.5% of the students failed their core lectures.

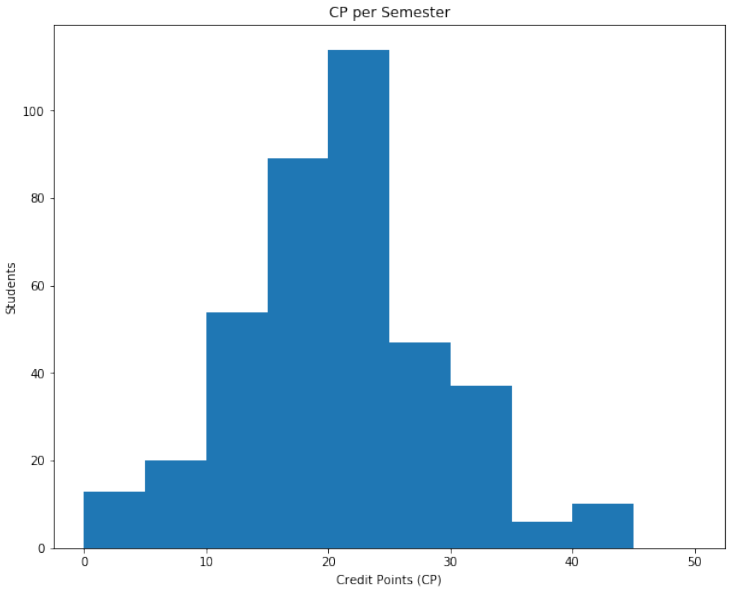


Figure 3.9.: Average credit points per semester of applicants who accepted their offer at Saarland University.

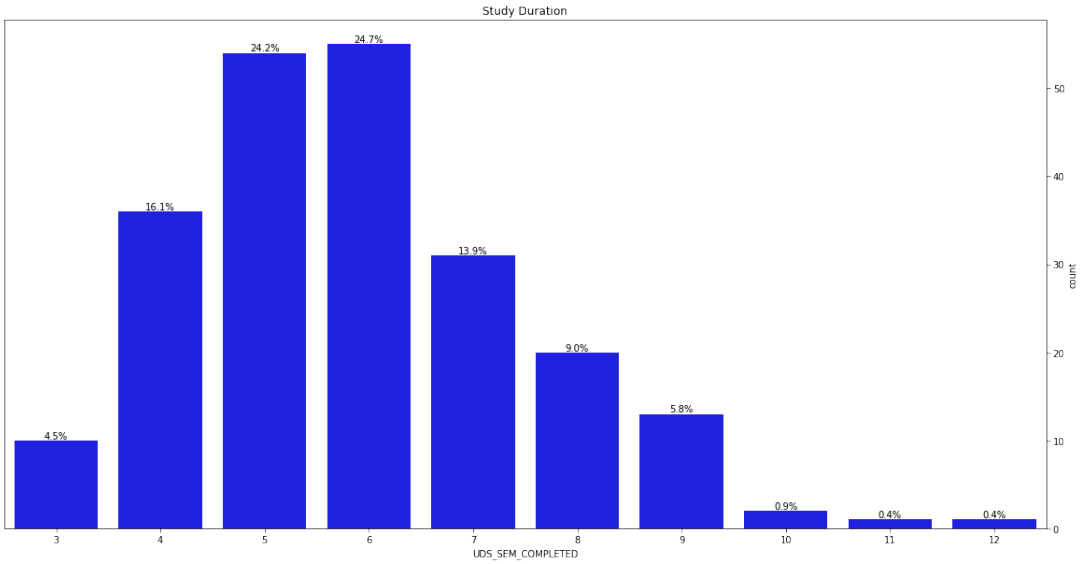


Figure 3.10.: Study duration of applicants who accepted their offer at Saarland University.

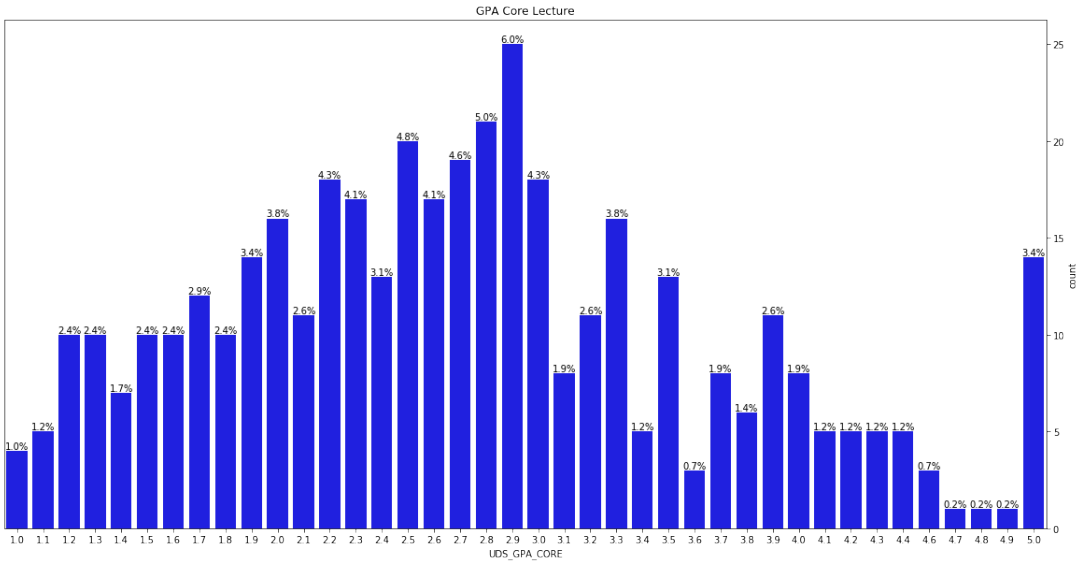


Figure 3.11.: GPA in core lectures of applicants who accepted their offer at Saarland University.

The final grades, the grades in core lectures and the number of semesters needed to obtain a degree are important to us to judge students performance and to label our data. This step is done and described in section 4.2. However, it would be interesting to see if there are correlations between features from the application form and the students performance. Figure 3.12 shows the achievements at Saarland University of the applicants grouped by GPA. Top results in core lectures and in the final GPA are mainly achieved by students with a high GPA in their prior degree. Students with a GPA of less than 75% barely achieve as high results as the other students. However, there are still students with a high GPA (over 91%) which do not perform well. Table 3.9 provides detailed statistics which show a similar trend. In general, the students who had a GPA of at least 91% achieve the best average final GPA at Saarland University. Also, they achieve the second best results in core lectures and get second most credit points. However, it is surprising that students with a GPA of $71\% - 74\%$ achieve the best grades in core lectures, have the most average credit points and do not have exmatriculations before graduation.

In this section the most important properties of the data set were prepared and visualised. It provides comprehensive statistics on the admission process and performance at Saarland University. In the next step, further features will be added to the data set, which should enable an optimal modelling.

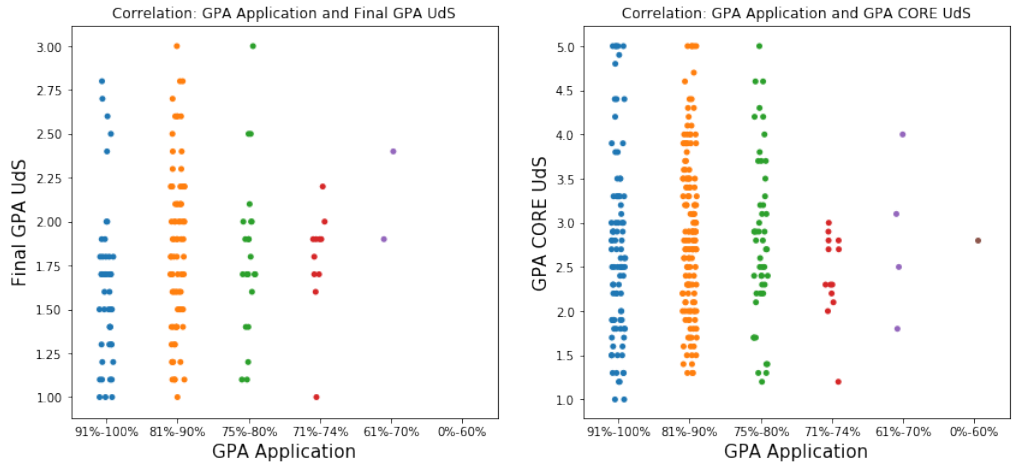


Figure 3.12.: Left: GPA application form versus final GPA UdS of applicants who accepted their offer at Saarland University. Right: GPA application form versus final GPA core lectures of applicants who accepted their offer at Saarland University.

GPA	UDS_GPA_CORE	UDS_GPA	UDS_GPA_FINAL	UDS_PRE_EXMA	UDS_AVG_CP
0%-60%	2.80	2.39	-	1.00	16.00
61%-70%	2.47	2.33	2.15	0.33	18.33
71%-75%	2.39	1.79	1.78	0.00	24.18
75%-80%	2.58	1.94	1.81	0.24	18.14
81%-90%	2.61	2.02	1.85	0.13	20.43
91%-100%	2.45	2.03	1.63	0.26	21.43

Table 3.9.: Detailed statistics about the performance of applicants who accepted their offer at Saarland University. Applicants are grouped by their GPA of their previous degree.

4. Modelling

The modelling of the data is an elementary step of every machine learning project. In the next two sections new features are created and the data is labelled. The data is prepared in such a way that machine learning models can be trained with this [54]. In the following two sections the packages in table A.2 were used.

4.1. Feature Engineering

The process of creating suitable features based on the available data and its context is called feature engineering. Feature engineering is a very important step in a machine learning project, as the performance of the model depends largely on the input data [54]. In this project it is important to gain additional information which can help to optimise the admission to Saarland University. In the following sections some new features are introduced for this purpose.

4.1.1. University Rankings

A machine learning model does not know how good universities really are. Until now universities are a categorical feature and the judgement would be based on experiences with accepted students. This leads to several problems. For example, it is not possible to judge the quality of a university from which a student has never been accepted and one could discriminate universities based on one bad experience. By adding rankings to the data set these problems can be partly solved. An algorithm could determine a minimum ranking which a university should have based on past experiences. Thus it could judge if an unknown, ranked university is good enough. In the following sections the added rankings are described.

Worldwide Rankings

Two of the major worldwide rankings are the THE World University Rankings [18] and the QS World University Rankings [50]. Both rank universities from all countries in computer science, based on citations, academic and employer reputation, etc. The THE World University Rankings includes 749 universities and QS World University Rankings ranks 500 universities. The QS ranking could be downloaded as a csv-file. THE ranking was extracted as a csv-file from the website using Octoparse [42]. However, there were similar problems with the university names as with the application form. The rankings sometimes used other names than Wikipedia, and also Octoparse sometimes made small mistakes. Therefore the university names were adjusted with the same tool as in Data Cleaning. To guarantee maximum correctness, all university names were checked and adjusted manually once again. Both rankings were inserted automatically in the data set

via a join on the university name. All applicants who have attended a ranked university have a rank as an additional feature now. From lower positions in the ranking only a range is given. For example, Saarland University is listed among the top $101 - 150$ universities in the QS ranking. In order for the decision tree to work better with the ranks, the average value of a range was assigned to the corresponding universities. Saarland University was thus assigned rank 125. The rankings should provide an initial orientation. Two rankings have been added because universities are ranked differently in each. For example, the QS ranking places Cornell University in 25th place worldwide, while THE ranking places it in 14th place. These differences become even greater in the lower rankings. Another problem is that most of the universities are not listed because there are of course significantly more than 500 and 750 universities worldwide. In particular, universities from Africa and Asia are not yet as well established. According to the study coordination, however, numerous students apply, especially from Asia. A local ranking for Asia and Africa would help to remedy this problem somewhat. However, it was possible to assign 1327 applicants a rank from the QS ranking and 1442 applicants a rank from the THE ranking which are 11.2% respectively 12.17% of all applicants. A more detailed analysis of the available data also shows that students who have studied at a university listed in the rankings are more likely to obtain an admission. At Saarland University, 1192 applicants received an offer. Of these, 233 applicants attended a university listed in the QS ranking and 209 applicants attended a university listed in THE ranking. That is 19.55% and 17.53% of all applicants who have received an offer. Furthermore, the universities of applicants who receive an offer are ranked higher than the universities of the other applicants. Universities of applicants with an offer and QS ranking are on average at position 316, while the average for all universities in the data with QS ranking is 341. If only the applicants without offers are considered, universities are ranked on average in 346th place. The same phenomenon can also be observed for applicants who have studied at a university listed in THE ranking. On average, universities in the data are ranked in 429th place, universities of an applicant with an offer in 386th place and universities of applicants without an offer are ranked in 436th place. This shows that the ranking, i.e. the reputation of the university, already plays a role, at least in the admission process. The question now is whether these students actually perform as well as those who did not attend a ranked university, or whether there is a correlation between rankings and students' performance at Saarland University. Basically, it can be said that the students who were at a university in the QS rankings have on average better grades than the other students. Students who studied at a university with a QS rank graduate with 1.62 on average. Also, their average in core lectures is 2.64. Students who do not belong to this group have a final grade of 1.74 and an average grade of 2.74 which means that they are less good. Students who were at a university with a THE rank before have a final grade of 1.74 and an average grade of 2.63 in core lectures while other students have a final grade of 1.72 and an average grade of 2.74 in core lectures. This means that students who have studied at THE ranked university do slightly worse. This is probably due to the thesis, since the core lectures and the current average of the enrolled students is higher than the others. In addition, there is a slight correlation for both rankings between the placement in the ranking and the achieved grade in core lectures. Both

rankings are slightly negatively correlated with the number of CP per semester. In other words, students with a higher ranking tend to achieve more CP per semester. All in all, it can be said that the inclusion of rankings in the data set makes sense and that an expansion of the data set with local rankings can bring further opportunities for optimal decisions.

Regional Rankings

Asian and especially African universities are significantly underrepresented in the two worldwide rankings introduced above. However, many of the applicants at Saarland University come from Asia and some from Africa. Missing rankings in those regions leads to uncertainty about the university quality of many applicants. Therefore regional THE rankings for Asia and Africa were added to rank as many students as possible [16, 17]. THE Asia Ranking includes 417 universities and THE Africa Ranking 56 universities. Unfortunately, these are not in specifically for computer science, but may offer a first orientation. Altogether 1779 applicants have applied from a university (INST_LAST_DEG_1) which is listed in the Asia ranking. On average the university of the applicants was in place 228. Applicants who have received an offer come from a university which is on average in place 195 while the other applicants are from universities which are on average in place 231. A similar phenomenon can be observed for students attending a university listed in the Africa ranking. Students who have been accepted have attended a university ranked 23rd, while the others and the average have attended a university ranked 26th. It can also be observed that students who have attended universities listed in the two rankings have lower admission rates than the other students. This also makes sense, as the quality in the breadth of universities is not as good as in western leading industrial countries. This is particularly reflected in the international rankings.

4.1.2. IT- and STEM-related Study Programmes

The data set contains a large number of study programmes. These have already been reduced to significantly fewer study programmes. The data set also includes a number of non-subject related study programmes, which have little to do with computer science. For this reason, the feature "Computer Science-Related" was added, which indicates whether the student has studied something in the field of computer science or not in one of his degrees. Also, students with computer science as a minor subject are positively labelled. The same was done for STEM-related programmes. This allows students studying a mathematical or technical subject to be placed in a group. Students who have 2 degrees will be positively labelled if one is related to computer science or STEM-related. The advantage is that applicants who have studied non-technical subjects can be rejected more easily by a decision tree. The decision tree does not have to be so deep. In addition, study programmes that have not previously been recorded in the database can be better assigned to a already known group. Figure 4.1 shows how many applicants studied an IT-related subject. 80.1% attended an IT-related programme, 17.9% studied something different and about 1.9% it is not possible to say if they did

or did not. Computer science related also includes subjects such as mathematics and physics with minors in computer science. For about 8.6% of the applicants it is not possible the say what they studied since they did not enter anything or they entered irrelevant data. Figure 4.1 also shows if applicants studied a STEM-related programme. 96.0% studied something STEM-related (biology and chemistry excluded) and only 2.0% studied something not STEM-related. Figure 4.2 shows whether the applicants studied something computer science related or STEM-related respectively and their offer status. One can see that most students who obtained an offer studied computer science. Also, the absolute majority of those who did not study computer science and obtained an offer studied something STEM-related. Some of the students that one does not know if they studied something computer science related got an admission too. Probably, most of them studied computer science and achieved good results.

4.1.3. Spelling of the Study Programme in the Application Form

Numerous spelling mistakes were found during the data cleaning. Spelling mistakes indicate that the student did not invest much effort in the application and therefore may not necessarily want to study at Saarland University. That is why a system is introduced which evaluates the spelling when entering the course of study. If students who have spelling mistakes in their applications' would be accepted less often, and if they would perform worse in their studies, then there would be another feature for a machine learning model. In the system, spelling mistakes are rated with points from 1 to 3.

1: The study programme was entered without any mistakes.

2: Upper and lower case letters were ignored.

3: A spelling error, an abbreviation or nothing was entered.

As applicants specify up to 2 study programmes, both were evaluated and the maximum value was assigned according to the points allocation. It is ensured that only applicants who did not specify a study programme in both fields were assigned 3 points for nothing entered. Basically, the lower the points the better the spelling. The allocation of the points was done manually. Table 4.1 shows some study programme inputs and their corresponding rating. Figures 4.3 and 4.4 provide information about the spelling abilities of the applicants and of those who were rejected, admitted and accepted their offer. 80.1% of the applicants were able to spell their study programme correctly, 13.3% ignored upper and lower case and 6.5% made spelling errors, used abbreviations or did not enter a study programme. It is clear that applicants who are accepted make serious spelling mistakes almost half as often as applicants who have been rejected. Also, successful applicants paid more attention to upper and lower case. While 14.1% of rejected applicants disregard rules on upper and lower case, only 6.6% of accepted applicants have done so. The numbers of applicants who accepted their offer are very similar to those who were admitted. It is therefore clear that there is a correlation between spelling and admission, so this feature is a useful addition to the others.

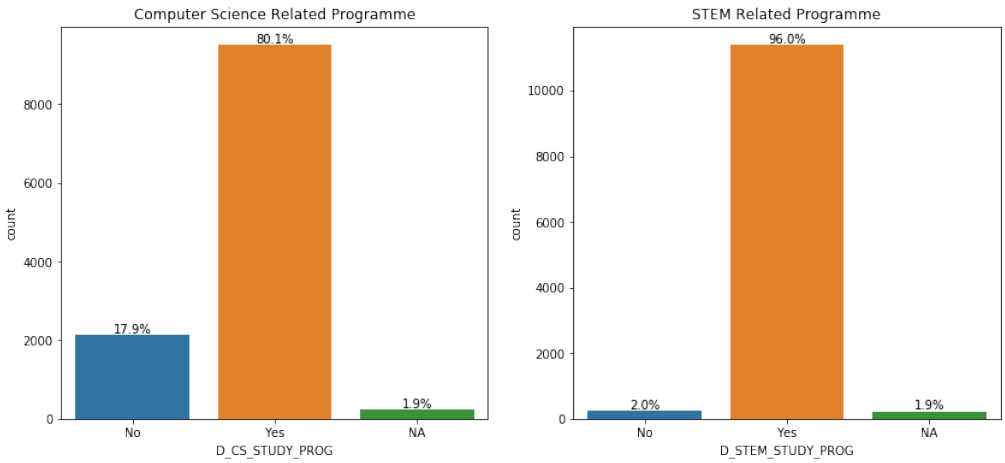


Figure 4.1.: Left: Applicants who studied an IT-related programme. Right: Applicants who studied a STEM-related programme.

Input University Study Programme	**Spelling**
Aerosapce Engineering	3
Applied Computer Science	1
Automation and management	2
biochemistry	2
CSE	3
Computer Systems	1
Liberal Arts and Sciences	1
MBA	3
Electrical Telecommunication Engineering	1
Electrical Telecommunication engineering	2

Table 4.1.: Study Programme Input Examples

4.1.4. Summary

Feature engineering has now been completed and various features have been added. Global and regional rankings have been added and the study programmes have been divided into the categories computer science related and STEM-related. In addition, the spelling of the students was evaluated based on their entries of their study programmes. Overall, a total of 11 new features were added. These are listed in table 4.2. These features provide additional information and can help to optimise the decision trees. In particular, they also help to better evaluate study programmes or universities for which no observations are available. In the next step the data is labelled.

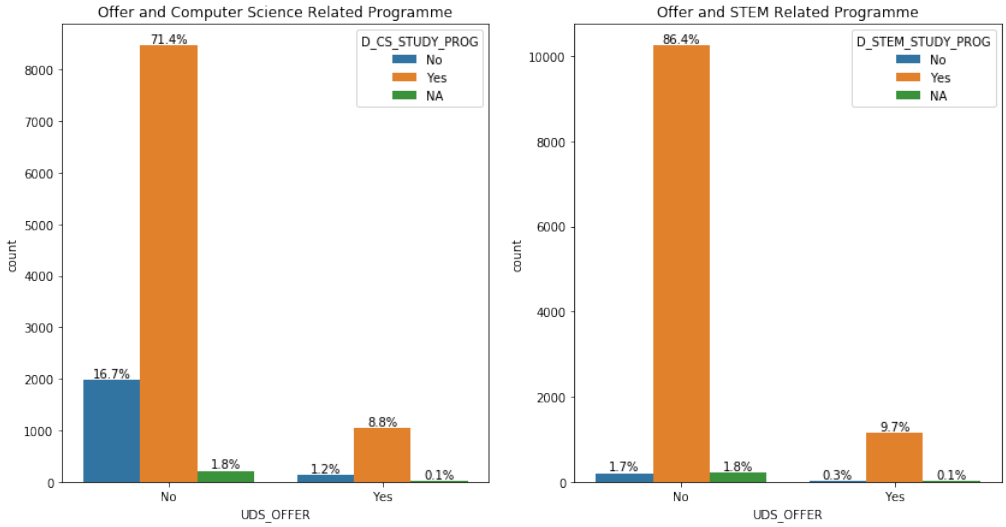


Figure 4.2.: In both figures the connection between academic background and admission is visualised. The figure on the left distinguishes between CS-related study programmes and other degrees. The figure on the right shows the same, but for STEM-related study programmes.

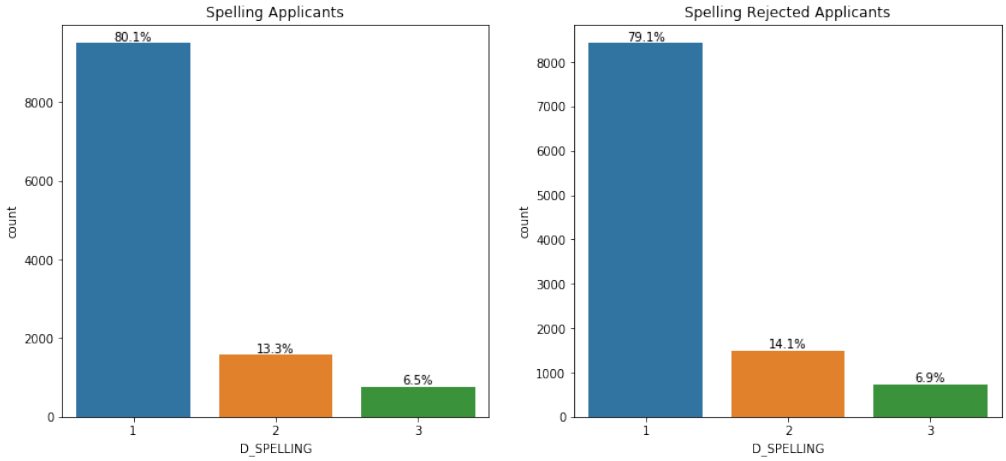


Figure 4.3.: Left: Quality of spelling of the applicants overall. Right: Quality of spelling of rejected applicants.

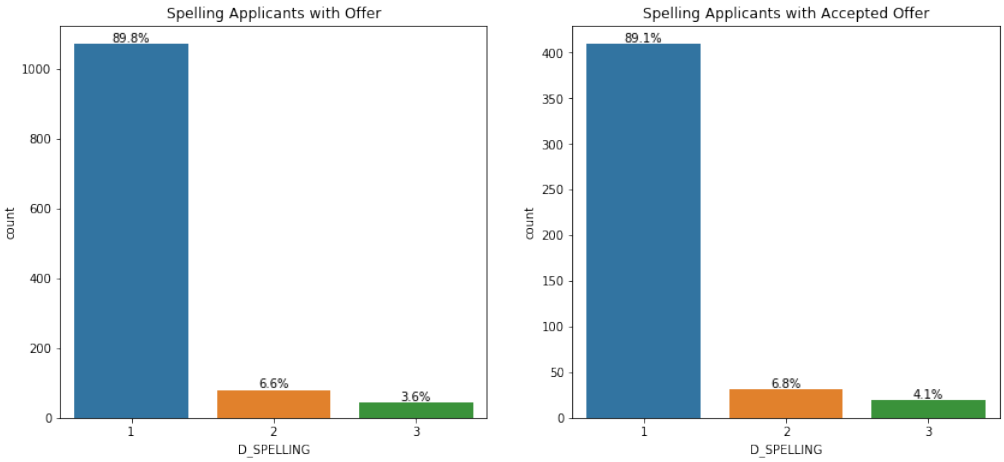


Figure 4.4.: Left: Quality of spelling of the applicants with an offer. Right: Quality of spelling of applicants with an accepted offer.

New Features
D_QS_RANK_INST_LAST_DEG_1
D_QS_RANK_INST_LAST_DEG_2
D_THE_RANK_INST_LAST_DEG_1
D_THE_RANK_INST_LAST_DEG_2
D_THEASIA_RANK_INST_LAST_DEG_1
D_THEASIA_RANK_INST_LAST_DEG_2
D_THEAFRICA_RANK_INST_LAST_DEG_1
D_THEAFRICA_RANK_INST_LAST_DEG_2
D_STEM_STUDY_PROG
D_CS_STUDY_PROG
D_SPELLING

Table 4.2.: The names of the new features introduced by feature engineering. For each ranking 2 features were introduced, because applicants are allowed to enter up to two universities.

	Accept	Reject	No Label
UdS Offer Accepted	195	235	29
Other Applicants	0	0	11394
Total	195	235	11423

Table 4.3.: Summary of the strong labelling.

4.2. Data Labelling

Since the following section mainly works with decision trees, which are among the supervised learning methods, the data must be labelled. The labelling of the data is a very important step because it tells the machine learning model which students should be re-accepted and which ones should be rejected. In the following sections, different labels are introduced, which are tested in the modelling chapter.

4.2.1. Data Labelling based on Performance at Saarland University

After consultation with the computer science faculty, it was decided to label the data twice. It turned out that under the original internal requirements, too few of the students who had studied at Saarland University would be re-admitted. Therefore, a labelling system was introduced which met these high requirements, and another one was also introduced which had slightly lower requirements. In the following sections the labellings are defined and statistics are provided. In this labelling only students whose performance is known are labelled. In the next section the two labellings are extended with students who did not show up at Saarland University.

Strong Labelling

The strong labelling requires students to have at least 20 CP per semester and an average grade better than or equal to 3.0. Figure 4.5 shows the labelling decision diagram. Overall, 459 students who accepted their offer are considered. However, it is not possible to label 29 of these students since they accepted their offer recently and hence grades are not available. 200 of these students did not achieve 20 CP per semester and 129 students did not achieve an average of 3.0 in core lectures. 94 students neither achieved 20 CP per semester nor an average of at least 3.0 in core lectures. 195 students in the data met the requirements, 5 of whom were exmatriculated before graduation. This means that only 45.12 achieved the internal targets at Saarland University and 54.88% would not obtain an offer anymore. As this is very selective, an additional labelling is introduced below. It allows all experiments to be performed in parallel and has lower requirements. A total of 430 applicants were labelled. A brief summary of these statistics is provided in table 4.3.

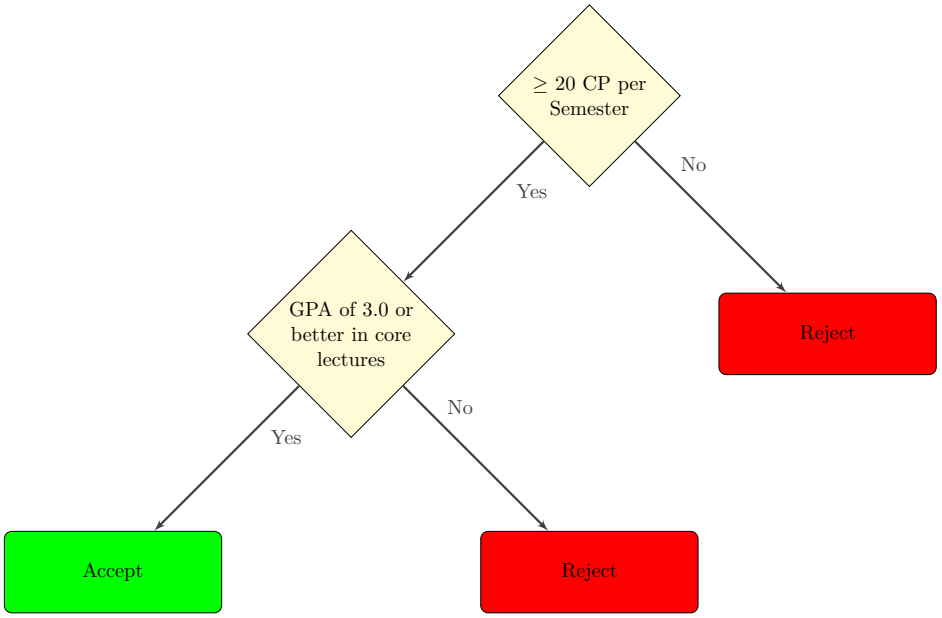


Figure 4.5.: Decision process for strong labelling of data. Only students for whom grades are available at Saarland University are considered.

	Accept	Reject	No Label
UdS Offer Confirmed	283	147	29
Other Applicants	0	0	11394
Total	283	147	11423

Table 4.4.: Summary of the weak labelling.

Weak Labelling

The weak labelling requires students to have at least 15 CP per semester and an average grade better than or equal to 3.3. Figure 4.6 shows the labelling decision diagram. As stated above, 459 students who accepted their offer are considered and we can only label 430 students since for the other 29 students grades are not available yet. 104 of those students did not achieve 15 CP per semester and 94 students did not achieve an average of 3.3 in core lectures. 51 of those students neither achieved 15 CP per semester nor an average of at least 3.3 in core lectures. 283 students met the requirements, 11 of whom were exmatriculated before graduation. This means that only 65.81% achieved these targets and 34.19% would not obtain an offer anymore according to this labelling. This is significantly less selective than the strong labelling. A total of 430 applicants were labelled. A brief summary of these statistics is provided in table 4.4.

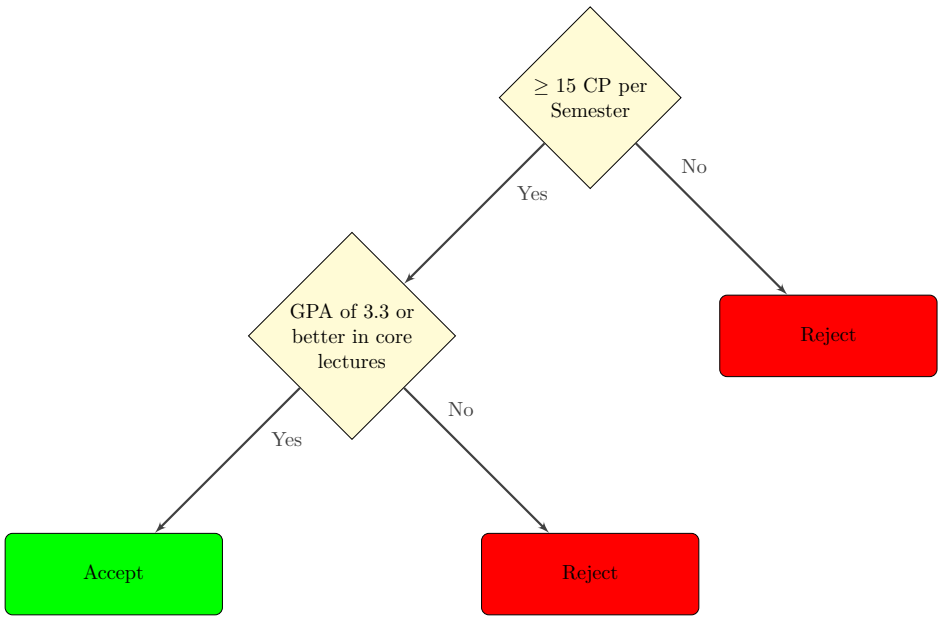


Figure 4.6.: Decision process for weak labelling of data. Only students for whom grades are available at Saarland University are considered.

4.2.2. Data Labelling for Applicants with unknown Performance

The two labellings introduced above are only based on the students performance at Saarland University. However, based on this information it is possible to draw conclusions about other applicants who have not been at Saarland University. So far three applicant groups have not been labelled yet. There is no information about the applicants who were rejected, who did not accept their offer and about those who accepted their offer but have not completed a semester yet. The applicants who were rejected by Saarland University are most likely weaker than those who were accepted. For instance, students with a certain GPA could be labelled negatively. According to application requirements, applicants should have a GPA of at least 75%. Only in exceptional cases, for example if the applicant has studied a particularly difficult subject at a very good university, applicants with a lower GPA are considered. In addition, one could label students who have not studied a STEM-based subject negatively because they do not have the necessary mathematical and technical knowledge. In practice, this group has hardly ever been admitted to studies in computer science. One further possibility is to negatively label applicants from universities from which students have done particularly badly at Saarland University from a certain grade point average. Also, one could try to label students who have declined their offer positively or negatively. This could be done by considering their university and their grades. Applicants who come from the group of rejected applicants cannot be labelled positively. The reason for the rejection of applicants is unclear, if the applicant had a good academic background. It is conceivable, for example, that the applicant's knowledge of English was insufficient. It was barely possible to reconstruct this since the database was not maintained. In the following

	Accept	Reject	No Label
UdS Offer Confirmed	195	235	29
Other Applicants	0	1358	10036
Total	195	1593	10065

Table 4.5.: Summary of the strong labelling. Some applicants with unknown performance are also labelled negatively. The size of the new labelled data set is 1788.

two paragraphs three labellings are introduced which implement the ideas above. First, only the new negatively labelled ones are added to the data and then the same is done for the new positively labelled ones. Finally, the combination of both is tested.

Adding negative labelling from applicants with unknown performance

As mentioned above, the negative labels for applicants who did not obtain an offer are added in order to have a bigger data set to learn from. The labelling process is shown in Figure 4.7. In principle, all applicants who have not completed a scientific or technical degree are rejected. In addition, students with a GPA of less than 60% are automatically rejected. Students who come from universities from which none or only one student has ever studied at Saarland University are not labelled. The same applies for students who come from a university where less than 70% of the students have not met the requirements (according to the weak or strong labelling). The others are labelled negatively. This guideline ensures that the students would not actually meet the requirements of Saarland University. A brief summary of the corresponding statistics of the strong labelling is provided in table 4.6 and of the weak labelling in table 4.6.

Adding positive labelling from applicants with unknown performance

A similar procedure is now applied to applicants who have received an offer but declined it. The system checks whether students from the same university have performed well (according to the weak or strong labelling) and positively labels all applicants who have at least the GPA of a student from the same university with the label "Accept". Applicants who come from universities from which no students have ever studied at Saarland University are not labelled. Students who have a worse grade than the worst grade of the positively graded university will not be graded either. The labelling process is shown in Figure 4.8. A brief summary of the corresponding statistics of the strong labelling is provided in table 4.7 and of the weak labelling in table 4.8.

Adding positive and negative Labelling from Applicants with unknown performance

In this labelling all labels for applicants who have never attended to Saarland University are simply added to the record. In principle, both procedures, which are listed above, are combined and help to obtain significantly more labelled data.

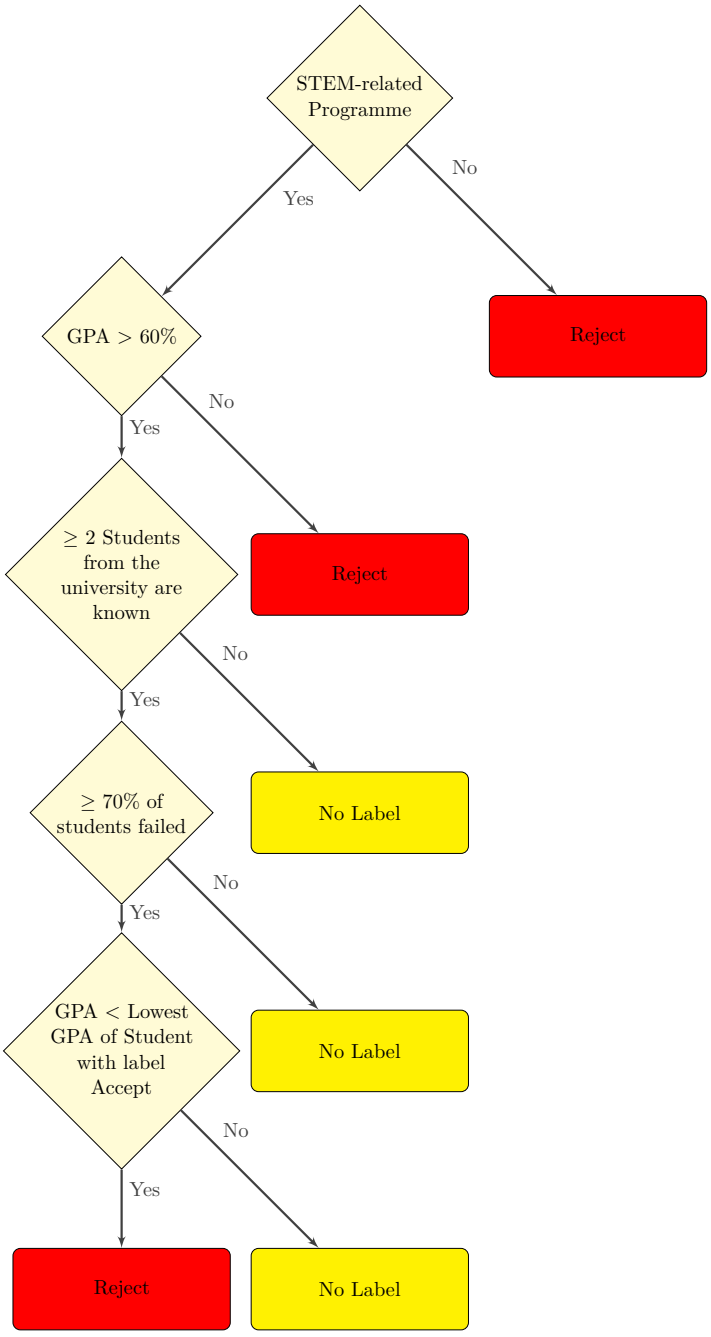


Figure 4.7.: Decision process for negative labelling of applicants with unknown performance at Saarland University.

	Accept	Reject	No Label
UdS Offer Confirmed	283	147	29
Other Applicants	0	698	10696
Total	283	845	10725

Table 4.6.: Summary of the weak labelling. Some applicants with unknown performance are also labelled negatively. The size of the new labelled data set is 1128.

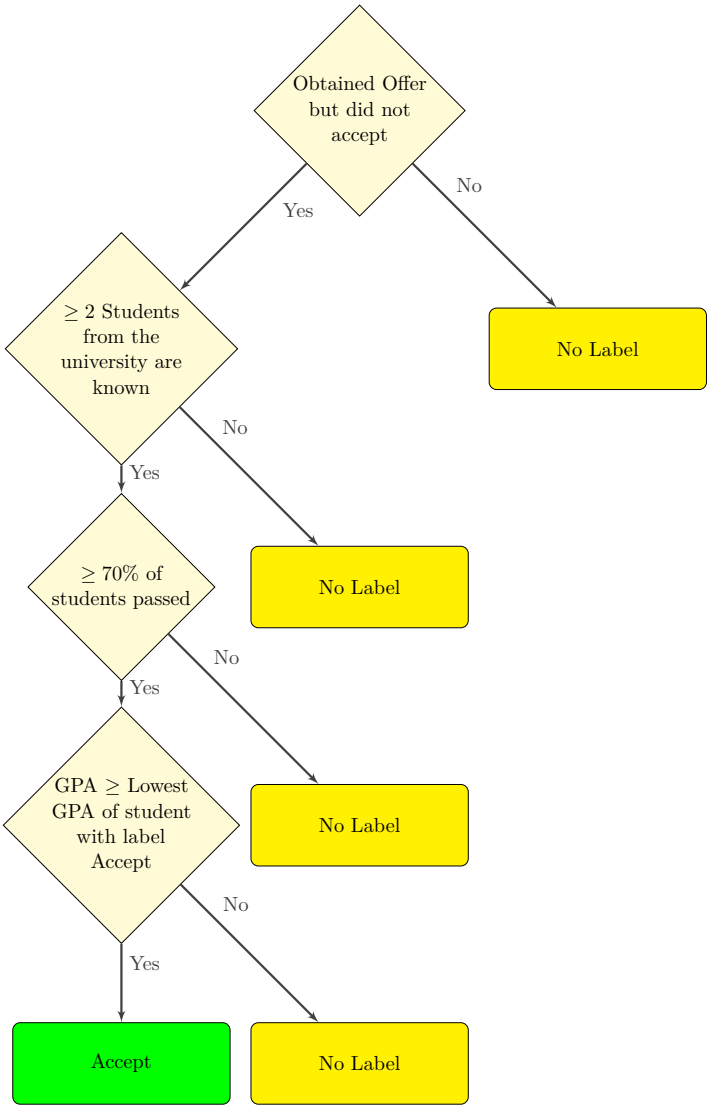


Figure 4.8.: Decision process for positive labelling of applicants with unknown performance at Saarland University.

	Accept	Reject	No Label
UdS Offer Confirmed	195	235	29
Other Applicants	21	0	11373
Total	216	235	11402

Table 4.7.: Summary of the strong labelling. Some applicants with unknown performance are also labelled positively. The size of the new labelled data set is 451.

	Accept	Reject	No Label
UdS Offer Confirmed	283	147	29
Other Applicants	328	0	11066
Total	611	147	11095

Table 4.8.: Summary of the weak labelling. Some applicants with unknown performance are also labelled positively. The size of the new labelled data set is 758.

5. Model Training and Evaluation

As already mentioned in the introduction, the focus of this thesis is on decision trees. In the following paragraphs, these are trained on the data set with the previously introduced labels. Performance is optimised by using different data preprocessing, different algorithms and parameters. First of all, the data set is considered which only contains the students who were at Saarland University and whose performance is known. Both, strong and weak labelling are considered. Table 5.1 shows the 24 features with which the models are trained. For the model training and the evaluation of the data, the python packages in table A.3 were used.

5.1. Training with Applicants with known Performance

The decision tree is trained on the record with the students from whom we know the performance. All steps are performed in parallel for both labels. The data set contains 430 applicants, of which 195 students would be re-admitted according to the strong labelling and 283 students would be re-admitted according to the weak labelling. A total of 24 features are available, which are gained from the application form and feature engineering. The features which have been recorded at Saarland University during the studies of the accepted applicants cannot be used for the training. The features which are considered to create the decision tree are listed in table 5.1. First a decision tree is trained without restrictions and parameters on the data. The data was divided into training data and test data. 80% of the data was used for training and 20% of the data for testing. This division is maintained in all subsequent experiments. An accuracy of about 59% is achieved with strong labelling and weak labelling. However, there are some differences. It is noticeable that in the case of labelling with high requirements, the students who will be rejected can be better identified than those who will be accepted. For the data based on the weak labelling, the exact opposite is the case. Students to be accepted are more likely to be predicted than those to be rejected. This also makes sense, since the data sets are correspondingly unbalanced. But overall the performance is rather weak, because a classifier, which would always predict the class that occurs most, would achieve an accuracy of 54.65% or 65.8%. This low accuracy illustrates that further effort must be put into the decision trees and data. In several experiments, different algorithms and techniques are considered and finally combined.

Decision Tree Modelling

As stated in the above paragraph, the performance of a simple decision tree is not very good. This is partly due to the fact that the data records are unbalanced and not all features are always needed. There are also several parameters that can be optimised. In the following experiments, the different feature elimination techniques are combined with sampling methods. The abbreviations listed in table 5.2 apply to the following

Feature Name
AGE
GENDER
MARITAL_STATUS
NATIONALITY
STUDY_PROG_LAST_DEG_1
STUDY_PROG_LAST_DEG_2
DEGREE
LAST_DEG
GPA
STUDY_PROG_CHOICE_1
STUDY_PROG_CHOICE_2
INST_LAST_DEG_1
INST_LAST_DEG_2
D_QS_RANK_INST_LAST_DEG_1
D_QS_RANK_INST_LAST_DEG_2
D_THE_RANK_INST_LAST_DEG_1
D_THE_RANK_INST_LAST_DEG_2
D_THEASIA_RANK_INST_LAST_DEG_1
D_THEASIA_RANK_INST_LAST_DEG_2
D_THEAFRICA_RANK_INST_LAST_DEG_1
D_THEAFRICA_RANK_INST_LAST_DEG_2
D_STEM_STUDY_PROG
D_CS_STUDY_PROG
D_SPELLING

Table 5.1.: Features for Model Training

sections. Table 5.3 and table 5.4 show the performance of the decision trees for the strong and the weak labelling. For each experiment the accuracy, precision, recall and the F1 score is given. The precision corresponds to the positive predicted value (PPV) and the recall to the true positive rate (TPR). The positive class are the applicants with the label "Accept". Also, the negative predicted value (NPV) and the true negative rate (TNR) are provided. Only the best combinations of the individual methods are listed in the table. If a decision tree gives the best results for training data with the best 10 features according to recursive feature elimination, the results for the other possibilities (e.g. 9 features) are not provided.

Evaluation

No major improvements were achieved in terms of accuracy. In the strong labelling, the maximum value is achieved using the best 18 features according to recursive feature elimination in combination with random oversampling. The accuracy of this combination is 0.605, which is only marginally better than a decision tree on unprocessed data. With this combination the best F1 score was also achieved. The lowest accuracy and the lowest F1 score is achieved with decision trees that have been trained on data, that have been undersampled and that have been selected with the best 7 features according to chi-square tests. The accuracy here is 0.558 and the F1 score 0.345. Overall, it is noticeable that the precision is less than the NPV in each experiment. With the recall and TNR scores, the same can be observed in almost all experiments.

Similar observations can be made for the data with weak labelling. The maximum accuracy and the maximum F1 score are achieved with decision trees, which are based on data balanced with oversampling and reduced to their 12 most important features by recursive feature elimination. The accuracy is 0.616 and the F1 score is 0.697. The worst performing decision trees are those trained on data balanced by undersampling. It is noticeable that the precision is always higher than the NPV. This is also true for recall and TNR, except for the experiments where the data is balanced with undersampling.

In terms of accuracy, the performance of the decision trees in the experiments is very poor. Simple classifiers, which always return only the class that occurs most often, achieve a similar or even better accuracy. The performance is mainly due to the limited data. This is also shown by the fact that when the data is balanced by undersampling, the accuracy decreases significantly. Often the most important features of the better models are grades, study programmes, former universities or nationalities. However, since each university appears only a few times in the data set, this makes the learning process more difficult. There are also some coded null values for the grades. In the next step, these are removed and the performance is checked again.

The same experiments were carried out again, but now the applicants who did not indicate grades or subjects were removed. This reduced the number of applicants in the data set from 430 to 319. In the data set with the strong labelling, 64% of the applicants were rejected and 36% of the applicants were accepted. In the data set with weak labelling, 54.7% of the applicants were accepted and 45.3% were rejected. The results of the weak labelling have not changed significantly in terms of metrics such as accuracy.

K best features with recursive feature elimination	KBest RFE
K features with highest chi-square score	KBest Chi2
Random oversampling	OS
Random undersampling	US
Synthetic minority oversampling technique	SMOTE

Table 5.2.: Abbreviations of Methods

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
-	0.581	0.523	0.605	0.643	0.563	0.561
9Best Chi2	0.581	0.545	0.316	0.594	0.792	0.4
12Best RFE	0.593	0.538	0.553	0.638	0.625	0.545
22Best Chi2, OS	0.581	0.524	0.579	0.636	0.583	0.550
7Best Chi2, US	0.558	0.5	0.263	0.576	0.792	0.345
7Best Chi2, SMOTE	0.558	0.500	0.316	0.581	0.750	0.387
18Best RFE, OS	0.605	0.548	0.605	0.659	0.604	0.575
2Best RFE, US	0.570	0.512	0.579	0.628	0.562	0.543
20Best RFE, SMOTE	0.593	0.545	0.474	0.623	0.687	0.507

Table 5.3.: Performance of decision trees on strong labelled data.

It should be noted, however, that these have been obtained on a more balanced data set with less data. The data set with the strong labelling is less balanced than before. In most of the experiments, similar results were obtained like above. However, for the 6 features with the highest six chi-square values a higher accuracy (0.672) could be achieved. Indeed, the F1 score is only 0.488 due to the low precision and low recall.

Random Forests

The experiments were also carried out for random forests. These performed slightly better than the decision trees. In the data set with strong labelling an accuracy of 0.616 was achieved using the 12 best features determined by recursive feature selection. The F1 score is 0.571, mainly due to the lower recall and precision for positively labelled applicants. In the data set with the strong labelling and without null values an accuracy of 0.639 and an F1 score of 0.537 was achieved. The random forests were trained to data that were over-sampled using SMOTE. In addition, only the 12 most important features identified with recursive feature elimination were considered. A maximum accuracy of 0.639 is also achieved for data with weak labelling. The F1 score of 0.73 is significantly higher, but this is due to the high precision and recall and the unbalanced data set. This was only achieved with random forests and the 20 features with the highest chi-square values. In the data set without null values the accuracy could be increased to 0.656. The F1 score is 0.73. This was achieved using random forests and data over-sampled with SMOTE and the 9 features with the highest chi-square values. Table 5.5 shows once again the exact values for the best experiments.

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
-	0.593	0.711	0.649	0.412	0.483	0.679
20Best Chi2	0.605	0.725	0.649	0.428	0.517	0.685
10Best RFE	0.616	0.74	0.649	0.444	0.552	0.692
22Best Chi2, OS	0.581	0.706	0.632	0.4	0.483	0.667
20Best Chi2, US	0.535	0.757	0.439	0.396	0.724	0.535
22Best Chi2, SMOTE	0.581	0.733	0.579	0.415	0.586	0.647
12Best RFE, OS	0.616	0.731	0.667	0.441	0.517	0.697
22Best RFE, US	0.535	0.774	0.421	0.400	0.759	0.545
13Best RFE, SMOTE	0.605	0.735	0.632	0.432	0.552	0.679

Table 5.4.: Performance of decision trees on weak labelled data.

Dataset	Accuracy	Precision	Recall	NPV	TNR	F1 Score
Strong Labelling	0.616	0.564	0.579	0.660	0.646	0.571
Strong Labelling NA	0.639	0.621	0.474	0.649	0.771	0.537
Weak Labelling	0.639	0.724	0.736	0.464	0.448	0.730
Weak Labelling NA	0.656	0.667	0.743	0.640	0.552	0.703

Table 5.5.: Performance of the best random forest models.

5.2. Adding negatively labelled Applicants with unknown Performance

In section 4.2.2, a labelling system was introduced which gives negative labels to applicants who have never studied at Saarland University. The labelling is shown in figure 4.7. In the following section, these applicants are added and the above experiments are carried out analogously. Also, the applicants who did not indicate a GPA were removed in the following experiments. It has been shown that despite the loss of information, this adds value to performance.

Strong Labelling

The data set with the strong labelling is thus increased to 1675 students. However, the data set is now extremely unbalanced. 91.18% of applicants are labelled negatively and 8.82% positively. As described above, different combinations of feature selection and sampling methods were tried. The best results are shown in table 5.6. Accuracy has been significantly improved and is over 80% in all cases and over 90% in some cases. However, it should be noted that the classes are extremely unbalanced and a classifier that would only reject applicants would receive an accuracy greater than 90%. It is interesting that the addition of negatively labelled applicants increased the recall of positively labelled applicants in some experiments. When the training data was balanced with undersampling, a decison tree trained on the 14 best features according to chi-square scores resulted in a recall of 0.829. This is significantly better than in all previous experiments. However, this is accompanied by a lower accuracy and precision,

as more negatively labelled applicants are classified as positive. Nevertheless, a model with an accuracy, recall and TNR of about 80% is a very solid model. In other listed experiments a higher accuracy was achieved, but the recall is always less than or equal to 50%. This is too low for a model that is supposed to support the application process.

Weak Labelling

As for the data with strong labelling, the negatively labelled applicants are now also added to the data with weak labelling. The new data set includes 1013 applicants of which 79.8% would be accepted and 20.2% would be rejected. Table 5.7 shows the best results for the various combinations. The accuracy in all experiments is over 80%, which is better than a classifier that would always reject an applicant. As in the experiments listed above, by adding negatively labelled applicants, the recall could be increased significantly. In some experiments, the recall is above 0.8 and the precision is also higher, which is also due to more positively labelled applicants in the data set. Overall, significantly better classifiers than before could be trained. The best model was trained on the 16 features with the highest chi-square score on a data set that was undersampled. The accuracy in this model is 0.847, the recall is 0.814 and the TNR is 0.856. Another good model could be trained on the three best features according to recursive feature elimination on an oversampled training data with SMOTE. The three features are GPA, nationality and the university that was visited before. The accuracy is 0.867 and the F1 score is 0.703. It is questionable whether such a model is practical, as it only uses three features. There are other models in the list that use more features and also achieve very good results.

Comparison with Random Forests

The same experiments were also conducted with random forests. The results of data with the strong labelling that random forests provide are comparable to the decision trees. With undersampling and feature selection, random forests also achieve an accuracy and recall of about 0.85. In some cases the accuracy is slightly higher, but in these cases the minority class is discriminated against. In the data set with the weak labelling, random forests performed very similar to decision trees. Random forests trained on the 17 best features according to recursive feature selection and on an undersampled data set achieved an accuracy of 0.847. The recall is 0.907 and the TNR is 0.831. The F1 score and the precision are therefore also very similar.

Summary

Overall, adding negatively labelled applicants has improved the performance of the decision trees enormously. Without additional information from positive-labelled applicants, the recall could be increased significantly. This is mainly due to the use of feature selection and sampling methods. In most cases, random undersampling was the best sampling method. Interestingly, the undersampling method, condensed nearest neighbour, used on the dataset brought less added value than random undersampling.

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
12Best Chi2, OS	0.919	0.682	0.429	0.936	0.977	0.526
14Best Chi2, US	0.821	0.349	0.829	0.976	0.820	0.491
9Best Chi2, SMOTE	0.872	0.395	0.429	0.933	0.923	0.411
7Best RFE, OS	0.904	0.571	0.343	0.927	0.970	0.489
7Best RFE, US	0.848	0.389	0.800	0.973	0.853	0.523
16Best RFE, SMOTE	0.869	0.385	0.429	0.932	0.920	0.405

Table 5.6.: Performance of decision trees on strong labelled data. The data includes negatively labelled applicants with unknown performance.

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
15Best Chi2, OS	0.852	0.651	0.651	0.906	0.906	0.651
16Best Chi2, US	0.847	0.603	0.814	0.945	0.856	0.693
17Best Chi2, SMOTE	0.828	0.577	0.697	0.914	0.863	0.632
8Best RFE, OS	0.842	0.628	0.628	0.900	0.900	0.628
3Best RFE, US	0.842	0.590	0.837	0.951	0.844	0.692
3Best RFE, SMOTE	0.867	0.667	0.744	0.929	0.900	0.703

Table 5.7.: Performance of decision trees on weak labelled data. The data includes negatively labelled applicants with unknown performance.

The next step is to see if the model can be improved by adding positively labelled students.

5.3. Adding positively labelled Applicants with unknown Performance

The same experiments were performed analogously for the data set with additional positively labelled students. Again, applicants who did not give any grades or study programme were removed. The data set with strong labelling includes 367 students, of whom 43.3% would be accepted again. The data set with the weak labelling was increased to 674 applicants, of whom 79.3% were positively and 20.7% negatively labelled. As before, all possible combinations were tried out. However, the performance of the models could only be slightly improved if the decision trees were trained on balanced data. In most cases, the accuracy was around 60%, while recall and TNR scores were very similar. In cases where accuracy was higher, the majority class was preferred and most of the minority class was assigned to it. As this method did not add any value to our models, the results are not presented here. The combination of both labellings is also not considered.

5.4. Model Tuning

In this section, work will continue with the data set containing the additional negatively labelled students. The successful models discussed above are now being optimised using various parameters. These parameters are systematically tested with a grid-search. All different combinations of feature selection, resampling and decision trees are tested on the training data and the parameters are selected. Finally, the final model is tested on the test data.

Some parameters of decision trees in scikit-learn are (parameter names and descriptions are reprinted and adapted from [72]):

- criterion: The function to measure the quality of a split - Gini impurity or Information gain.
- max_depth: The maximum depth of a tree.
- min_samples_split: The minimum number of samples required to split an internal node.
- min_samples_leaf: The minimum number of samples required to be at a leaf node.
- max_leaf_nodes: The maximum number of leaf nodes. [72]

For these parameters different values have now been defined and tested. For example, the depth was set to be at least two, but not more than six. Information gain and gini impurity were also tried. For min samples split, values from two to 40 were tried, considering the data size. Mathematically, only 64 leaf nodes in a binary tree with a depth of six are possible. Therefore different values were tried up to 60.

Model trained on strong labelled Data

With the strong labelling a decision tree could be constructed, which achieves an accuracy of 0.884 on the test data. The F1 score is 0.598 and the recall and TNR are well over 0.8. Table 5.8 shows the exact values and figure 5.1 the confusion matrix of the decision tree. The model was trained on randomly over sampled data on the seven features with the highest chi2 value. These seven features are:

- GPA
- INST_LAST_DEG_1
- NATIONALITY
- D_QS_RANK_INST_LAST_DEG_1
- D_QS_RANK_INST_LAST_DEG_2
- D_THE_RANK_INST_LAST_DEG_1
- D_THE_RANK_INST_LAST_DEG_2

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
7Best Chi2, OS	0.884	0.467	0.829	0.978	0.890	0.598

Table 5.8.: Performance of a decision tree after parameter tuning on strong labelled data. The data includes applicants with unknown performance.

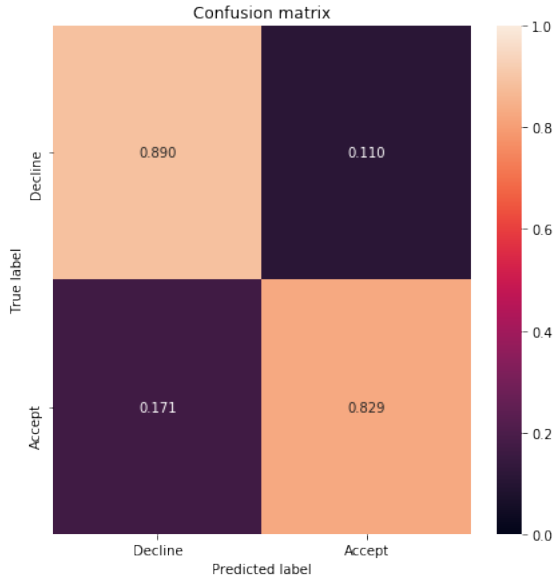


Figure 5.1.: Confusion matrix of the best model after parameter tuning on strong labelled data. The data includes applicants with unknown performance.

It is noticeable that all features refer to the academic achievements and the university and country of study. In particular, the feature engineering could provide more information for the decision tree with the rankings. Another positive aspect is that the decision tree does not make decisions based on gender, age or marital status, which means that at least in this form it makes very neutral decisions based on the academic background. However, this decision tree raises the question whether, despite the very good performance, it is a good decision tree for the admission process. The fact that the decision tree does not take the study programme into account already speaks against it. This decision tree would admit students who have not studied natural sciences and are therefore not suitable for a study of computer science at all.

In this model the decision tree is trained on data that have been balanced with random oversampling. In addition, the 8 most important features were selected according to recursive feature selection. The decision tree achieves an accuracy of 0.857 and the recall and TNR are at least 0.8. However, the precision of 0.406 is relatively low, which is related to the unbalanced data set on which the tests were performed. The results are summarised in table 5.9.

It was also examined whether the decision tree could be improved with cost-complexity pruning. Figure 5.2 shows that removing sub-trees does not improve the accuracy of

the decision tree. With increasing alpha the accuracy decreases steadily.

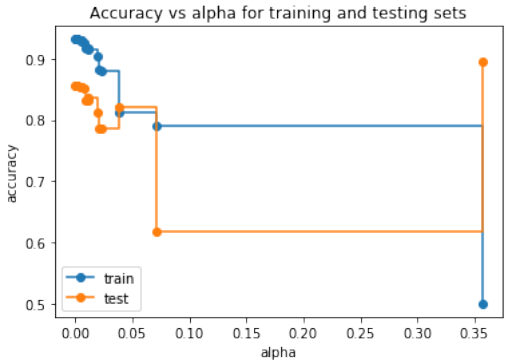


Figure 5.2.: Accuracy of the decision tree with increasing alpha. The decision tree was fitted on data with the strong labelling. The data includes applicants with unknown performance.

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
8Best RFE, OS	0.857	0.406	0.800	0.974	0.863	0.538

Table 5.9.: Performance of a decision tree after parameter tuning on strong labelled data. The data includes applicants with unknown performance. This decision tree considers the study programme.

Model trained on weak labelled Data

As for the strong labelling, an attempt was also made to increase the performance of the weak labelling by parameter tuning. The decision tree trained on weak labelled data achieves an accuracy of 0.847, with a depth of 5, on the best 14 features, according to recursive feature elimination, and on data balanced by oversampling. The number of leaf nodes was limited to 20 and there must be at least 4 samples per split. The recall and TNR are clearly above 80. The detailed statistics are provided in table 5.10. It was also investigated whether cost-complexity pruning leads to an improvement in accuracy [69]. It is only possible to reduce the complexity of the tree while maintaining accuracy. Figure 5.3 shows the accuracy with increasing alpha. However, the pruned decision tree would lead to a lower TNR. Since the actual decision tree has a limited complexity due to its limited depth, we continued to work with it. In this model the features GPA and nationality are very important. The sum of their feature importance is over 0.9. Asia rankings, university and other features are also important, but these have a very small share in the classification.

Random Forests

The same experiments were also conducted for random forests. These have the parameters of the decision trees and a parameter that indicates the number of estimators.

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
14Best RFE, OS	0.847	0.600	0.837	0.951	0.850	0.699

Table 5.10.: Performance of a decision tree after parameter tuning on weak labelled data. The data includes applicants with unknown performance. This decision tree considers the study programme.

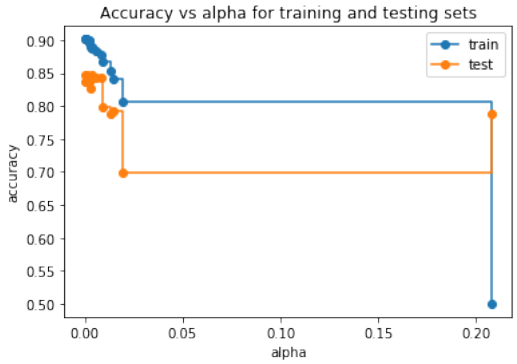


Figure 5.3.: Accuracy of the decision tree with increasing alpha. The decision tree was fitted on data with the strong labelling.

Strong Labelling

With the strong labelling data set, a classifier was trained on data balanced with SMOTE and the best 9 features, achieving an accuracy of 0.866. The recall and TNR are above 0.8. The exact values are in table 5.11. The maximum tree depth is 6 and 40 estimators were used. The main features were grades, university, nationality, age and QS rankings. Together these make up 85% of the feature importance.

Weak Labelling

The results of random forests on the weak labelling data set are very similar to the results of the decision tree. The 8 best features were selected according to recursive feature elimination. Some of these are nationality, GPA, university, age and the study programme. An accuracy of 0.847 was achieved and the recall and TNR are over 83%. The results are summarised in table 5.12.

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
9Best RFE, SMOTE	0.866	0.424	0.800	0.974	0.873	0.554

Table 5.11.: Performance of random forests with parameter tuning on strong labelled data. The data includes applicants with unknown performance.

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
8Best RFE, OS	0.847	0.594	0.884	0.964	0.837	0.710

Table 5.12.: Performance of random forests with parameter tuning on weak labelled data. The data includes applicants with unknown performance.

5.5. Comparison with Boosting Methods

One way to achieve a better performance with decision trees, besides bagging, is also boosting. Since boosting generally gives very good results, the different boosting algorithms are tried out for comparison with the decision trees mentioned above.

AdaBoost

As in the previous sections, the model was trained on the data with the strong and the weak model. In both models the data sets were balanced with undersampling and the most important features were selected with recursive feature elimination. With the AdaBoost an accuracy of over 86% was achieved on the data set with the strong labelling. The recall and TNR are above 0.85. An accuracy of about 87% was achieved on the data set with the weak labelling. The values for the recall are also similar to the model on the strong labelling. Overall, the performance of both models is very solid. AdaBoost will not be discussed further, as significantly better results have been achieved with gradient boosting.

Gradient Boosting

In the following section, models are created for both data sets using extreme gradient boosting. This algorithm performs very well in practical applications. Gradient boosting is therefore mainly used for comparison with the other models, since decision trees, for example, have a lower performance than more complex models.

On the data set with the strong labelling a model could be trained, which performed significantly better than those seen before. The data was oversampled and the classifier was trained on the 20 most important features according to recursive feature elimination. 13 estimators were used in the model. An accuracy of over 90% is achieved and the recall and TNR are over 80%. Table 5.13 summarises the values again and in figure 5.4 the confusion matrix is depicted. However, the use of such a model also results in less explainability compared to decision trees. The five most important features are:

- GPA
- NATIONALITY
- INST_LAST_DEG_1
- D_THE_RANK_INST_LAST_DEG_1
- GENDER

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
XGBoost, 20Best RFE, OS	0.901	0.518	0.800	0.975	0.913	0.629

Table 5.13.: Performance of XGBoost on strong labelled data.

Model & Methods	Accuracy	Precision	Recall	NPV	TNR	F1 Score
XGBoost, 17Best RFE, US	0.867	0.625	0.930	0.978	0.850	0.748

Table 5.14.: Performance of XGBoost on weak labelled data.

These features are very similar to the other models. But the asia rankings, age, spelling and whether a student has a science degree are also important.

On the data set with the weak labelling, XGBoost [11] was used to train a classifier with an accuracy of 0.867, which is better than the other classifiers on the same data set. The classifier was trained on the best 17 features according to recursive selection. Furthermore, the data was balanced with undersampling. The recall and TNR are at least 85%, for those applicants to be accepted, even over 90%. Table 5.14 summarises the values again and in figure 5.5 the confusion matrix is depicted. The main features are similar to those listed above. However, there are differences in feature importance and order. The five most important features are:

- GPA
- NATIONALITY
- AGE
- GENDER
- D_QS_RANK_INST_LAST_DEG_1

In summary, one can say that the boosting methods perform better than individual decision trees or random forests. However, there is the disadvantage that decisions made by boosted classifiers are much more difficult to explain.

5.6. Wilson Lower Bound Score

So far we have looked at performance using various metrics, such as accuracy. However, it would also be very interesting to know how many of the applicants are at least correctly classified. The Wilson lower bound score is a good choice for this. The Wilson lower bound score helps us to compute how many students are correctly classified with at least 95% probability. The score is given for the best models. The results are shown in table 5.15. According to the test data, most models achieve about 80% as their lower bound score. This means that at least 80% of the applicants are correctly classified. The XGBoost model, which was trained on the strong labelled data, achieves the highest value with 86.5%. The evaluation means that the trained models are very reliable.

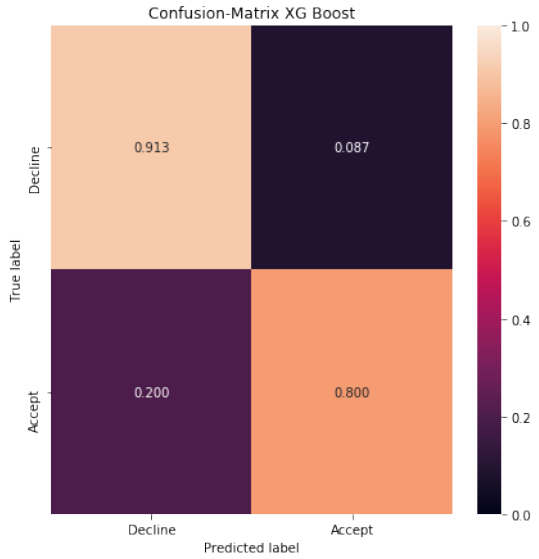


Figure 5.4.: Confusion matrix for a XGBoost model trained on strong labelled data.

Model	Wilson Lower Bound Score
Decision Tree, strong labelled data, 8Best RFE, OS	0.8151
Random Forests, strong labelled data, 9Best RFE, SMOTE	0.8249
XGBoost, strong labelled data, 20Best RFE, OS	0.8649
Decision Tree, weak labelled data, 14Best RFE, OS	0.7914
Random Forests, weak labelled data, 8Best RFE, OS	0.8134
XGBoost, weak labelled data, 17Best RFE, US	0.7914

Table 5.15.: Wilson lower bound scores for the best models.

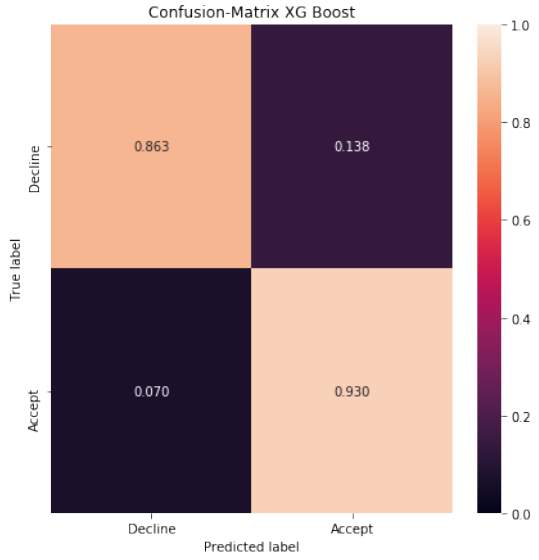


Figure 5.5.: Confusion matrix for a XGBoost model trained on weak labelled data.

6. Approaches for Explainability

An important objective in this thesis is to explain the decisions of the models. In the following chapter, different approaches are discussed to explain the previously presented models. In this chapter the python packages in table A.4 were used.

6.1. Decision Trees

The interpretability of a decision tree is usually very simple. We start at the root of the tree and go to the next node according to the splits until we reach a leaf node. At the end the rules, such as GPA<4 are linked together and we get the reason for the decision of the tree [57]. An important step is therefore the visualisation of the tree and creating the sequence of the corresponding splits. In the following paragraphs, the decision trees are visualised. The decision tree for the strong labelling is shown in figure 6.1. One can see that certain features like grades are especially important. It is also noticeable that some leaf nodes are relatively impure in terms of their classes. This results from the limited depth. The decision tree for the weak labelling is shown in figure 6.2.

Figure 6.3 shows an example of how the decision is made for an applicant whose profile is listed in table 6.1. For this example, the decision tree with the weak labelling was used. Since the applicant's GPA is worse than 2.5 (2 corresponds to $75-79\%$, 5 corresponds to $0-60\%$), only the right sub-tree is considered. If the applicant's GPA is also worse than 3.5, it is rejected.

In figure 6.4 and figure 6.5 we can see the feature importances of the two decision trees. These give us information which features are especially important for the classification. Therefore, this is important additional information that contributes to the understanding of the model. In both models the GPA is by far the most important feature. In the decision tree trained on strong labelled data, the next most important feature is the university, while in the other model it is the nationality. Nationality also plays an important role in strong labelling. The other features have less influence on the features.

The decision of the tree is easy for a human to understand. It is also possible to have these decisions outputted textually. With the feature importances we also have a possibility to estimate which features are especially important in the decision tree. By tracing the path in the decision tree, it is easy to understand which features are responsible for a concrete decision. However, this is not that easy for all models. In the following sections, we explore ways to analyse predictions of the other models, which are rather black-box models.

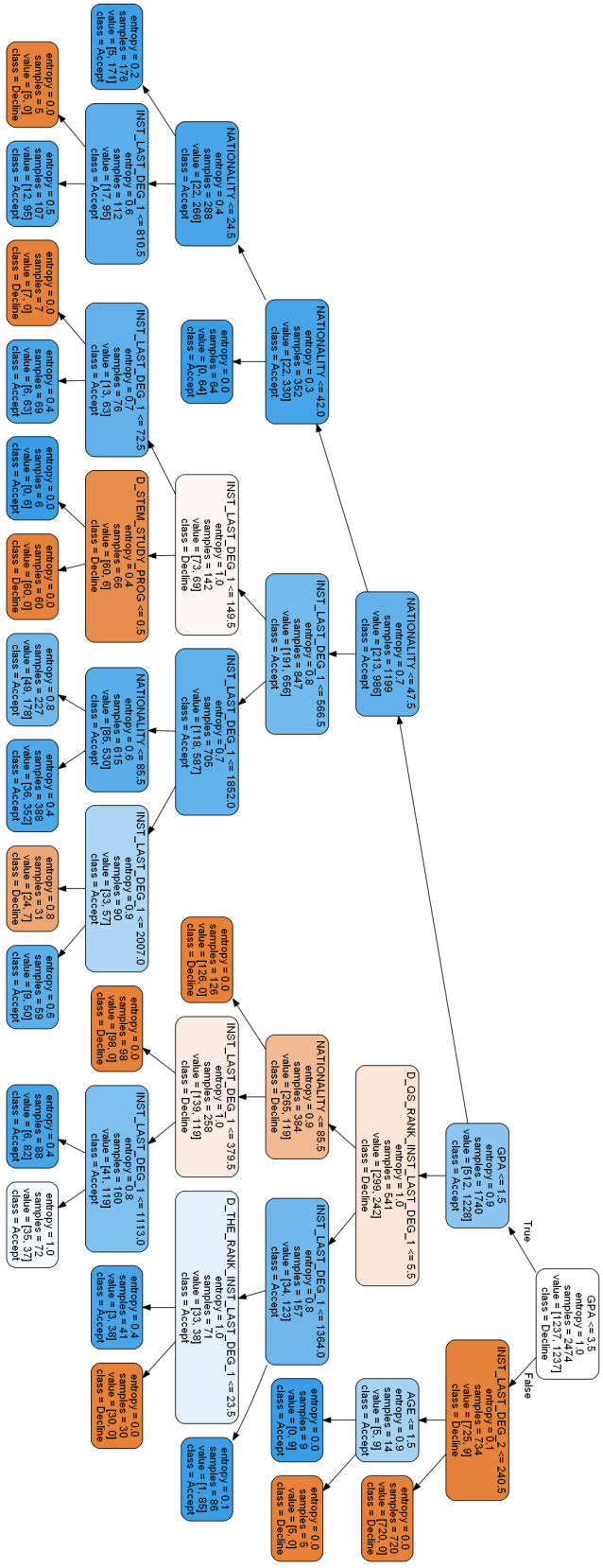


Figure 6.1.: Final decision tree trained on the best 8 features according to recursive feature elimination on strong labelled data. The data includes applicants with unknown performance and was oversampled.

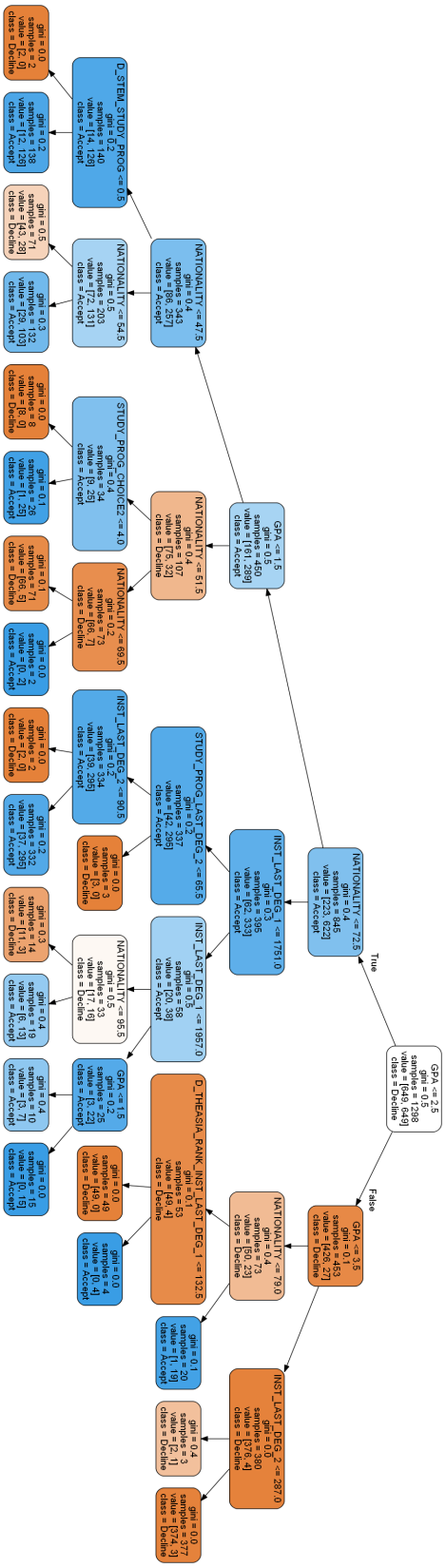


Figure 6.2.: Final decision tree trained on the best 14 features according to recursive feature elimination on weak labelled data. The data includes applicants with unknown performance and was oversampled.

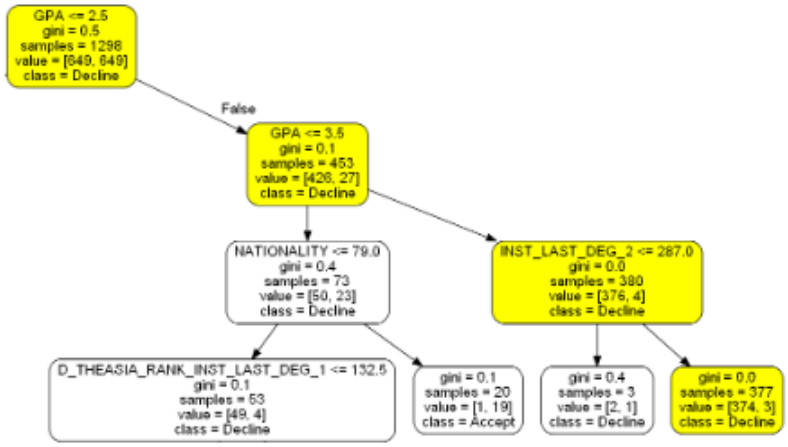


Figure 6.3.: Example: Classification of an applicant. By following the corresponding nodes it is easy to understand why an applicant should get an admission or should be declined.

Feature	Value
STUDY_PROG_CHOICE2	3
GENDER	0
GPA	5
AGE	4
INST_LAST_DEG_1	1898
INST_LAST_DEG_2	1414
NATIONALITY	91
D_THEASIA_RANK_INST_LAST_DEG_1	354
D_THEAFRICA_RANK_INST_LAST_DEG_1	0
D_THEAFRICA_RANK_INST_LAST_DEG_2	0
STUDY_PROG_LAST_DEG_1	0
STUDY_PROG_LAST_DEG_2	1
D_SPELLING	2
D_STEM_STUDY_PROG	1

Table 6.1.: Example: Applicant for the master programme in computer science at Saarland University. The corresponding decision is shown in figure 6.3.

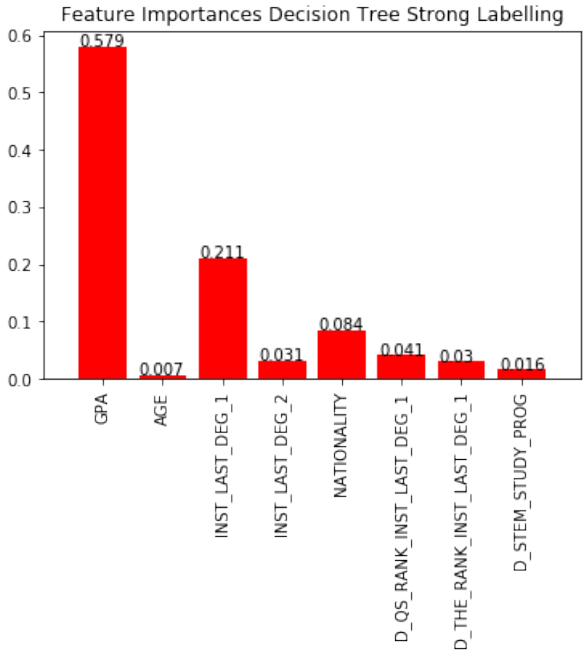


Figure 6.4.: Feature importance of the decision tree trained on strong labelled data.

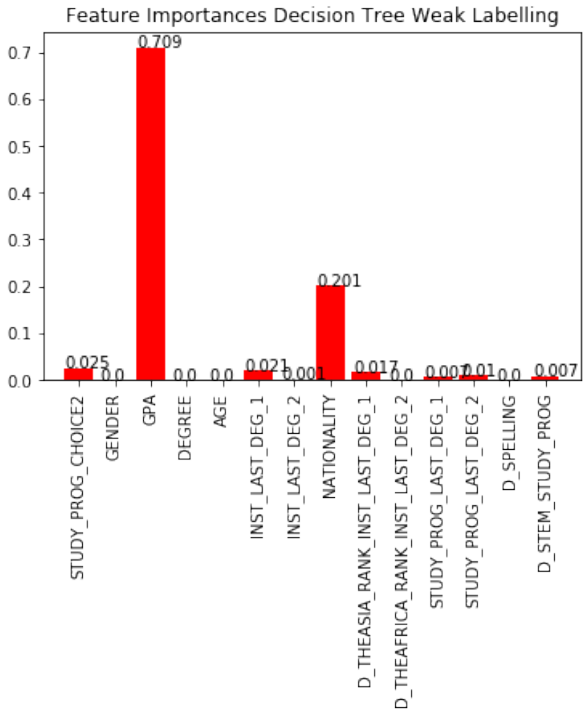


Figure 6.5.: Feature importance of the decision tree trained on weak labelled data.

6.2. Random Forests

Random forests are more complex and difficult to interpret than a decision tree because they consist of numerous decision trees. One way to learn which features are important in the decision of random forests is feature importance. Figure 6.6 shows the feature importances for the strong labelling and figure 6.7 for the weak labelling. In both models the GPA is the most important feature. In addition, nationality and the university at which the student has previously studied play an important role. The feature importance provides a first overview which features are important in the prediction process of the model. However, a better method to identify the most important features are the so-called Shapley values [6, 23]. These allow an evaluation of the importance of the features for the whole model, but also for an individual applicant. In figure 6.8 are the Shapley values for the random forest model trained on the data set with the strong labelling. These are visualised in more detail with values in figure 6.9. In terms of importance, these are similar to feature importance. Various influences on the prediction are visualised. For example, a poor GPA or a high age has a very negative effect on the prediction. The same visualisations were made for the model for data with weak labelling. In figure 6.10 it becomes clear that for the prediction the most important features are the GPA, the age and the nationality. Figure 6.11 visualises the effect of the values of the different features on the predictions. It can be seen that a low GPA has a positive effect on the acceptance of the applicant and that a high age has a negative effect. For the rankings, it should be noted that all universities that are not ranked have a value of 0. In the figure, it seems as if good rankings are penalised. However, that is because most universities are not listed in the THE Asia rankings and therefore have a ranking of 0. In general, it seems a ranked university, i.e., where the ranking is greater than 0, has a very positive impact on the admission.

6.3. Gradient Boosting

Gradient Boosting is a black-box model and explanations for the classifications are much more difficult than in decision trees. But it is possible to explain the predictions of an XGBoost model.

The most important features for the model trained on the strong labelling data are located in figure 6.12. Here it is considered how often a feature is used to split in the individual trees. The most important features are the university of first degree and the study programme. Furthermore, the second preference for a study programme at Saarland University plays a major role. The gender of an applicant and whether the applicant has studied a STEM-related study programme are the least important. However, the feature importances are very inconsistent and offer only a first orientation. The so-called Shapley values are better [51]. These are consistent and indicate which values are most important. In figure 6.13 are the average Shapley values for the data set. Most important is the university where the first degree was obtained, the final grade and the nationality. The Shapley values can also be analysed in more detail. Figure 6.14 shows which features have which effect on prediction at high or low values. While this

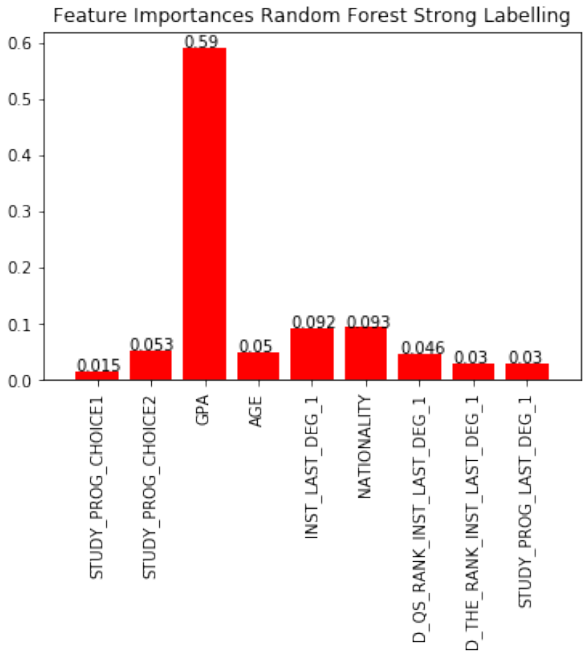


Figure 6.6.: Feature importance of random forests trained on strong labelled data.

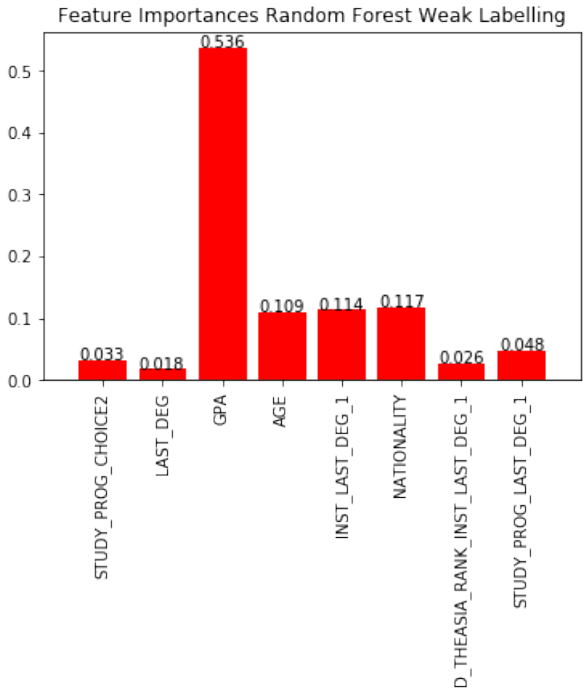


Figure 6.7.: Feature importance of random forests trained on weak labelled data.

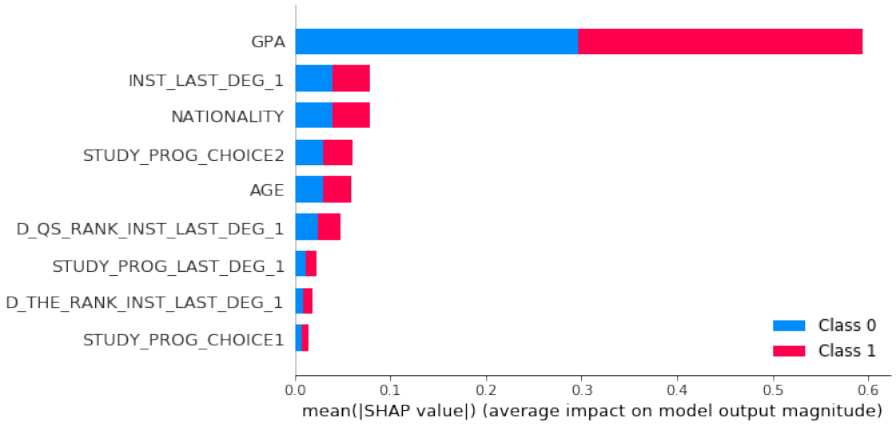


Figure 6.8.: Shapley values of random forests trained on strong labelled data.

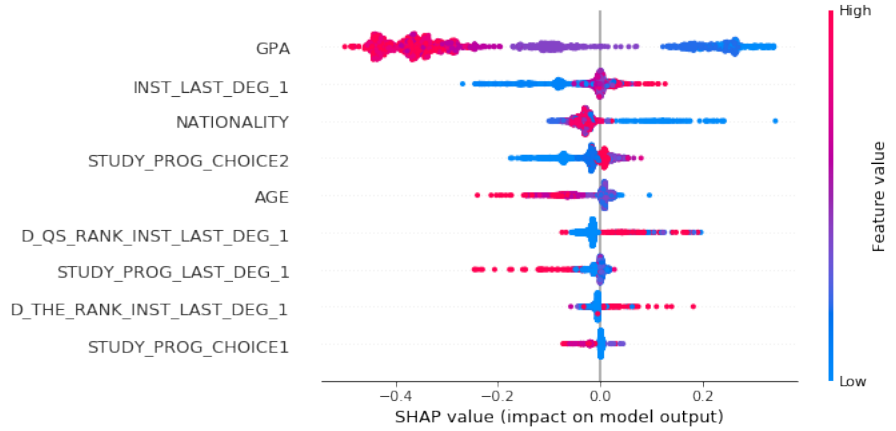


Figure 6.9.: Impact of feature values on Shapley values of random forests trained on strong labelled data.

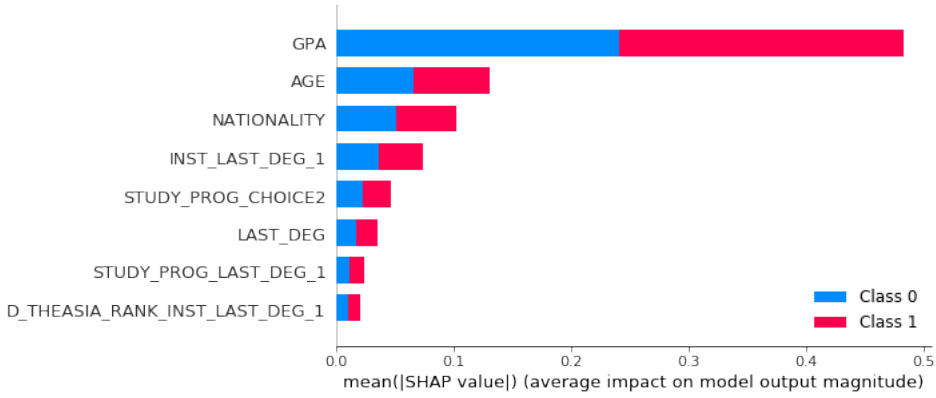


Figure 6.10.: Shapley values of random forests trained on weak labelled data.

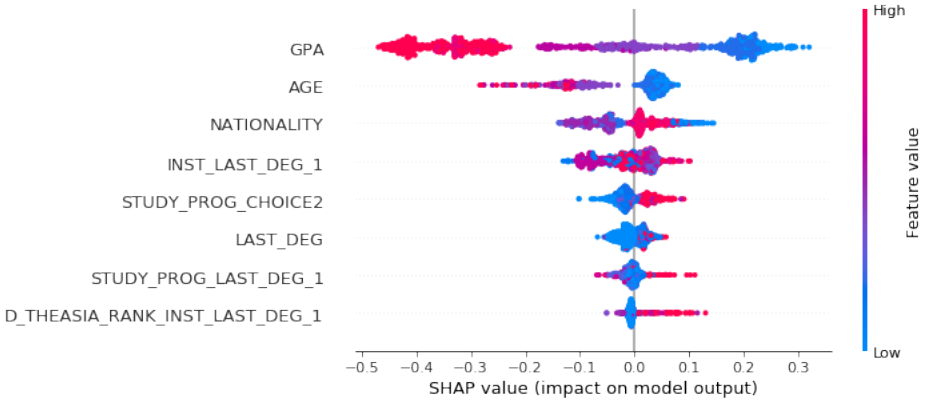


Figure 6.11.: Impact of feature values on Shapley values of random forests trained on weak labelled data.

makes little sense for features like university, as they are categorical, there are clear tendencies for features like age. A high age has a negative influence on the admission of the applicant. Interestingly, the model also seems to slightly favour the gender 1. The Shapley values can also be displayed for individual applicants. This allows to see which features are important and have been crucial in the prediction of the model. In addition, ceteris paribus analyses could be used to determine which effect the change in a single feature would have on the prediction [3]. This can also provide further information about the prediction.

The most important features for the model trained on the strong labelling data are located in figure 6.15. The most important features are the university of first degree, the study programme and the nationality. The average Shapley values for XGBoost on the data with the weak labelling are in figure 6.16. The most important features are similar to the other XGBoost classifier, although there are differences. For example, here the GPA is the most important feature. In figure 6.17 the features are displayed by values and their effect on the prediction. It is noticeable that a bad GPA has a very negative influence on the approval. The model also seems to favour women who have the value 1 over men. At first glance, it also appears that a low value in the QS rankings is a disadvantage. However, this is not the case, as all universities that are not listed in the rankings are given the value 0. Therefore, slightly higher values are much better, as this means that the students are listed in the rankings. As mentioned above, the Shapley values can also be determined for a single applicant.

In the previous sections we have seen different approaches to better understand decisions of machine learning. For humans, decision trees are the easiest to understand, while other models are black-box models. For them there are methods, for example Shapley values, to make their decisions more understandable.

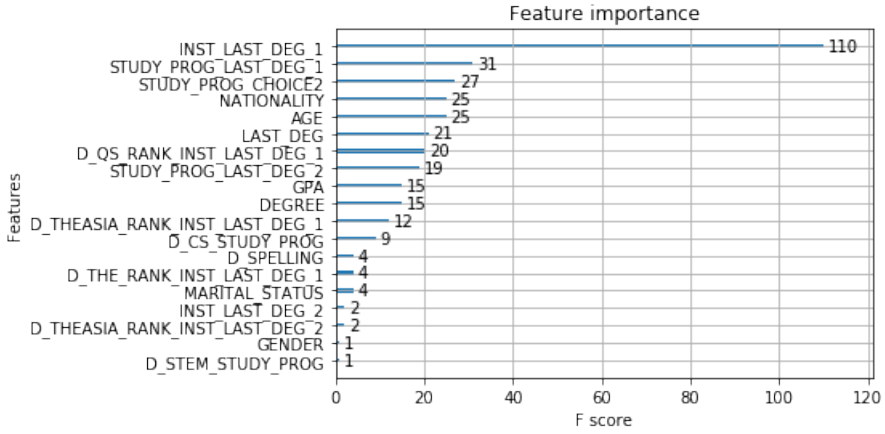


Figure 6.12.: Feature importance of XGBoost trained on strong labelled data.

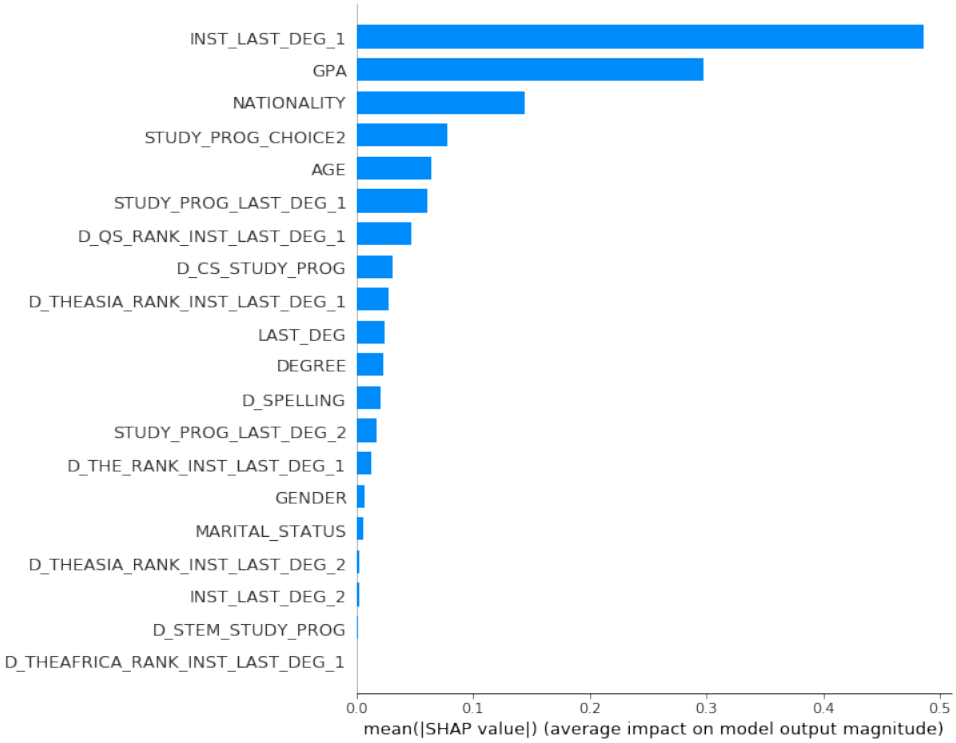


Figure 6.13.: Shapley Values of XGBoost model trained on strong labelled data.

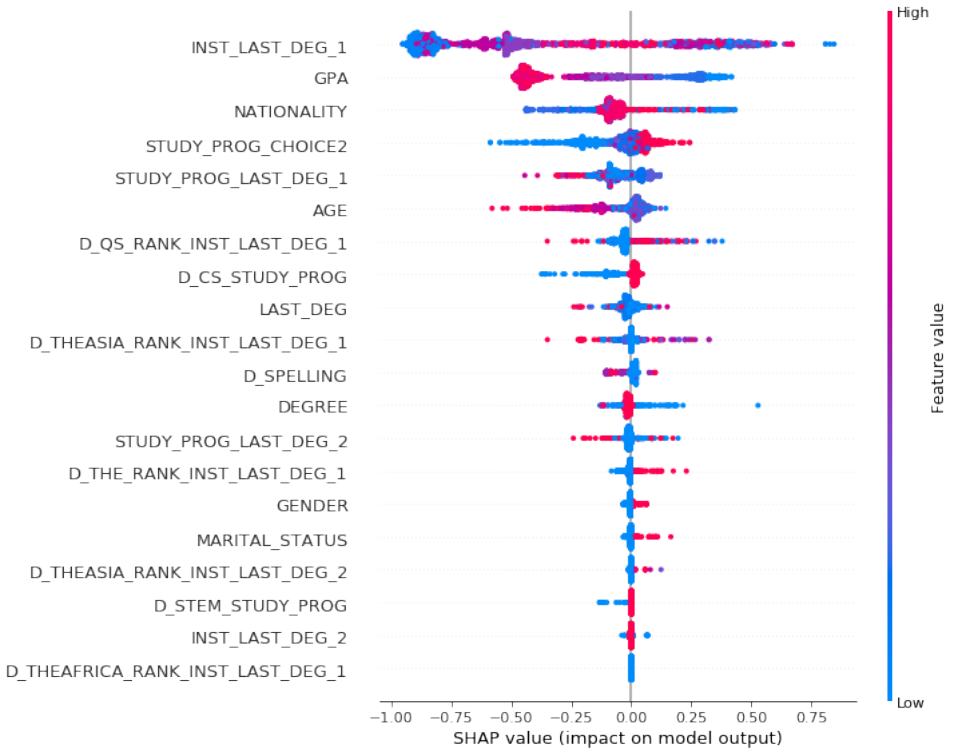


Figure 6.14.: Impact of feature values on the decision. Impact is computed with Shapley values. The XGBoost model was trained on strong labelled data.

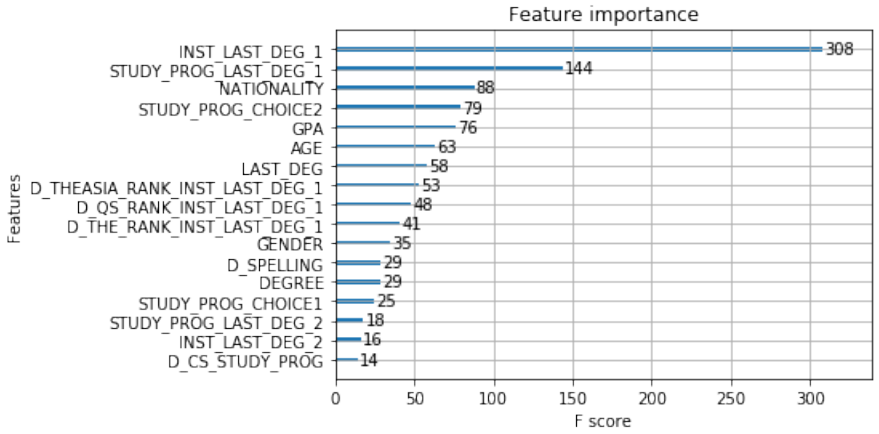


Figure 6.15.: Feature importance of XGBoost trained on weak labelled data.

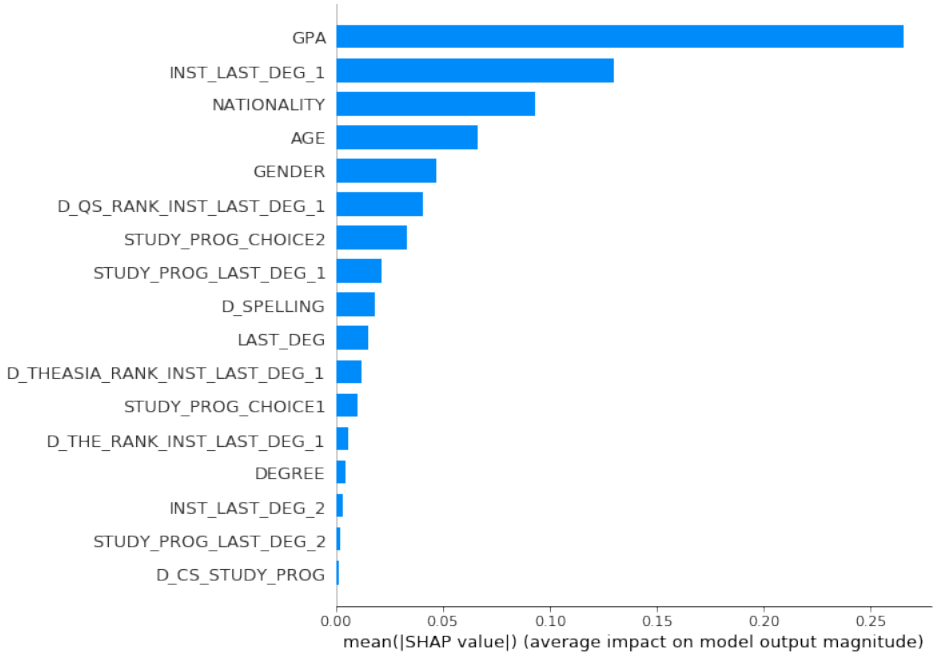


Figure 6.16.: Shapley Values of XGBoost model trained on weak labelled data.

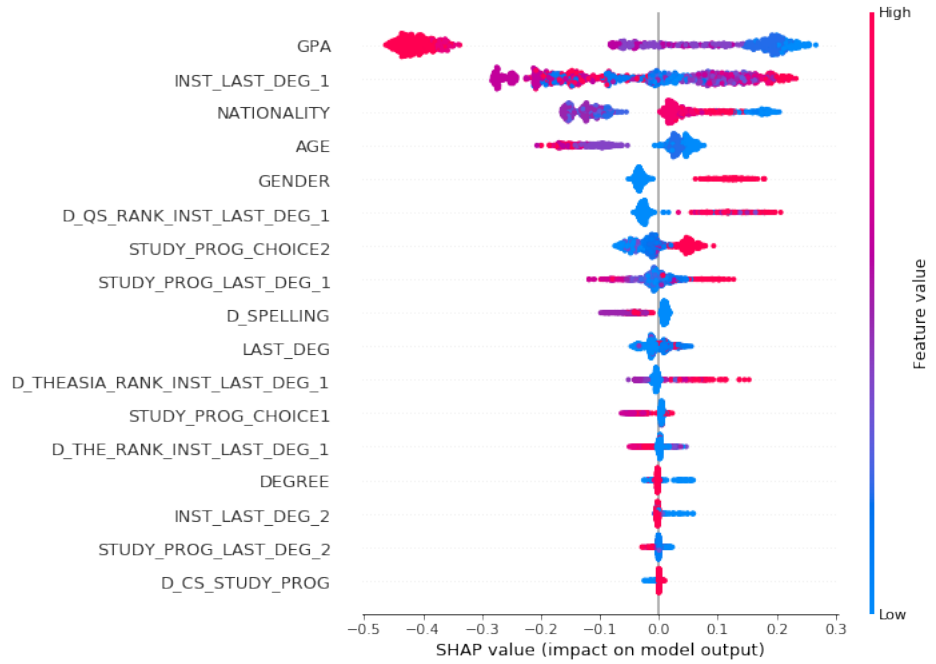


Figure 6.17.: Impact of feature values on the decision. Impact is computed with Shapley values. The XGBoost model was trained on weak labelled data.

7. Conclusion and Future Work

In this thesis all essential steps of a practical machine learning project were processed. The applicant data set was cleaned, analysed and labelled in agreement with the study coordination. In addition, further features were added to the data set, such as rankings or spelling. Finally, models based on decision trees were trained on the processed data and explanatory approaches were developed for their predictions. The models for decision trees, random forests and gradient boosting achieved very good results on the training and test data. All models achieved an accuracy of over 80% and XGBoost an accuracy of over 90%. This is very impressive considering the small amount of data with different characteristics. In most cases, however, grades, university and nationality were most important. However, this also leads to problems, because the decision tree cannot make a reliable prediction for applicants who, for example, have attended a university unknown to the decision tree. This will be analysed in more detail below. In addition, the explainability will be evaluated in more detail and possible improvements regarding the current system will be given, as well as an outlook on how this thesis can be expanded and used meaningfully in the future.

7.1. Quality of the Model

To determine the quality of the models we have trained for the various labelled data, we looked at metrics such as accuracy, precision and recall, but also whether the models are explainable. In the next paragraphs the results of the modelling are discussed and summarised.

7.1.1. Modelling

In the chapter on modelling we have already seen the values for the metrics accuracy, precision, recall and F1 score. The decision trees as well as the models for random forests and gradient boosting achieve high values and clearly exceed the previously defined target of 80%. This is especially due to the addition of negatively labelled students from the data set who never studied at Saarland University. This has significantly improved the prediction for acceptance, but also for rejection. Despite the improvements that labelling has brought about, it must also be viewed critically. For example, if two out of every two students at a university have done poorly at Saarland University, all others will also be labelled negatively, which can have a negative effect on the acceptance of future applicants in the trained models. In order to make such decisions, larger amounts of data should actually be available. However, a larger threshold was not feasible due to the given amount of data. Also, the Wilson lower bound score was used to check how many students are at least correctly classified with a probability of 95%. Good results were also achieved here. The effect of nationality must also be viewed critically. Even

Model	Acceptance Rate
Decision Tree, strong labelled data, 8Best RFE, OS	4871/11853
Random Forests, strong labelled data, 9Best RFE, SMOTE	3154/11853
XGBoost, strong labelled data, 20Best RFE, OS	3675/11853
Decision Tree, weak labelled data, 14Best RFE, OS	5495/11853
Random Forests, weak labelled data, 8Best RFE, OS	5412/11853
XGBoost, weak labelled data, 17Best RFE, US	5668/11853

Table 7.1.: Acceptance rates of the best models on the complete data set.

if some countries have lower educational standards than Germany, only the university should be considered. However, more data would also have to be available for this. In most models, gender is not considered. This ensures that there is no discrimination. In the models that consider gender, any disadvantage due to gender must be taken into account. In several smaller experiments, the omission of nationality and gender was tested. Good results were also achieved with this. The models were slightly worse than the presented ones. However, in these cases the nationality is missing as additional information if the university is unknown. There are also other problems. Since the training data sets are very small and some features are missing, the models cannot reliably predict acceptance for some applicants. If essential information is missing, such as the university, subject, grades or nationality, it is almost impossible to predict the admission. Also, even if a previously unknown university is listed in the rankings, a prediction might not be reasonable. In these cases, one should at least point out that the decision of the decision tree should be taken with caution. The models were also applied to the data which included all applications, i.e., the data which were not labelled too. It turned out that all models would accept significantly more applicants than the commission of Saarland University. Detailed statistics can be found in table 7.1. The problems mentioned above also explain the deviation of the values in the table from the Wilson lower bound score to the commission. Since the data that has been added is sometimes very different from the data on which the models have been tested, the Wilson lower bound score is no longer completely valid for these models. Also, it must also be considered that due to the anonymisation of data in the thesis it is more difficult to analyse the models completely. For example, it is not possible to check which university exactly has a negative influence on a decision because it is encrypted.

7.1.2. Explainability

The decision trees offer the possibility to easily follow the decisions regarding the admission. In particular, the limited depth and the possibility to visualise the decision trees makes this easy. The decision trees trained in this thesis only use the most important features chosen by feature selection. In addition, the feature importance and the Shapley values were specified for more complex models. The Shapley values are more reliable than the feature importance, and indicate the importance of the different features. The visualisations, which show which values have a positive or

negative influence on the Shapley values, make it easier to understand the acceptance or rejection.

7.1.3. Other

The thesis developed an interesting method to standardise the university names, especially in the field of data cleaning. This was a prerequisite to use the data for the further steps in the machine learning project. In particular, the use of a Wikipedia will enable the commission in the future to have quick access to general information of the university. Through feature engineering, further interesting features such as rankings and spelling could be introduced. These have also improved the quality of the model to a certain extent. However, the Asia and Africa rankings are also subject to criticism. Since these do not focus on computer science, but on the university as a whole, it is assumed that a good university also has a good computer science faculty. It would make sense to work with computer science rankings for these regions in the future.

7.2. Deployment Recommendation

In this thesis some things have come to light which should generally be improved. These mainly concern the application process and the maintenance of the information system. These are listed below as recommendations for action.

7.2.1. Adaptation of the Application Form

Probably the most important recommendation for action is to improve the application form. The application form has very few drop-down menus. This leads to unclean data. For example, the university, the subject or the language test should be selected from such a menu. This would ensure that students enter the same names for the same university or subject. Thereby, the extremely time-consuming data cleaning would be omitted in the future. In addition, it should only be possible to send the application form if it is completely filled out - mandatory fields should therefore be introduced. With this restriction null values would be avoided. This would make sense especially for the language tests. One could ensure that only common tests and the corresponding scores can be uploaded. These steps would result in a complete and clean data set and would allow for better work with this data in the future.

7.2.2. Maintenance of the Database

It is very important that the databases are handled correctly and maintained accordingly. Database providers often offer the possibility to create maintenance plans to optimise and secure the database and to detect possible inconsistencies [56]. Such a maintenance plan should definitely be created, and triggers could be built in to maintain consistency. There are some inconsistencies in the database in which the applicant data is stored.

Furthermore, data is not updated regularly. For example, many applicants write in the application form that their language certificate is not yet available. However, this status is no longer updated in the database. Actually, all students who submit their certificate should have it entered in the database and all others should be assigned a uniform value indicating the absence of the certificate.

7.3. Future Work

The models presented in this thesis provide good results for applicants whose university, nationality and subject is known. However, if some features have not been included in the training data set before, the models do not perform very well. Especially when considering the acceptance rate of the trained models on the entire applicant data set, it is very high compared to reality. It is unlikely that the selection committee, is so wrong. The problem is much more likely that the trained models do not know the universities, for example, and therefore make the decision based on other features. The rankings are of limited help here. In the future one could try to reduce the acceptance rate and thus make it more realistic. To do so, one would have to increase the training data set, among other things. One approach would be to label all students who have been rejected negatively. However, this would lead to students who were rejected by the selection committee but could have passed the course being classified incorrectly. In such a case only the decision of the commission would be trained. However, this would not solve the problem. In addition, one should also try to remove the nationality and the gender from the training data in future to avoid discrimination against applicants. Regarding the explainability, we have good possibilities to justify the decisions of the models by visualisations and by Shapley values. The decision trees in particular meet our requirements for explainable AI. The decisions are easy to understand and can also be understood by persons not familiar with computer science. If the future models that are not based on decision trees are to be used, further explanatory approaches can be considered. For example, a ceteris paribus analysis could be used to see how a value would have to change in order to change the decision if all other values remained the same [3]. Also, one should check the use of neural networks. Neural networks show a very good performance in practice, but they are black box models. It would be conceivable to combine decision trees and neural networks. This has been proven in practice and could lead to a better performance [74]. It is certainly useful to use a tool that displays similar students with their performance for applicants. This would already help to get a first overview and be able to initially judge applicants better. In general, the topic still offers a lot of scope for new ideas and corresponding opportunities for further optimisation.

Bibliography

[1] Alexander Armbruster. Computer bringt sich selbst Go bei – und wird Weltklasse. https://www.faz.net/aktuell/wirtschaft/kuenstliche-intelligenz/computer-bringt-sich-selbst-go-bei-und-wird-weltklasse-15253783.html, 2017. Accessed: 22.08.2020.

[2] Microsoft Azure. Bing Websuche. https://azure.microsoft.com/de-de/services/cognitive-services/bing-web-search-api/. Accessed: 12.04.2020.

[3] Hubert Baniecki and Przemyslaw Biecek. The Grammar of Interactive Explanatory Model Analysis. Preprint, https://arxiv.org/abs/2005.00497, 2020.

[4] Przemyslaw Biecek and Tomasz Burzykowsk. 7 Break-down Plots for Additive Attributions. https://pbiecek.github.io/ema/breakDown.html. Accessed: 11.05.2020.

[5] Przemyslaw Biecek and Tomasz Burzykowsk. 8 Break-down Plots for Interactions (iBreak-down Plots). https://pbiecek.github.io/ema/iBreakDown.html. Accessed: 11.05.2020.

[6] Przemyslaw Biecek and Tomasz Burzykowsk. Explanatory Model Analysis, 9 Shapley Additive Explanations (SHAP) and Average Variable Attributions. https://pbiecek.github.io/ema/shapley.html. Accessed: 11.05.2020.

[7] Aleksey Bilogur. Missingno: a missing data visualization suite. *Journal of Open Source Software*, 3(22):547, 2018. DOI: https://doi.org/10.21105/joss.00547.

[8] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*, volume 2, pages 77–82 vol.2, 1994.

[9] Leo Breiman. Random Forests. In *Machine Learning*, volume 45, pages 5–32, 2001. doi: https://doi.org/10.1023/A:1010933404324.

[10] Leo Breimann. Bagging Predictors. In *Machine Learning*, volume 24, pages 123–140, 1996. DOI: https://doi.org/10.1007/BF00058655.

[11] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *KDD*, pages 785–794. ACM, 2016. DOI: http://doi.acm.org/10.1145/2939672.2939785.

[12] CrowdFlower. 2016 Data Science Report. https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf. Accessed: 12.06.2020.

[13] Rodrigo Fernandes de Mello and Moacir Antonelli Ponti. Machine learning. pages 1–10. Springer International Publishing AG, part of Springer Nature, 2018. DOI: https://doi.org/10.1007/978-3-319-94989-5_1.

[14] XGBoost Developers. Introduction to Boosted Trees. https://xgboost.readthedocs.io/en/latest/tutorials/model.html. Accessed: 17.08.2020.

[15] Defense Advanced Research Projects Agency (DARPA) Dr. Matt Turek. Explainable Artificial Intelligence (xai). https://www.darpa.mil/program/explainable-artificial-intelligence. Accessed: 12.04.2020.

[16] Times Higher Education. THE Africa University Rankings 2019. https://www.timeshighereducation.com/student/best-universities/best-universities-africa. Accessed: 12.12.2019.

[17] Times Higher Education. THE Asia University Rankings 2019. https://www.timeshighereducation.com/world-university-rankings/2019/regional-ranking#!/page/0/length/25/sort_by/rank/sort_order/asc/cols/stats. Accessed: 12.12.2019.

[18] Times Higher Education. THE World University Rankings by Subject: Computer Science. https://www.timeshighereducation.com/world-university-rankings/2020/subject-ranking/computer-science#!/page/0/length/25/sort_by/rank/sort_order/asc/cols/stats. Accessed: 12.12.2019.

[19] Omar Elgabry. The ultimate Guide to Data Cleaning. https://towardsdatascience.com/the-ultimate-guide-to-data-cleaning-3969843991d4. Accessed: 15.04.2020.

[20] Dr. Dirk Hecker et al. Zunkunftsmarkt Künstliche Intelligenz Potenziale und Anwendungen. http://publica.fraunhofer.de/dokumente/N-497661.html, 2018. Accessed: 22.08.2020.

[21] Michael Waskom et al. seaborn, (Python Library), Version 0.9.0. https://seaborn.pydata.org/. Accessed: 19.12.2019.

[22] Pauli Virtanen et al. Scipy, (Python Library), Version 1.4.1. https://www.scipy.org/. Accessed: 20.12.2019.

[23] Scott M Lundberg et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nat Biomed Eng 2*, page 749–760, 2018. DOI: https://doi.org/10.1038/s41551-018-0304-0.

[24] Sebastian Bank et al. Graphviz, (Python Library),Version 0.13.2. https://pypi.org/project/graphviz/. Accessed: 13.04.2020.

[25] Tom Augsburger et al. pandas, Version: 1.0.3. https://pandas.pydata.org/about/team.html. Accessed: 11.04.2020.

[26] Tom Fawcett. Introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 06 2006. DOI: https://doi.org/10.1016/j.patrec.2005.10.010.

[27] Python Software Foundation. 7.4. difflib — Helpers for computing deltas. Preprint, https://docs.python.org/2/library/difflib.html#module-difflib. Accessed: 14.08.2020.

[28] D. Oliveira G. Lemaitre, F. Nogueira and C. Aridas. User Guide, Imbalanced-learn. https://imbalanced-learn.readthedocs.io/en/stable/user_guide.html. Accessed: 17.08.2020.

[29] Sampath Kumar Gajawada. Chi-Square Test for Feature Selection in Machine Learning. https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223. Accessed: 13.04.2020.

[30] Jonathan Goldsmith. Wikipedia, (Python Library), Version 1.4.0. https://pypi.org/project/wikipedia/. Accessed: 12.12.2019.

[31] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning*, 46:389–422, 01 2002. DOI: 10.1023/A:1012487302797.

[32] P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.

[33] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, chapter 9.2 Tree Based Methods, pages 307–317. Springer, 2017. DOI: 10.1007/b94608.

[34] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, chapter 15. Random Forests, pages 587–604. Springer, 2017.

[35] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, chapter 15. Random Forests, pages 587–589. Springer, 2017.

[36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, chapter 10. Boosting and Additive Trees, pages 337–358. Springer, 2017. DOI: 10.1007/b94608.

[37] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, chapter 10. Boosting and Additive Trees, pages 338–340. Springer, 2017. DOI: 10.1007/b94608.

[38] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, chapter 10. Boosting and Additive Trees, pages 353–369. Springer, 2017. DOI: 10.1007/b94608.

[39] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, chapter 7.10 Cross-Validation, pages 241–249. Springer, 2017. DOI: 10.1007/b94608.

[40] Andreas Holzinger. Explainable ai (ex-ai). *Informatik Spektrum*, 41(2):138–143, 2018. DOI: 10.1007/s00287-018-1102-5.

[41] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[42] Octopus Data Inc. Octoparse. `https://www.octoparse.com/`. Accessed: 12.12.2019.

[43] Wikimedia Foundation Inc. Wikidata. `https://www.wikidata.org/wiki/Wikidata:Main_Page`. Accessed: 15.08.2020.

[44] Sebastian Kalinowski. pydot, (Python Library),Version 1.4.1. `https://pypi.org/project/pydot/`. Accessed: 13.04.2020.

[45] Alexander Kowarik and Matthias Templ. Imputation with the R package VIM. *Journal of Statistical Software*, 74(7):1–16, 2016. DOI: 10.18637/jss.v074.i07.

[46] Victor Lavrenko and Dr Nigel Goddard. *Lecture Slides Introductory Applied Machine Learning*, chapter Thinking about Data, pages 1–12. School of Informatics, The University of Edinburgh, 2018. Youtube playlist of previous lectures: `https://www.youtube.com/c/VictorLavrenko/playlists?view=50&sort=dd&shelf_id=10`, Accessed: 11.05.2020.

[47] Victor Lavrenko and Dr Nigel Goddard. *Lecture Slides Introductory Applied Machine Learning*, chapter Decision Trees, pages 7–10. School of Informatics, The University of Edinburgh, 2018. Youtube playlist of previous lectures: `https://www.youtube.com/c/VictorLavrenko/playlists?view=50&sort=dd&shelf_id=10`, Accessed: 11.05.2020.

[48] Victor Lavrenko and Dr Nigel Goddard. *Lecture Slides Introductory Applied Machine Learning*, chapter Decision Trees, pages 1–20. School of Informatics, The University of Edinburgh, 2018. Youtube playlist of previous lectures: `https://www.youtube.com/c/VictorLavrenko/playlists?view=50&sort=dd&shelf_id=10`, Accessed: 11.05.2020.

[49] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[50] QS Quacquarelli Symonds Limited. QS World University Rankings by Subject: Computer Science. `https://www.topuniversities.com/subject-rankings/2019`. Accessed: 12.12.2019.

[51] Scott Lundberg. Interpretable Machine Learning with XGBoost. `https://towardsdatascience.com/interpretable-machine-learning-with-xgboost-9ec80d148d27`, 2018. Accessed: 04.05.2020.

[52] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[53] Aditya Kumar Medium.com. Wilson Lower bound Score and Bayesian Approximation for K star scale rating to Rate products. https://medium.com/tech-that-works/wilson-lower-bound-score-and-bayesian-approximation-for-k-star-scale-rating-to-rate-products-c67ec6e30060. Accessed: 14.07.2020.

[54] Microsoft. The Team Data Science Process lifecycle. `https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle`. Accessed: 05.04.2020.

[55] Bartosz Mikulski. Wilson score in Python - example. https://www.mikulskibartosz.name/wilson-score-in-python-example/. Accessed: 17.08.2020.

[56] Mircosoft. Maintenance Plans. `https://docs.microsoft.com/de-de/sql/relational-databases/maintenance-plans/maintenance-plans?view=sql-server-ver15`. Accessed: 13.04.2020.

[57] Christoph Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*, chapter 4.4 Decision Tree. `https://christophm.github.io/interpretable-ml-book/tree.html`. Accessed: 11.05.2020.

[58] Christoph Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*, chapter 5.9 Shapley Values. `https://christophm.github.io/interpretable-ml-book/shapley.html`. Accessed: 11.05.2020.

[59] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. `https://scikit-learn.org/stable/`.

[61] Andrey Petrov. urllib3 (Python Library), Version 1.25.7. `https://pypi.org/project/urllib3/`. Accessed: 12.12.2019.

[62] Erhard Rahm and Hong Do. Data Cleaning: Problems and Current Approaches. *IEEE Data Eng. Bull.*, 23:3–13, 01 2000.

[63] Muhammad Summair Raza and Usman Qamar. *Understanding and Using Rough Set Based Feature Selection: Concepts, Techniques and Applications*, chapter Introduction to Feature Selection, pages 1–25. Springer, Singapore, 2019.

[64] Kenneth Reitz. Requests, (Python Library), Version 2.22.0. `https://pypi.org/project/requests/`. Accessed: 12.12.2019.

[65] Jeffrey Dastin (Thomson Reuters). Amazon scraps secret AI recruiting tool that showed bias against women. `https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G`. Accessed: 25.04.2020.

[66] Scikit-learn. 1.10. Decision Trees. https://scikit-learn.org/stable/modules/tree.html#tree. Accessed: 10.04.2020.

[67] Scikit-learn. 3.1. Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html. Accessed: 12.04.2020.

[68] Scikit-learn. *Feature importance evaluation*, chapter 1.11. Ensemble methods. https://scikit-learn.org/stable/modules/ensemble.html#random-forest-feature-importance. Accessed: 25.08.2020.

[69] Scikit-learn. Post pruning decision trees with cost complexity pruning. https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html#sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py. Accessed: 13.04.2020.

[70] Scikit-learn. Precision-Recall. https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html. Accessed: 13.04.2020.

[71] Scikit-learn. sklearn.tree.decisiontreeclassifier. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.feature_importances_. Accessed: 10.05.2020.

[72] Scikit-learn. sklearn.tree.DecisionTreeClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.get_depth. Accessed: 20.04.2020.

[73] Raymond Sheh and Isaac Monteath. Defining Explainable AI for Requirements Analysis. *KI - Künstliche Intelligenz*, 32, 10 2018. DOI: 10.1007/s13218-018-0559-3.

[74] Eyal Shulman and Lior Wolf. Meta Decision Trees for Explainable Recommendation Systems. pages 365–371, 02 2020. DOI: 10.1145/3375627.3375876.

[75] Koo Ping Shung. Accuracy, Precision, Recall or F1. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. Accessed: 13.04.2020.

[76] V. S. Spelmen and R. Porkodi. A review on handling imbalanced data. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–11, 2018.

[77] Rajebhosale Supriya and Prof. Shilpa Gite. A Survey on Data Mining Based POI Recommendation System Using Geo Tagged Images. *International Journal of Innovative Research in Computer and Communication Engineering*, 3:12271–12275, 12 2015. DOI: 10.15680/IJIRCCE.2015. 0312037.

[78] Manohar Swamynathan. *Mastering Machine Learning with Python in Six Steps*, chapter Step 2: Introduction to Machine Learning, pages 82–84. Apress, Berkeley, CA, 2019. DOI: https://doi.org/10.1007/978-1-4842-4947-5.

[79] Manohar Swamynathan. *Mastering Machine Learning with Python in Six Steps*, chapter Step 4: Model Diagnosis and Tuning, pages 279–307. Apress, Berkeley, CA, 2019. DOI: https://doi.org/10.1007/978-1-4842-4947-5.

[80] Manohar Swamynathan. *Mastering Machine Learning with Python in Six Steps*, chapter Step 2: Introduction to Machine Learning, page 90. Apress, Berkeley, CA, 2019. DOI: https://doi.org/10.1007/978-1-4842-4947-5.

[81] Manohar Swamynathan. *Mastering Machine Learning with Python in Six Steps*, chapter Step 3: Fundamentals of Machine Learning, page 150. Apress, Berkeley, CA, 2019. DOI: https://doi.org/10.1007/978-1-4842-4947-5.

[82] Saarland University. Online Application for Ph.D./M.Sc. Programs. https://oas.cs.uni-saarland.de/index.php?authorsInstructions=1. Accessed: 28.04.2020.

[83] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[84] Sean Wallis. Binomial Confidence Intervals and Contingency Tests: Mathematical Fundamentals and the Evaluation of Alternative Methods. *Journal of Quantitative Linguistics*, 20:178–208, 07 2013.

[85] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. DOI: 10.25080/Majora-92bf1922-00a.

Appendices

A. Appendix Stuff

Package	Version
NumPy [59, 83]	1.18.3
Matplotlib [41]	2.2.3
missingno [7]	0.4.2
pandas [25, 85]	1.0.3
Requests [64]	2.22.0
seaborn [21]	0.9.0
urllib3 [61]	1.25.7
wikipedia [30]	1.4.0

Table A.1.: Packages used in chapter 3

Package	Version
NumPy [59, 83]	1.18.3
Matplotlib [41]	2.2.3
pandas [25, 85]	1.0.3
Requests [64]	2.22.0
seaborn [21]	0.9.0
urllib3 [61]	1.25.7
wikipedia [30]	1.4.0

Table A.2.: Packages used in chapter 4

Package	Version
imbalanced-learn [49]	0.6.2
NumPy [59, 83]	1.18.3
Matplotlib [41]	2.2.3
pandas [25, 85]	1.0.3
seaborn [21]	0.9.0
scikit-learn [60]	0.22.1
scipy [22]	1.4.1
xgboost [11]	1.0.2

Table A.3.: Packages used in chapter 5

Package	Version
Graphviz [24]	0.13.2
imbalanced-learn [49]	0.6.2
NumPy [59, 83]	1.18.3
Matplotlib [41]	2.2.3
pandas [25, 85]	1.0.3
pydot [44]	1.4.1
seaborn [21]	0.9.0
shap [52]	0.35.0
scikit-learn [60]	0.22.1
scipy [22]	1.4.1
xgboost [11]	1.0.2

Table A.4.: Packages used in chapter 6