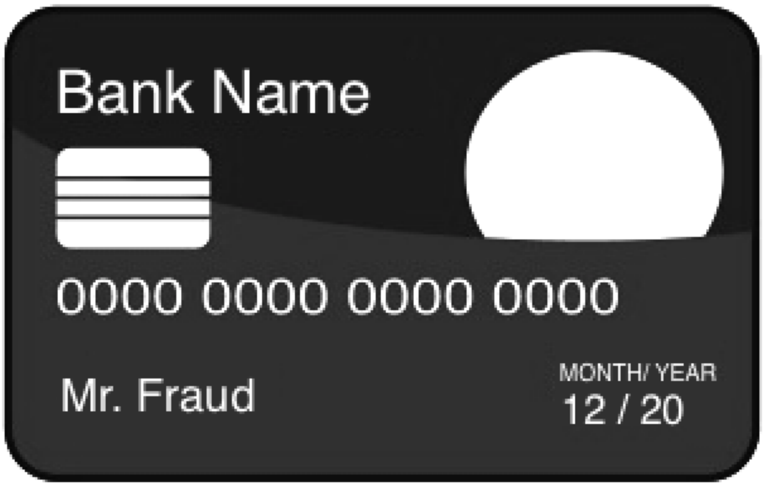
Umeå University

Department of Statistics

One Year Master Thesis, 15 Credits

CREDIT CARD FRAUD DETECTION (Machine learning algorithms)



Student:
Fredrik Westerlund

Supervisor:
Priyantha Wijayatunga

VT 2017

Abstract

Credit card fraud is a field with perpetrators performing illegal actions that may affect other individuals or companies negatively. For instance, a criminal can steal credit card information from an account holder and then conduct fraudulent transactions. The activities are a potential contributory factor to how illegal organizations such as terrorists and drug traffickers support themselves financially. Within the machine learning area, there are several methods that possess the ability to detect credit card fraud transactions; supervised learning and unsupervised learning algorithms. This essay investigates the supervised approach, where two algorithms (Hellinger Distance Decision Tree (HDDT) and Random Forest) are evaluated on a real life dataset of 284,807 transactions. Under those circumstances, the main purpose is to develop a "well-functioning" model with a reasonable capacity to categorize transactions as fraudulent or legit. As the data is heavily unbalanced, reducing the false-positive rate is also an important part when conducting research in the chosen area. In conclusion, evaluated algorithms present a fairly similar outcome, where both models have the capability to distinguish the classes from each other. However, the Random Forest approach has a better performance than HDDT in all measures of interest.

Keywords: Credit card fraud detection, Machine learning, Supervised learning algorithms, Classification, Unbalanced data.

Sammanfattning

Titel: Kreditkortsbedrägeri med användning av maskininlärningsalgoritmer

Kreditkortsbedrägeri är ett område med kriminella aktörer som utför olagliga gärningar vars påverkan har en negativ effekt på andra individer eller företag. Det kan till exempel ske genom att en förövare stjäl kreditkortsuppgifter i syfte att förskingra pengar från innehavarens bankkonto. För illegala organisationer som terrorister och narkotikahandlare är dessa gärningar möjligtvis ett sätt att ekonomiskt livnära sig på. Inom statistiken finns det flera maskininlärningsalgoritmer som har kapacitet att upptäcka bedrägliga transaktioner, dvs "supervised learning" och "unsupervised learning" metoder. I följande uppsats beaktas den förstnämnda där två algoritmer (Hellinger Distance Decision Tree (HDDT) och Random Forest) utvärderas på en datamängd med 284.807 verkliga transaktioner. Under dessa förutsättningar är det huvudsakliga syftet att träna upp en "väl fungerande" modell med god kapacitet att kunna skilja mellan legitima och bedrägliga transaktioner. Reducera andelen falska positiva är också en väsentlig del eftersom datamängden är väldigt obalanserad i denna typ av forskningsområde. Sammanfattningsvis visar de utvärderade värdena att modellerna har liknande resultat, där båda har potential att kategorisera klasserna. Random Forest presenterar dock en bättre klassificeringsförmåga än HDDT.

Populärvetenskaplig sammanfattnig

Kreditkortsbedrägeri är ett område med kriminella aktörer som utför olagliga gärningar vars påverkan har en negativ effekt på andra individer eller företag. Det kan till exempel ske genom att en förövare stjäl kreditkortsuppgifter i syfte att förskingra pengar från innehavarens bankkonto. För illegala organisationer som terrorister och narkotikahandlare är dessa gärningar möjligtvis ett sätt att ekonomiskt livnära sig på. Inom statistiken finns det ett flertal metoder som har potential att upptäcka och eventuellt bekämpa bedrägeri. I följande uppsats utvärderas två modeller på en datamängd med verkliga transaktioner genomförda av europeiska individer. Under dessa förutsättningar är det huvudsakliga syftet att kunna särskilja mellan legitima och icke-legitima tranasktioner på bästa möjliga sätt. Sammanfattnigsvis visar studien att de utvalda modellerna har liknande resultat, men ena presterar lite bättre än den andra.

Acknowledgement

First of all, I would like to thank my supervisor Priyantha for his support and knowledge within supervised learning algorithms. Secondly, the statistical department at Umeå University for letting me write a One Year Master thesis even though I was working full time at a different location. Lastly, it has been an interesting and busy period of my life, therefore I also would like to thank my family and friends for their motivation support during the writing.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The first chapter in this essay consists of three sections. Firstly, a walkthrough of various credit card fraud techniques. It is followed by previous work within detection of fraudulent transactions using machine learning. Lastly, the main purpose of the thesis will be presented.

1.1 Credit card fraud background

Credit card fraud is a field with criminals performing illegal acts that may affect other individuals or companies negatively (Quah & Sriganesh 2008, 1722). It contains numerous techniques that are constantly being used by perpetrators in order to conduct fraudulent transactions. More specifically, gaining access to credit cards from other individuals may occur through physical contact or via the internet (Masoumeh, Seeja & M.Afshar 2012, 35). In the first situation, a criminal must steal the credit card from an account holder to execute a fraud. For instance, skimming, where perpetrators use card readers to capture data from magnetic stripes (i.e. a victim physically swipes the card in a store which then gives lawbreakers the ability to re-encode new credit cards with that information). Hence, a potential victim may not be aware of what has happened to his or her bank account before it is too late.

Not to mention, the second course of action described above contains sensitive information[1] that are exposed through emails, conversations and other applications. These leaks give the perpetrators the opportunity to purchase various items over the internet with other people's credit card information.

Chaudhary, Yadav & Mallic (2012, 39-40) further highlights additional techniques such as bankruptcy fraud and application fraud. To demonstrate an example from the first one, criminals must put themselves in personal bankruptcy, e.g. uses a credit card to order several merchandises with the knowledge that they will not be able to pay back (Delamaire, Abdou & Pointon 2009, 59-60). Under those circumstances, it could cause a significant impact on banks since they have to recover the losses by themselves. In a similar perspective, the application fraud happens when an individual provides false documents during the application process for a new credit card (Jha & Westland 2013, 374).

[1]Sensitive information: Credit card holder, credit card number, expiration date and CVV code.

Despite all illegal approaches mentioned so far, there are still multiple of other fraud systems with similar characteristics, but those will not be presented in this paper. Generally speaking, they all hold one thing in common, i.e. criminals seek to withdraw money that does not belong to them. Mahmoundi & Duman (2015, 2510) describe credit card fraud as an immense problem that costs billions of dollars every year. On that note, it could cause extensive consequences throughout the world. Fraudulent transactions may be a potential contributory factor to how illegal organizations as terrorists and drug traffickers support themselves financially (Bhattacharyya et al. 2011, 602).

1.2 Detecting frauds using machine learning

Detection of fraudulent transactions combined with machine learning has become an exciting subject over the past years (Correa Bahnsen et al. 2016, 134). The main purpose is to gather information from historical data and then draw conclusions by advanced analysis. In further detail, there are supervised learning and unsupervised learning methods available for this particular problem (Carnerio, Figueira & Costa 2017, 3). These approaches contain statistical models that have the capacity to classify transactions as fraudulent or legitimate. As a result of a supervised problem, detection models investigate previously known behaviors and patterns to determine the outcome of new observations (Dal Pozzolo et al. 2014, 4916). Therefore, each feature variable in a dataset must have one labeled response variable (fraud or legit). In that case, one could view it as an input-output relationship where statistical models use both independent variables (input) and a dependent variable (output) for classification (Geurts, Irrthum & Wehenkel 2099, 1594). Unlike the supervised learning algorithms, the unsupervised learning procedure does not use a specific class; they only exploit feature variables to find similar groups or patterns that are unknown (Jha & Westland 2013, 376).

Statistical methods frequently used in the field of credit card fraud detection are K-Means clustering, Support Vector Machine (SVM), K-Nearest Neighbor (KNN) and Genetic algorithms (Chaudhary, Yadav & Mallic 2012, 40). All very powerful and possesses the ability to distinguish the dependent class; fraudulent and genuine. Nonetheless, using Neural network (NN) is also an appropriate model, which has the capacity to recognize patterns over time. Masoumeh, Seeja & M.Afshar (2012, 35) describes NN as the most popular machine learning algorithm when detecting fraudulent transactions due to its advantage over time (i.e. it adapts from previous behaviors and improve its results). However, in the recent year's Random Forest has become more interesting since it also can be viewed as an effective method (Jha & Westland 2013, 376).

The implementation of machine learning on credit card usage has its advantages and disadvantages. Due to a greatly increased number of transactions over the past few years, the credit frauds have also gained an increasing trend (Correa Bahnsen et al. 2016, 134). Thus, databases are now storing a huge amount of information about whether transactions are fraudulent or legit. Jha

& Westland (2013, 373) explains that a general problem with these two classes is the massive imbalance between them. In other words, learning from previously recorded patterns may cause major complications, since most models do not handle the skewness well. Batista, Carvalho & Monard (2000, 316) advocate that supervised learning methods are not adapting accurately when classes are unbalanced, and classification scores are often good for the majority class but the minority is intolerable. Therefore, applying machine learning algorithms to credit card transactions could lead to a large proportion of false positive[2] predictions. Correa Bahnsen et al. (2016, 134-135) describes it as a cost-sensitive problem, where financial institutions have to pay an administrative price for each false positive. Consequently, it would be convenient to have as few as possible during the classification stage in order to obtain an accurate credit card fraud detection model. At the same time, resampling techniques are usually implemented to handle the skewness (Dal Pozzolo et al. 2014, 4915). This could additionally lower the amount of false positive values and increase the effectiveness. Bhattacharyya et al. (2011, 603) mention that the most common strategies are random undersampling of the majority group and random oversampling of the minority group. But there exist other approaches as well, a different splitting criterion may also improve the classification performance (Cieslak et al. 2012, 137-138).

An additional problem with detection of credit card fraud transactions using machine learning is the time factor, i.e. criminals are constantly adapting their strategies to avoid trouble. This could lead to poor prediction accuracy with machine learning algorithms categorized as supervised learning since they rely on previously known behaviors. On that note, over a longer time period, trained models may have a disadvantage on credit card usage due to potential undetected frauds. The development of detection techniques must expand faster than criminals fraud actions (Carnerio, Figueira & Costa 2017, 1; Mahmoudi & Dunman 2014, 2510). In this case, there are various of unsupervised learning methods that have the capability to investigate new behaviors and patterns (Dal Pozzolo et al. 2014, 4916).

1.3 Purpose

The intention of this essay is to classify credit card transactions as fraudulent or genuine, where supervised learning algorithms are used. Therefore, each individual transaction in the provided dataset is already assigned to one of the known classes (fraud or legit). Under those circumstances, the purpose is to develop a "well-functioning" model based on known behaviors[3], and as new transactions occur (training-set/test-set), the algorithm will either classify them as genuine or fraudulent. However, the data is highly unbalanced and could cause some significant trouble. With that said, the machine learning algorithms used are Random Forest and Hellinger Distance Decision Tree (HDDT).

[2]False positive: predictions that classify as fraudulent but they are genuine.

[3]Known behaviors: feature variables that can explain credit card fraud in some way.

Chapter 2

Theory

The second chapter begins with a brief introduction to the general concept of decision trees. It is followed by a theoretical documentation of each machine learning algorithm used during the classification stage, where their proceedings are presented. Lastly, a section describing the evaluation technique is found.

2.1 Decision tree (Basic intro)

In the field of statistics there exist both regression and classification decision trees (Biau & Scornet 2016, 200). These are simple to utilize and have good interpretability in various problems. More importantly, the value of the target variable has an essential role, within regression there is a continuous response and categorical for classification. Since this paper has a binary outcome, the latter method will be our main focus. Song & Lu (2015, 131) describes the structure as a hierarchical pattern, where there are several components linked together by numerous branches, i.e. a single root node, internal nodes and leaf nodes. The root is always stationed at the beginning (top) of a decision tree, and all other nodes are connected to it in some way. Furthermore, the internal nodes have one entering node and two or more subsets of directions (Rokach & Maimon 2007, 8). Lastly, all leaf nodes are located at the bottom and specify the predicted outcomes of the target variable.

Additionally, three other vital parts that determine the structure of a decision tree is splitting, pruning and stopping (Song & Lu 2015, 131). The variables used when splitting a node is usually based on importance, where the most important one is referred as the parent node (root). In other words, nodes are divided based on the relationship between feature variables and the target variable. Moreover, a tree continues to grow with different variables (or same) until a certain stop criterion is fulfilled. A stopping process avoids wide and complex decision tree structures, which could cause lack of robustness due to few observations in each leaf if the tree is large. Therefore, the number of cases in a leaf or the size of a tree are two common stopping criterions (Rokach & Maimon 2007, 9). Those implementations prevent a tree to increase its format. On the other hand, in the pruning process we start by growing a large decision tree that has no stopping rules at all, then "trim" the structure by removing nodes that do not add any important information (Song & Lu 2015, 132). It means that we go from a large hierarchical structure to a smaller one.

An alternative option to demonstrate a decision tree is to see it as multiple paths of conditions that classify certain outcomes. For instance, trees could be translated into IF-THEN expressions (Masoumeh, Seeja & M.Afshar 2012, 37). If condition 1 and condition 2 are true, then event k occurs (i.e. a hierarchical shape is created by combining all rules together). This results in a tree structure based on various "questions" that explains the target variable. For illustration purpose, imagine a transaction dataset with three feature variables (V1, V2 and V3) and one binary response variable (fraud, legit). Figure 2.1 presents a single decision tree and its process during the classification stage. Our initial step is the question "is V1 less than 200?", if the answer is yes, there will be follow-up question "is V2 greater than 4.5?", which results in either a legitimate or fraudulent transaction. On the other side, if the first question is "no", the algorithm will take the third variable into account. Thus, "is V3 larger than 60?", which also results in either of the two outcomes. Hence, the feature variables can take both continuous and discrete values.

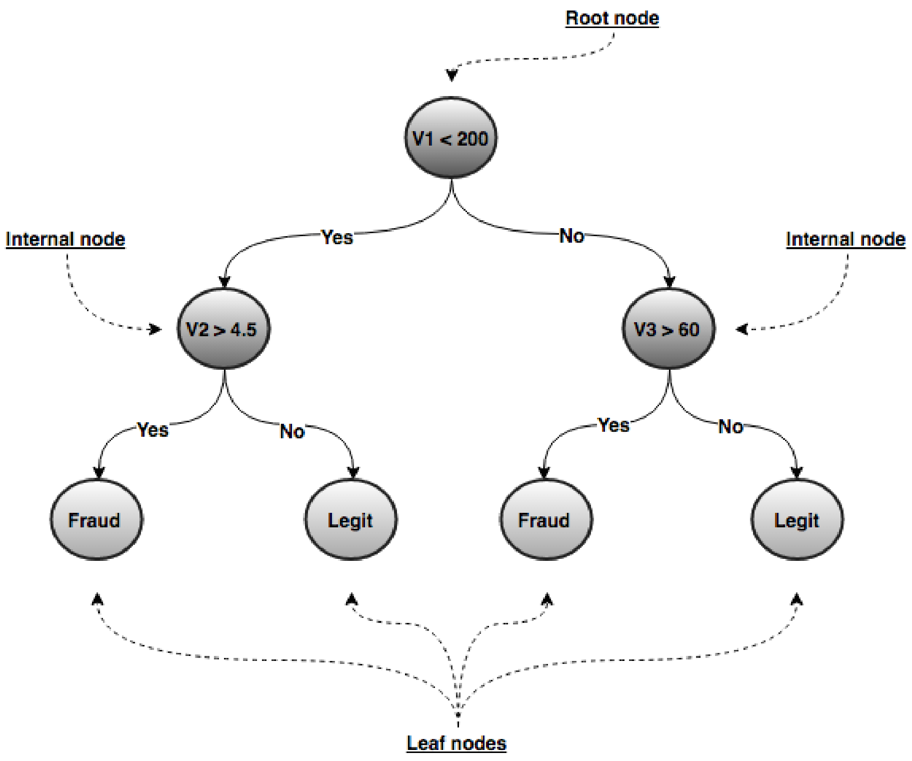


Figure 2.1: An illustration of the standard decision tree procedure, where three feature variables are attempting to classify the binary dependent variable (legit, fraud).

2.2 Random Forest

There are several techniques available to modify a standard decision tree in order to achieve better accuracy within a classification problem. For instance, Bagging and Random Forest are models with more robust prediction capability (James et al. 2013, 316). The intention is to reduce the variance by using more advanced statistical methods, e.g. a single decision tree suffers from high variance (Hastie, Tibshirani & Friedman 2008, 312). In other words, when fitting a standard tree on numerous evaluation sets (training/test), a quite different result could occur. Instead of taking one individual decision tree into account, we may therefore improve the precision by evaluating multiple grown independent trees and then let them cast a majority vote (Breiman 2001, 5). This could be viewed as an ensemble of trees that predicts a certain outcome.

The Random Forest machine learner algorithm mentioned above was originally developed by the famous statistician Leo Breiman (2001, 5-32). As a result of that, it is now a popular tree-based method that is extensively used in various fields due to its simple interpretability. From high-dimensional spaces to smaller samples are areas where it has shown great success (Biau & Scornet 2016, 198). It consists of an ensemble of unpruned randomly grown trees (i.e. de-correlated). Thus, the model does not overfit due to the law of large number, and therefore it may be seen as a power model in prediction (Breiman 2001, 29).

Further details, both Random Forest and Bagging are using bootstrapping techniques to produce a large number of decision trees from the original (training) dataset (Prasad, Iverson & Liaw 2006, 184). These algorithms are very similar to each other, where the primary difference is the construction of nodes. In the Bagging approach, all p feature variables in the selected sample are used to determine the optimal split within each node (Liaw & Wiener 2002, 18). This usually leads to a high correlation between all developed decision trees, and for that reason, almost all look identical. However, in Random Forest the splitting process for an individual node is based on a randomly chosen set of m feature variables of p total (Bohm & Zech 2010, 326). In that case, the most influenced one within the subset is selected to represent the node split (both continuous and discrete variables are allowed). Normally $m = \sqrt{p}$ with a minimum of one node throughout the construction process (Hastie, Tibshirani & Friedman 2008, 592). By creating a hierarchical structure on this condition there will exist numerous decision trees in the bootstrap samples that deviates considerably from each other. James et al. (2013, 321-320) suggest that an alteration like this could provide extra randomness to the model and bypass the highly correlated trees which occur in Bagging. Keep in mind that, if the selection of m feature variables are equal to p, the outcome will remain the same as Bagging (Svetnik et al. 2003, 1948).

Additionally, in contrast to a standard decision tree (section 2.1), there is no pruning in the Random Forest machine learner algorithm. Each individual grown tree among all bootstrap samples are generated to the maximum size (Bohm & Zech 2014, 383).

In event of a classification problem, new observations are classified by evaluating the class votes of all bootstrap samples (Hastie, Tibshirani & Friedman 2008, 592). Therefore, all leafs in each individual grown decision tree consist of one class (e.g. fraud or genuine), and the most predicted class will be the representative for the *b*th tree. By then taking the majority vote of the entire forest (Tree-1,..,Tree-b) we can determine the final outcome (Breiman 2001, 6). A picture of this process is displayed in Figure 2.2, where numerous trees are grown to the fullest maximum size.

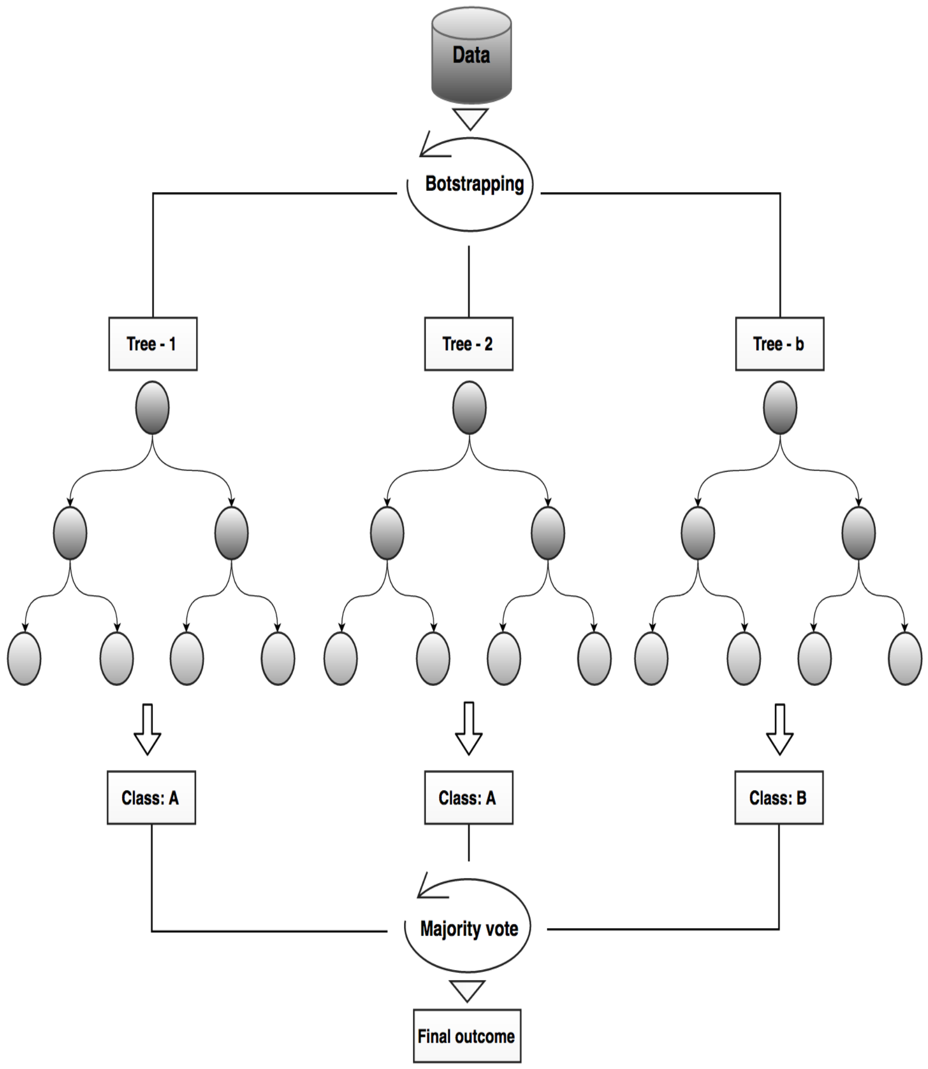


Figure 2.2: An illustration of the Random Forest approach, where multiple trees are assembled with bootstrapping using the original data.

2.2.1 Variable importance

Creating a classification algorithm with multiple bootstrap samples and several variables is one thing, but determine which feature variables that are most relevant for the classifier is another task. Therefore, it is critical to identify the variables that are best suited for our trained model in order to get a better insight of the chosen subject (Bireman 2001, 23). An ensemble of trees have both advantages and disadvantages in terms of interpretability, for instance, we can not examine individual trees separately (Liaw, Iverson and Prasad 2006, 185). In that case, the Random Forest method could be viewed more as a "black box" approach. Thus, the prediction accuracy may increase but has to pay in interpretability (James et al. 2013, 319). On the other end, there are numerous metrics that supports the understanding of evaluating bagging techniques.

The variable importance (VI) measure has the ability to locate and determine which feature variables that hold the largest impact on the prediction scores, but also reveal the non-influential ones. A high VI value is an indication of an important variable that is frequently occurring during the splitting process of nodes when training a classification algorithm (James et al. 2013, 319). Whereas the low ones tend to have a poor relationship and could probably be left out from further investigation, since they most likely do not bring any important information.

More importantly, the Random Forest model implemented in this essay is based on the R-package *randomForest*. It uses the Gini Index to determine the importance of variables throughout the splitting process (explanation is presented underneath). James et al (2012, 312) mention that it may also be specified as a measure of node "purity", i.e. a node that consists of observations from mainly one class is an indication of a small value.

Haste, Tibshirani & Friedman (2008, 309) begins the formula by establishing the proportion of class k observation within a node q as

$$\hat{v}_{qk} = \tfrac{1}{N_q} \textstyle\sum_{x_i \in R_q} I(y_i = k),$$

where R_q represents a nodes region with the number of observations refereed as N_q. Consequently, the Gini Index arithmetic differentiates between a regression and a classification problem. As we are handling a binary outcome, metrics associated with the last mentioned one will only be explained. Thus, let the formulation be defined by

$$Purity(q) = \textstyle\sum_{k=1}^{K} \hat{v}_{qk}(1 - \hat{v}_{qk}),$$

which calculates the total variance among all K classes (Haste, Tibshirani & Friedman 2008, 310). If the $\hat{v}_{qk}$ term is near zero or one it will generate a low "purity" value. Hence, for bagged trees, we determine the overall importance of each individual feature by adding the total amount that the Gini Index has decreased within splits over a given feature, which is then averaged across all trees (James et al. 2013, 319).

2.2.2 Classification algorithm

This section presents a more mathematically explanation of the Random Forest machine learning algorithm. As mentioned earlier, it is a collection of B trees created by bootstrap samples from the original data (training). But let us first have a walkthrough of two important formulas that will further make it easier to understand the classification algorithm underneath. Thus, the set of all developed decision trees (an ensemble of classifiers) may together be expressed as follow

$$\{T_b\}_1^B = \{T_1(X), ..., T_b(X)\},$$

where $b = (1, 2, ..., B$: size of the forest) and for each individual tree a independent identically distributed random vector X is created (Brieman 2001, 2; Svetnik et al. 2003, 1948). Consequently, by producing multiple decision trees, there will exist numerous outcomes that are predicting a certain class. Important to realize, all constructed trees and their final outputs can then be established as

$$\{\hat{C}_b\}_1^B = \{\hat{C}_1 = T_1(X), ..., \hat{C}_b = T_b(X)\},$$

(Svetnik et al. 2003, 1948). The ensemble are then generating one final estimation, i.e. calculating the majority vote of all predicted classes. Hence, as the above terminology hopefully provides a better understanding, one can now continue to the final proceedings of the Random Forest approach (Algorithm A). This set up is developed by Hastie, Tibshirani & Friedman (2008, 592).

```
Algorithm A: Classification with Random Forest
1. For all bootstrap samples (decision trees), i.e. $T_1$, $T_2$,...,$T_b$ do:
    (a) Using the training data, draw a bootstrap sample of size $N$.
    (b) Given the drawn bootstrap sample, grow a random forest tree $T_b$
        until the minimum node size $n_{min}$ is reached (i.e. no further splits
        are possible). For construction of each individual node there are three
        conditions:
        (I) Among all the $p$ feature variables in the dataset, randomly
            select $m$ of them ($m = \sqrt{p}$).
        (II) Find the optimal variable within $m$.
        (III) Separate the individual node into two other daughter nodes
              based on the best chosen variable in the previous step.
2. Review the collection of trees $\{T_b\}_1^B$.
3. Prediction process at a new point $X$:
    Let $\hat{C}_b$ be our estimated class outcome of the $b$th tree. The new predicted
    value can then be achieved by taking the majority vote of all random for-
    est tree outcomes, $\hat{C}_{rf}^B$ = majority vote of $\{\hat{C}_b\}_1^B$.
```

2.3 Hellinger Distance Decison Tree (HDDT)

Imbalance of classes in a dataset combined with basic machine learning algorithms may often lead to poor classification performance, since the minority class is usually ignored (Rokach 2016, 115). Thus, one of the major issues with standard decision tree methods on unbalanced data is the splitting criterion, which often refers to the Gini Measure and Information Gain (Kang and Ramamohanarao 2014, 213). These two measurements are heavily skew sensitive[1] and have a large disadvantage in terms of classification performance when operating on minority groups. Hence, to capture the smaller class in unevenly distributed datasets, more advanced models and techniques must be taken into account.

As a demonstration of the last sentence above, standard statistical algorithms usually combine sampling routines in order to increase the predictability. Thus, in case of large datasets, one could apply undersampling strategies to reduce the amount of observations from the majority group. Although this may lead to a more balanced distribution of the classes, it might also result in loss of important information (Chawla 2010, 879). For instance, the method may eliminate potentially critical data that could improve the performance. On the other hand, oversampling techniques have the advantage to replicate observations from the minority group and then make the classes more equal (often used on small datasets). But again, there are consequences, statistical models are more keen to overfitting by this approach (Chawla 2010, 879). Therefore, both benefits and drawbacks arise when implementing additional techniques to enhance the predictability.

Consequently, a fairly recent developed model that takes a different splitting criterion into account has proven to be skew insensitive, i.e. the Hellinger distance decision tree (HDDT) algorithm (Cieslak and Chaw 2008, 138). For imbalanced data it was recorded that an increased classification performance can be accomplished without the implementation of sampling procedures. The method uses the Hellinger Distance terminology throughout the decision tree making process in order to capture divergence in distributions. Additionally, in the same fashion as the Random Forest design, there is no pruning within the HDDT algorithm (a single tree is grown to the fullest maximum size). This is due to the fact that, if trees were pruned, leafs with few observations would be eliminated and are most likely the ones associated with the minority class (Dal Pozollo 2015, 109). Hence, an essential part when working with unbalanced data is to maintain the tree as a whole. Because unpruned trees are capable of finding more splits in the dataset and further differentiate the class of interest, i.e. an algorithm has a greater chance of discovering more unusual splits. In matter fact, the HDDT has shown to outperform models like C4.4 in terms of deeper trees with more leafs (Cieslak et al. 2012, 151).

All things considered, the fundamental theory behind this algorithm is, com-

[1] Skew sensitive: bias toward the majority group, i.e. the class priors has an influence and the smaller class will usually be ignored (unbalanced problem).

pared to others, no sampling techniques and another splitting criterion (in event of unbalanced problems). Moreover, the HDDT method has also in related work shown to be competitive with favorable algorithms such as C4.5 in terms of predictive accuracy, time efficiency and etc (Dal Pozoollo 2015, 115).

2.3.1 Hellinger Distance implementation

In considerations of distributional divergence, the Hellinger distance is a non-negative and symmetric measurement that is used to quantify the affinity among two probability distributions (Lyon, Brooke and Knowles 2014, 1971). As the theory behind the measurement is suggested to be skew insensitive, it is further implemented as a splitting criterion within decision tree construction (Cieslak et al. 2012, 138). A low Hellinger distance value implies that the given distributions are close to each other. Because of that, while splitting nodes one strive to maximize the distance between the two probability distributions (i.e. minimal affinity). The formula has close roots to the Bhatacharyaa coefficient (BC), and one could acquire the Hellinger distance by taking advantage of its terminology. Cieslak et al. (2012, 139) describes it as, let (Ω, L, s) denote a measure space, and Q the collection of all probability values on L (under the condition that they remain absolutely continuous with respect to s). Consequently, the BC between two probability measures are determined by

$$BC = p(Q_1, Q_2) = \int_{\Omega} \sqrt{\frac{dQ_1}{ds} \cdot \frac{dQ_2}{ds}} ds, \tag{2.1}$$

where $Q_1, Q_2 \in Q$. The 2.1 formula is then implemented in the Helliner Distance computation as

$$h_H(Q_1, Q_2) = \sqrt{2\Big[1 - BC\Big]} = \sqrt{\int_{\Omega} \left(\sqrt{\frac{dQ_1}{ds}} - \sqrt{\frac{dQ_2}{ds}}\right)^2 ds}. \tag{2.2}$$

Moreover, a countable space is expected when using equation 2.2 as a decision tree splitting criterion within a binary classification problem (Cieslak and Chawla 2008, 243). Instead of comparing continuous functions, conditional probabilities from discrete data are desired. For example, P(X = x | C = c), where c is taken from a limited set of classes such as + or -, while the x term comes from a limited set of attribute values T {low, medium, high} (Cieslak et al. 2012, 139). An important notation to remember is that all continuous feature variables are discretized into bins or partitions (i.e. a number of splits are examined and the collection of such values may then be expressed as {down, up}). This allow us to make further modifications on the Hellinger distance formula, for instance, we may now convert the integral into a summation of all values and rewrite the equation as

$$d_H(P(C_+), P(C_-)) = \sqrt{\sum_{i \in T} \left(\sqrt{P(X_i|C_+)} - \sqrt{P(X_i|C_-)}\right)^2}, \tag{2.3}$$

(Cieslak et al. 2012, 139). This formulation (2.3) is assumed to be highly skew insensitive, and enable us to calculate the affinity between a binary class for discrete data. More importantly, the Hellinger Distance splitting criterion has three major properties within decision tree splitting; non-negative, symmetric and bounded in a finite interval (Cieslak et al. 2012, 139). A broader description of these characteristics can be examined in Table 2.1.

Table 2.1: Properties of the Hellinger distance splitting criterion.

Property	**Definition**
$d_H(P(C_+), P(C_-)) \geq 0$	Non-negative
$d_H(P(C_+), P(C_-)) = d_H(P(C_-), P(C_+))$	Symmetric
$d_H(P(C_+), P(C_-)) \in [0, \sqrt{2}]$	Bounded

2.3.2 Classification algorithm

The HDDT approach is further explained underneath in two combined algorithms, where both procedures are practicing on a training set referred as Z and feature value f (Cieslak et al. 2012, 143-144). Keep in mind, for all continuous feature variables, a slight adjustment of Binary_Hellinger is implemented, i.e. it sorts in terms of the feature value and assesses all relevant splits, then return the greatest Hellinger distance that has been recorded among all individual splits. The Z_i term in Algorithm B1 specifies the subset of training observations coming from Z which contains all class i occurrences (Cieslak & Chawla 2008, 7). Additionally, $Z_{x_k=j}$ denotes a subset that contains the value j for feature k. Lastly, $Z_{k,j,i}$ defines the subset that consists of class i with value j for the feature variable k.

```
Algorithm B1: Binary_Hellinger
1. Start by letting Hellinger ← −1
2. Let T_f be a set of values of feature f
3. begin for each value $t \in T_f$ do following
4.     Let $p \leftarrow T_f \setminus t$
5.     HD_value ←$(\sqrt{|Z_{f,t,+}|/|Z_+|} - \sqrt{|Z_{f,t,-}|/|Z_-|})^2 + (\sqrt{|Z_{f,p,+}|/|Z_+|}$
                  $- \sqrt{|Z_{f,p,-}|/|Z_-|})^2$
6.     if HD_value is larger than Hellinger then
7.         Set Hellinger ← HD_value
8.     end if
9. end for
10. return $\sqrt{\text{Hellinger}}$
```

Altogether, the final proceeding is then summarised in Algorithm B2, where C specifies a fixed cut-of-size and n denotes a tree node.

```
Algorithm B2: HDDT
1. if $|Z|$ is less than $C$ then
2.     return
3. end if
4. $n \leftarrow argmax_f Binary\_Hellinger(Z, f)$
5. begin for each value t of $b$ do following
6.     construct $n´$, i.e. a child node of $n$
7.     HDDT $(Z_{x_b=t}, C, n´)$
8. end for
```

2.4 Evaluation technique

In regards to evaluation technique, there are many approaches available to determine a statistical models predictability, e.g. generalization error, ROC curve, f-measure, area under curve (AUC) and a confusion matrix (Rokach & Maimon 2007, 21-35). The key measurements in the present study uses the last-mentioned one, which is simple to interpret and explain in binary classification problems for supervised learning.

A confusion matrix (error matrix) consists of two columns and two rows that together represents true positives, true negatives, false positives and false negatives (Chawla 2010, 876-877). Firstly, all values found in the true positive cell are predicted outcomes matching the actual values in the dataset. In a transaction data, this would denote a prediction that classifies as genuine when the real value is also genuine. Secondly, the true negatives are the exact opposite, i.e. a predicted respectively an actual value is both labeled as fraud. Thirdly, the false positives are recognized as "Type l errors" that stands for outcomes classified as fraudulent but they are actually not. Last and final, the false negatives are "Type ll errors" and represents transactions (frauds) that are predicted as legitimate.

To summarize the confusion matrix specifications, one may see the off-diagonal values as misclassifications and the diagonal as accurately classified outcomes. Hence, Table 2.2 provides a visible illustration of the evaluation design.

Table 2.2: Confusion matrix.

		Predicted		
		True	False	Total
Actual	True	True Positives (TP)	False Negatives (FN)	TP + FN
	False	False Positives (FP)	True Negatives (TN)	FP + TN
	Total	TP + FP	FN + TN	N

More importantly, there are four central evaluation measures in this essay that uses the above terminology; model accuracy (MA), misclassification rate (MCR), false negative rate (FNR) and false positive rate (FPR). Those are relatively straightforward to calculate when the confusion matrix is displayed (see Table 2.3 for a better understanding).

Table 2.3: Description of the evaluation formulas.

Formula	**Definition**
MA = (TP + TN)/(Total)	Rate of correctly classified values (Overall)
MCR = (FP + FN)/(Total)	Rate of inaccurately classified values (Overall)
FNR = (FN)/(FN + TP)	Rate of classifying "actual false" as true
FPR = (FP)/(FP + TN)	Rate of classifying "actual true" as false

Chapter 3

Data

The third chapter describes the provided dataset with descriptive statistics using plots and tables, e.g explanation of feature variables and the distribution of genuine respective fraudulent transactions.

3.1 Descriptive statistics

The provided dataset is based on transactions from European cardholders that have been made during a two-day period in September 2013. It was originally collected by a research collaboration of Wordline and Univeristé Libre de Bruxelles (ULB) with the aim of analyzing big data and fraudulent transactions. In total there were 284,807 transactions throughout the time span, and the dependent class (fraud, legit) is heavily unbalanced. Table 3.1 describes all thirty-one variables in the data, where the feature variables besides Time and Amount are displayed with an unknown description due to a protection of sensitive information. Hence, these are not the original variables obtained during the collection of data. They have all been transformed with principal component analysis (PCA) to protect the true information from the analyst examining the data (or other third parties that may contribute to negative consequences). In other words, V1-V28 are principal components holding the real data in some fashion. All twenty-eight (Vxx) variables and Amount are categorized as numeric, while Class and Time are both integers.

Table 3.1: Definition of variables in the provided dataset.

Variable	**Type**	**Description**
Class	int	Response variable (1 = Fraudulent and 0 = Legitimate)
Time	int	Time between each transaction
Amount	num	Total money spent
V1	num	Feature variable with unknown information
.	.	.
.	.	.
V28	num	Feature variable with unknown information

Figure 3.1 presents an illustration of the heavily unbalanced dataset. There is certainly a tremendous difference between the fraudulent and legitimate class, where 492 transactions are assigned as frauds. That accounts for 0.172 % of all transactions made during the two-day period in September 2013. Consequently, a relationship plot (Figure 3.2) is also presented on next page, which gives an interpretation of the correlation between all feature variables and the response. In this particular plot V16, V14 and V12 are the ones that registered the largest value. On that note, this could add some helpful information during the training phase of an algorithm, since variables with higher correlation may have a stronger link to the dependent variable.

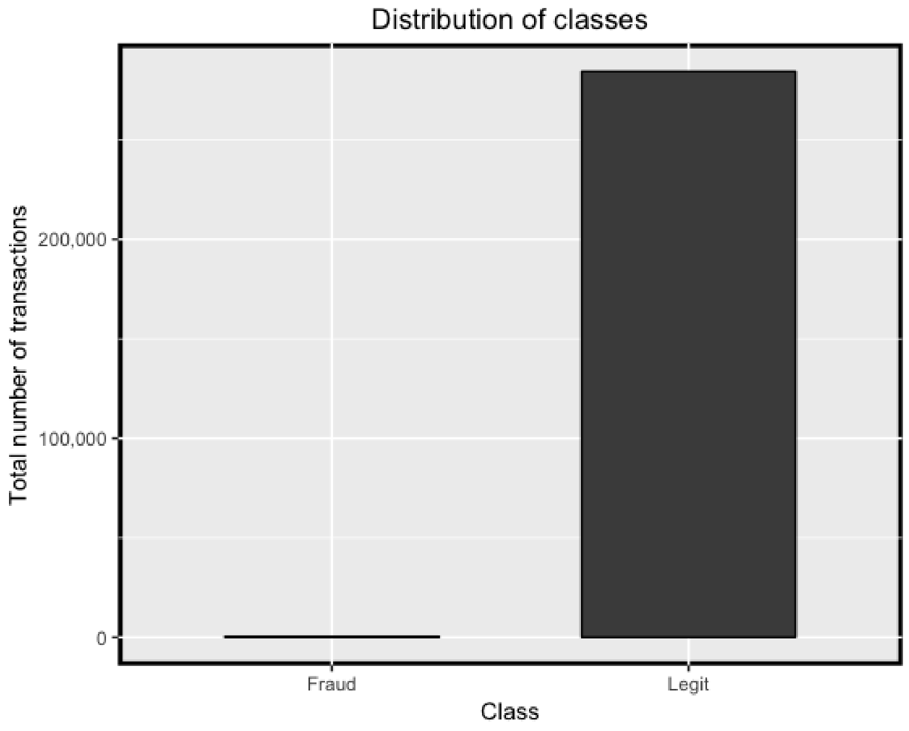


Figure 3.1: Following bar chart represents the class distribution of transactions made by European cardholders.

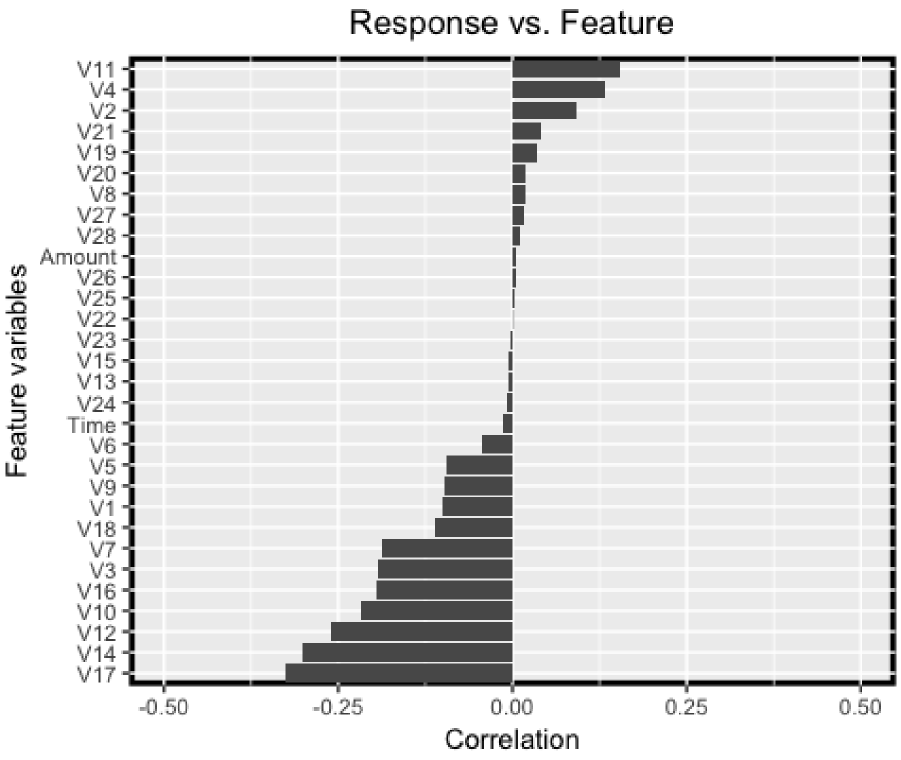


Figure 3.2: A plot presenting the correlations between all feature variables and the response. The bar charts on the left side represents a negative correlation, whilst the opposite side illustrates a positive correlation.

To further investigate the difference between the two classes of interest (fraud respective genuine), four feature variables were selected for comparison (the other ones displayed similar trends). In Figure 3.3 one can observe that V14 and V17 possess shapes that distinguish the classes from each other. Thus, data like this would be incredibly powerful to have while training an algorithm, since these feature variables are most likely the best-suited choice for separation of fraudulent and legitimate transactions. On the other hand, V13 respectively V26 are virtually indistinguishable, which may lead to the opposite result (i.e. hard interpretability in credit card fraud detection due to their distribution similarities).

Moreover, these inequalities and similarities are also summarized with numbers in Table 3.2, where the mean, standard deviation, minimum value and maximum value are computed for both the fraudulent and legitimate class. It is indeed a huge contrast when comparing values, e.g. V17, the genuine class has a mean of 0.01152 and the fraud group recorded a value of -6.666.

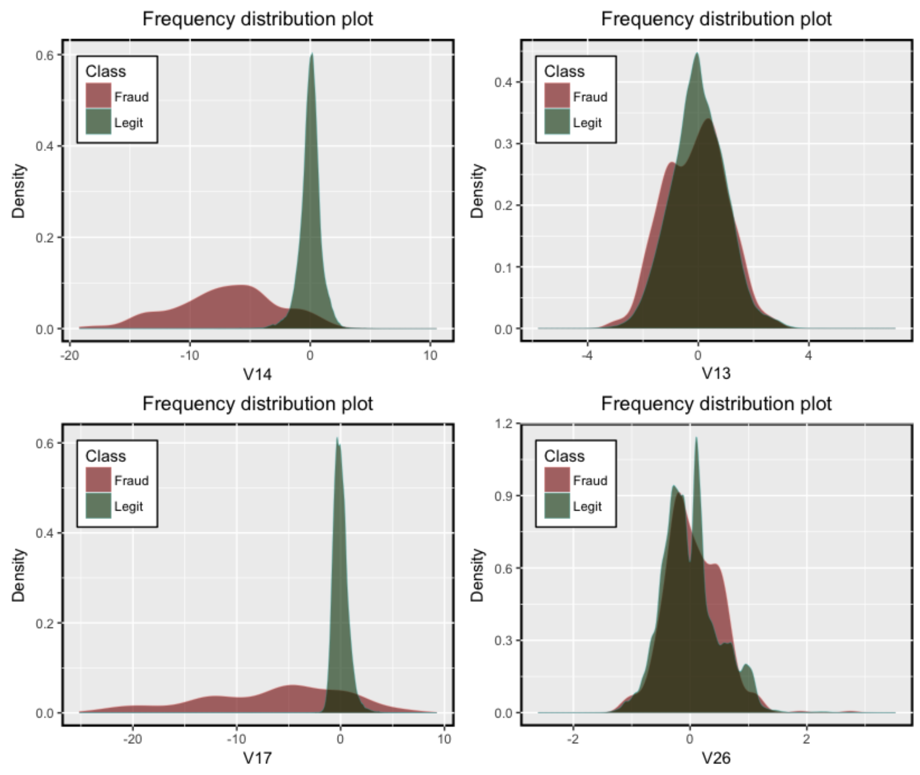


Figure 3.3: Multiple density plots showing the distributions of classes over four selected feature variables.

Table 3.2: Descriptive statistics on the selected variables.

Variable	Mean	Standard deviation	Min Value	Max Value
V13 (Legit)	0.00018	0.99506	-5.7920	7.12700
V13 (Fraud)	-0.10930	1.10451	-3.12800	2.81500
V14 (Legit)	0.01206	0.89700	-18.3900	10.5300
V14 (Fraud)	-6.972	4.27894	-19.210	3.442
V17 (Legit)	0.01153	0.74945	-17.100	9.25400
V17 (Fraud)	-6.666	6.97061	-25.160	6.739
V26 (Legit)	-0.00009	0.48224	-2.6050	3.51700
V26 (Fraud)	0.05165	0.47167	-1.15300	2.74500

Chapter 4

Results

The fourth chapter contains all results developed by the two algorithms, i.e. software/design, confusion matrix outcomes, measures of interest and variable importance plots. First Random Forest and then followed by HDDT.

4.1 Random forest performance

Results from the Random Forest algorithm are computed with the R-Studio software, where the default settings were used (an ensemble of 500 bootstrap trees). The original data was split into three separately distinct sets in order to estimate the classes (Fraudulent and legitimate), i.e. 60/40, 70/30 and 80/20 ratio. All feature variables in the data were involved during the training phase.

The predicted events are summarized in Table 4.1 and reveal the confusion matrix outcomes. One can observe that the majority of estimated transactions are found in the "True-True" cell. Consequently, all evaluation measures are documented in Table 4.2. The MA formula recorded values close to one, which on the other hand is an indication of low MCR scores. Same goes for the FNR value. However, there are three measurements (FPR) that registered a much larger outcome; 25.38 %, 17.56 % and 11.22 %.

Table 4.1: The Random Forest confusion matrix outcomes.

60/40	**Predicted**			**70/30**	**Predicted**			**80/20**	**Predicted**	
Actual	113721	5		**Actual**	85285	9		**Actual**	56855	8
	50	147			26	122			11	87

Table 4.2: A summary of final evaluation measures.

Formula	**60/40 Split**	**70/30 Split**	**80/20 Split**
MA	0.99951	0.99959	0.99967
MCR	0.00049	0.00041	0.00033
FNR	0.00004	0.00011	0.00014
FPR	0.25381	0.17568	0.11224

Figure 4.1 reveals the importance of each feature variable used throughout the training process, where values between 0 and <125 have been listed. The ones with the largest impact on the Random Forest machine learner algorithm is certainly V17, followed by V12 and V14. Feature variables such as V23, V25 and V24 appears to demonstrate a weak result in regards to the VI evaluation. Additionally, the features with a known description (Time respective Amount) are also presenting a low value.

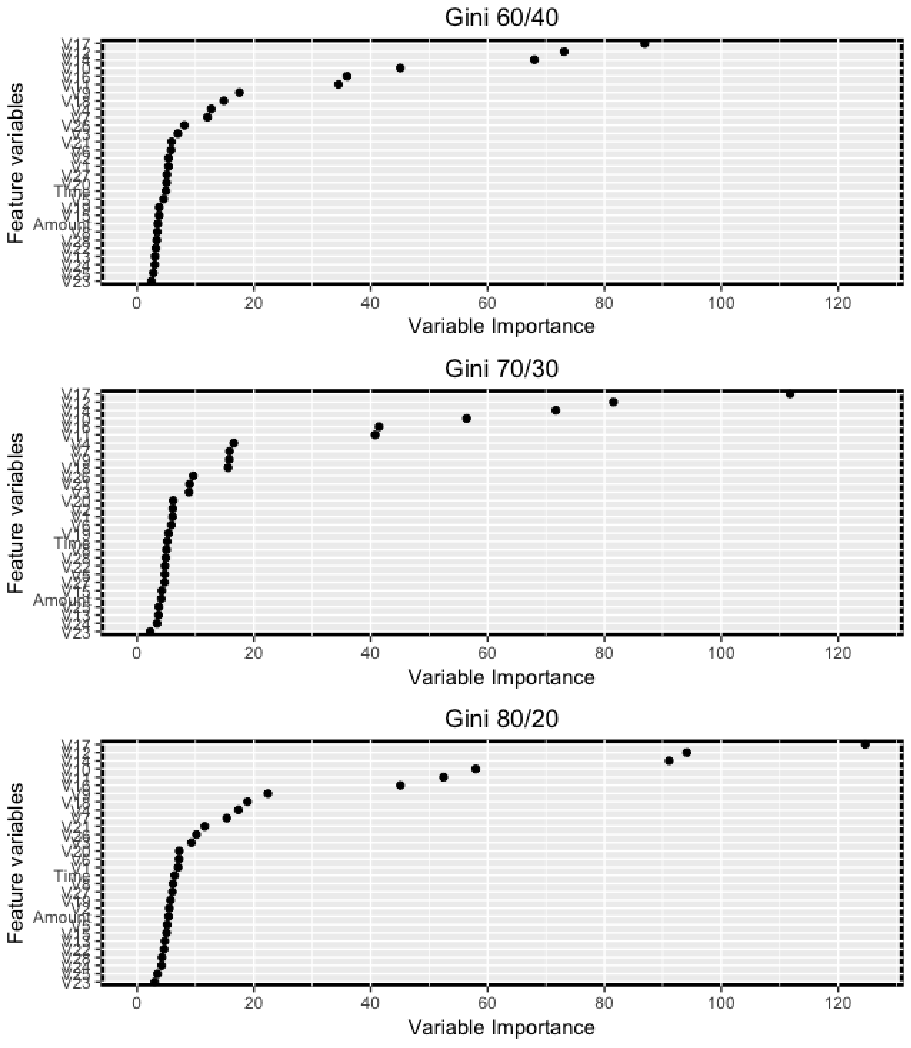


Figure 4.1: Variable importance plots that summarize the influence of each individual feature variable during the splitting process.

4.2 HDDT performance

The values in the upcoming part are all based on the HDDT algorithm using WEKA as the analytical software. First a presentation of the confusion matrix outcomes, and then followed by the final results obtained through calculation of important formulas. Same design as the Random Forest approach was used, i.e. all feature variables are involved throughout the modeling stage and the datasets are also identical (60/40, 70/30 and 80/20 split ratio).

Table 4.3 establishes all predicted values, where the "True Positive" cell has the largest amount of classified outcomes across all three individual separated datasets. Under those circumstances, one could observe that almost every predicted transaction is correctly classified, i.e. TP respective TN, and only a smaller proportion are presenting a negative result (FP and FN).

Table 4.3: The HDDT confusion matrix outcomes.

60/40	**Predicted**			**70/30**	**Predicted**			**80/20**	**Predicted**	
Actual	113688	38		**Actual**	85259	35		**Actual**	56847	16
	53	144			31	117			20	78

Furthermore, Table 4.4 completes the final classification performance of the trained algorithms across all datasets. In regards of MA, the calculated values present a fairly similar outcome (approximatively 99.9 %). Thus, the overall prediction accuracy is incredibly high. Same goes for the MCR and FNR numbers, which demonstrates extremely small values. Nevertheless, there is a slight difference between the FPR measurement, values of 26.90 %, 20.94 % and 20.40 % have been computed.

Table 4.4: A summary of final evaluation measures.

Formula	**60/40 Split**	**70/30 Split**	**80/20 Split**
MA	0.99920	0.99923	0.99937
MCR	0.00080	0.00077	0.00063
FNR	0.00033	0.00042	0.00028
FPR	0.26904	0.20946	0.20408

Chapter 5

Conclusion

The fifth chapter sums up the overall research. Firstly, a review of the calculated results with some reflections on the evaluation design. It is followed by deeper analysis of the feature variables and disadvantages that arise. Lastly, further studies within the field is discussed.

5.1 Discussion

In conclusion, this One Year Master thesis has provided a deeper knowledge within the field of credit card fraud detection using machine learning algorithms. Several techniques used by criminals have been documented, including the underlying terminology behind advanced statistical models that possess the capacity to prevent fraudulent behaviors. As discussed in the purpose section (1.3), a major concern has been the extremely unbalanced data set (Table 3.1). One could assume that achieving 99 % MA is close to a perfectly trained algorithm that will be worth millions for the bank companies. But with seconds thoughts, that is not the correct measure of performance in this problem since almost every transaction made is labeled as genuine. Thus, it will automatically contribute to a very high overall prediction accuracy (using the confusion matrix), and an essential part is not to be fooled by the computed results that MA is presenting. Chawla (2010, 876) has similar thoughts, where the overall accuracy is not the way to go when working with unbalanced data. For that reason, the most important evaluation measurement within this research area is definitely the FPR value, which banks seek to minimize due to a cost-sensitive problem (Correa Bahnsen et al. 2016, 134-135).

By further investigate the estimated confusion matrix outcomes in Chapter 4, it definitely reveals that MA respective MCR holds a quite misleading result if the focus lays on determine fraudulent transactions (i.e. Table 4.2 and Table 4.4). With no doubt, one should target the FPR measurement in order to present a compatible model that can distinguish the classes from each other. Hence, from a trained algorithms perspective, the Random Forest approach appears to outperform HDDT in all formulas. It recorded lower values for all three datasets, and an 80/20 ratio separation had the greatest result (11.22 %).

I also need to clarify one important fact, the reason why I did not implement any sampling techniques on this particular dataset (even though classes are extremely unbalanced) is the significant result that was obtained by the original authors of HDDT (Cieslak et al. 2012, 149). They advocate that a single HDDT has the potential to compete against other statistical methods without the need of sampling, and still improve the performance significantly. Not to mention, equal results were also presented in a different study, which presumed that HDDT operates well without e.g. undersampling (Dal Pozzollo 2012, 116). Thus, in order to compare algorithms of interest in this essay, both approaches were evaluated with same criterions (i.e. 60/40, 70/30 and 80/20 ratio split of the original data) without any sampling methods.

Moreover, one could also discuss which feature variables have the greatest influence on the classification algorithm in order to understand the underlying problem. Figure 4.1 in the result section demonstrates that V17, V14 and V10 are the variables that occur most often during the splitting process of nodes in decision tree making using Random Forest. This is quite similar to the correlation analysis (Figure 3.2), as the feature variables with highest variable importance are also the top performers in terms of correlation. Additionally, looking at the frequency distribution plots in Figure 3.3, this might be the explanation to why these usually ends up in the top. As the classes in V28 and V13 are much identical, it could be difficult to find any important information that differentiates fraudulent and legitimate transactions from each other. Instead, variables such as V14 and V17 presents immense differences, which most likely are the reason why they are leading in all analysis. For evaluation purpose, it would be convenient to gather as much information as possible from these feature variables (or similar variables), since they seem to show an important role in credit card fraud detection using machine learning algorithms on this chosen dataset.

Keep in mind, a disadvantage with supervised learning approaches on fraudulent transactions is the time factor. Since perpetrators are constantly adapting their strategies, trained machine learning algorithms may be useless due to the ineffectiveness of the feature variables over time. Therefore, these variables may be reasonable on this unique dataset, but how they compete in the 2017s market is another question. For me, it is close to impossible to determine if the feature variables used today are the same as then, since the majority of them are described as unknown information (no metadata).

Given the overall documentation of what has been considered so far, the only question left to answer is, do the trained algorithms work in further implementation? is the result in this paper a reflection of a "well-functioning" algorithm for credit card fraud detection? To answer these issues one first has to declare, what is the baseline for a well-operating model in real life data streams. Of course, "zero misclassification rate" is the optimal goal. But is that possible with these feature variables and the selected machine learning algorithms of interest? Obviously, the banks should definitely be concentrating on decreasing the FPR value if the data is extremely unbalanced as in this instance. On the other hand, it is tough to determine whether 11.22 % misclassification rate (Table 4.2) is the optimal and best model for this particular data. Multiple approaches and methods must therefore be tested in order to find out the true “lower limit” of the FPR. Generally speaking, I believe the result achieved is a representation of a functional model that has the ability to distinguish transactions as fraudulent respective genuine. However, further discussions should be made with a person or company that wants an algorithm that can stop fraudulent transactions. If they are satisfied with 11.22 % FPR, then one could draw the conclusion that this is a “well-functioning" model.

In future studies, it would be really interesting to investigate why the HDDT approach did not perform that well. As the algorithm was proposed to be skew-insensitive, one may assume that it should outperform most models that use a skew-sensitive splitting criterion. However, the conducted research has not proven that on this particular data. An important detail that is worth considering, maybe bagged HDDTs can provide a better classification performance. Cieslak et al. (2012, 138) did recommend that an ensemble of HDDTs are also operating well on unbalanced classes. Due to technical issues, it could not be implemented in this essay (I did not find an existing bagged HDDTs algorithm). Therefore, a future task is to examine the effects by using this technique rather than a single decision tree. In addition to that, a dataset with known variables should also be considered.

References

Batista, G., Carvalho, A and Monard, M. 2000. Applying One-Sided Selection to Unbalanced Datasets. *MICAI 2000: Advances in Artificial Intelligence.* 1793(1): 315-325. doi: 10.1007/10720076_29.

Bhattacharyya, S., Jha, S., Tharakunnel, K and Westland, J. 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 50(3): 602 - 613.

Biau, G and Scornet, E. 2016. A random forest guided tour. *TEST.* 25(2): 197 - 227. doi: 10.1007/s11749-016-0481-7.

Bohm, G and Zech, G. 2010. *Introduction to Statistics and Data Analysis for Physicists.* Hamburg: German Electron Synchrotron. http://www-library.desy.de/elbook.html (Accessed 2017-03-22).

Breiman, L. 2001. Random Forests. *Machine Learning.* 45(1): 5 – 32. doi: 10.1023/A:1010933404324.

Carnerio, N., Figueira, G and Costa, M. 2017. A data mining based system for credit-card fraud detection in e-tail. *Decision Support Systems.* 95(1): 91-101. doi: 10.1016/j.dss.2017.01.002

Chaudhary, K., Yadav, J and Mallick, B. 2012. A review of Fraud Detection Techniques: Credit Card. *International Journal of Computer Applications* 45(1): 39 - 44.

Chawla, N. 2010. Data Mining for Imbalanced Datasets: An Overview. In Maimon, O and Rockach L. 2nd ed. *Data Mining and Knowledge Discovery Handbook.* Springer: US. 875 - 886.

Cieslak, D., Hoens, T., Chawla N and Kegelmeyer W. 2012. Hellinger distance decision trees are robust and skew-insensitive. *Data mining and Knowledge Discovery.* 24(1): 136-158.

Cieslak, D and Chawla, N. 2008. Learning Decision Trees for Unbalanced Data. In Daelemans, W and Morik, K. 1st ed. *Machine Learning and Knowledge Discovery in Databases.* Springer: Verlag Berlin Heidelberg. 241 - 256.

Correa Bahnsen, A., Aouanda, D., Stojanovic, A and Ottersten B. 2016. Feature engineering strategies for credit card fraud detection. *Expert Systems With Applications.* 51(1): 134 - 142.

Dal Pozollo, A., Caelen, O., Borgne, Y., Waterschoot, S and Bontempi G. 2014. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications* 41(10): 4915 - 4928.

Dal Pozollo, Andrea. 2015. *Adaptive Machine Learning for Credit Card Fraud Detection.* Ph.D. diss., Univerité Libre De Bruxelles.

Delamaire, L., Hussein, A and Pointon J. 2009. Credit card fraud and detection techniques: a review. *Banks and Banks system* 4(2): 57 - 68.

Geurts, P., Irrthum, A and Wehenkel, L. 2009. Supervised learning with decision tree-based methods in computational and systems biology. *Molecular BioSystems.* 5(12): 1593 – 1605.

Hastie, T., Tibshirani, R and Friedman J. 2008. *The Elements of Statistical Learning.* 2^{nd} ed. Springer: New York.

James, G., Witten, D., Hastie, T and Tibshirani, R. 2013. An Introduction to Statistical Learning: with Application in R. 6^{th} ed. Springer: New York.

Jha, S and Westland, J. 2013. A Dscpriptive Study of Credit Card Fraud Pattern. *Global Business Review* 14(3): 373 - 384.

Kang, S and Ramamohanarao, K. 2014. A Robust Classifier for Imbalanced Datasets. In Tseng, V., Ho, T., Zhou, Z., Chen, A and Kao, H *Advances in Knowledge Discovery and Data Mining.* Springer: Cham. 212 - 223.

Liaw, A and Wiener M. (2002). Classification and Regression by randomForest. *R News.* 2(3): 18 - 22.

Lyon, R., Brooke, J., Knowles, J and Strappers, B. 2014. Hellinger Distance Trees for Imbalanced Streams. *2014 22nd International Conference on Pat tern Recognition.* 1969 - 1974. doi: 10.1109/ICPR.2014.344

Mahmoudi, N and Duman, E. 2015. Detecting credit card fraud by Modified Fisher Discriminant Analysis. *Expert System With Applications* 42(5): 2510 - 2516.

Masoumeh Z., Seeja K and M.Afshar A. 2012. Analysis of Credit Card Fraud Detection Techniques: based on Certain Design Criteria. *International Journal of Computer Applications* 52(3): 35 - 42.

Quah, J and Sriganesh, M. 2008. Real-time credit card fraud detection using computational intelligence. *Expert Systems With Applications* 35(4): 1721 - 1732.

R Core Team. (2015). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Rokach, L and Maimon, Z.O. 2007. Data mining with decision trees: Theory and Applications. 1^{st} ed. World Scientific Publishing Co. Pte. Ltd: Singapore.

Rokach, L. 2016. Decision forest: Twenty year of research. *Information Fusion, January 2016.* 27: 111-125.

Song, Y and Lu, Y. 2015. Decision tree methods: application for classification and prediction. *Shanghai archives of psychiatry* 27(2): 130 - 135.

Sventnik, V., Liaw, A., Tong, C., Culberson, J.C., Sheridan, R and Feuston, B. 2013. Random forest: a classification and regression tool for compound classification and QSAR modelling. *Journal of chemical information and computer schiences.* 43(6): 1947-1958.

University of Notre Dame. 2012. Data Science for the common good. *Hellinger Distance Decision Tree (HDDT).* http://www3.nd.edu/∼dial/hddt/. (Accessed 2017-01-23).