FYS-STK3155/4155 Project 3: Comparisons of machine learning techniques for credit card fraud detection

Michael Bitney and Magnus Grøndalen

December 18, 2019

Abstract

In this paper we have looked at Credit card transaction for 48 hours to see if we can train models using machine learning to detect fraudulent transactions. We saw that our data set was very unbalanced, where the frauds was 0.0172 % of all transactions. We decided to undersample the data set, where we removed random parts of legitimate transactions, so we can train and test the model with more balanced data. We decided to try different ratio between the fraudulent and legitimate transactions, which were 1:1 and 1:100. (100 is the legitimate ones) We also decided that Recall score function was the focus for our project, because we wanted to see how many frauds our methods could find.
We tested four methods in this project, which were Logistic Regression, Neural Network, Decision Tree and Random Forest. Here we saw that Neural Network performed the best with a Recall score of 0.9355 for 1:1, also having both a good k-fold score and test score in both ratios, but was time-consuming, Logistic Regression was in second, but took a lot less time and was unstable for ratio 1:100. Random Forest and Decision tree was ranked third, as they were prone to overfitting at oversampled values, although they performed well at the ratio 1:1.
Based on these result we recommend Neural Network as the best classifier in this case, but Logistic Regression is better if you want a less time-consuming method.

The material for this project can be found in our GitHub-repository at:
https://github.com/michaesb/machine_learning_3.git

1 Introduction

In this paper we look at fraud detection in credit card data. Today, credit cards are susceptible to fraudulent transactions and this can charge customers for items they did not purchase. Credit card companies need to recognize these frauds, but because of the sheer number of transaction that happen, they become needles in a haystack. This makes it a big task to investigate manually, but machine learning techniques can help us perform the data analysis. Machine Learning is today a growing field with numerous breakthroughs, which can offer high accuracy identifiers to find the fraudulent transactions. This allow banks to better handle frauds and consequently prevent the cost for the customers. We will here look at credit card transactions for a 48-hours period and create a model of the fraud and non-frauds using four machine learning methods and compare their performance.

We will first go through the theory of some of the methods and the score functions and present the data set and the results of the methods. Then, we discuss the results in and come with our conclusion.

2 Theory

Credit Card data

Here we look at data from the Kaggle site. They were taken over two days in September 2013 and it's specified that this is European cardholders. The data sets contains features which is the information around the transaction and whether the transaction is fraudulent or not.

Classification Methods

A regression method is a method using a statistical process to determine the output based on the relationship between one or multiple input parameters. We will in this paper utilize 4 methods, but we will explain the function only for decision tree and Random forest.

For more information on how Logistic Regression and Neural Network works, we invite the reader to check out a previous report on Regression Methods and Neural Network [5].

Logistic Regression

We will not go into the workings of Logistic Regression, but we will mention the different parameters it needs to function. Logistic regression is a simple, yet powerful method for creating classifiers with a good prediction. The parameters for Logistic Regression include a learning rate, which defines how much it varies the parameters to find the best minimum point. It also includes two different options for the penalty called L1 and L2 and we can also put a limit on the number of iterations the function performs. Note that it is the parameters for `Scikit-Learn` that is our focus in this assignment.

Neural Network

As with Logistic Regression, we will mainly focus on the parameters given to the Neural Network.

Neural Network has a great deal of parameters, from the number of layers in network, number of neurons in a layer, penalty choice, and limit to the iterations. Neural network has an almost infinite amount of permutations and is difficult to find the best solution for our data.

Decision Tree

Decision Tree is one of the easier methods to understand and is less of a black box compared to other machine learning algorithms. It bases on sorting information and by using a condition, for example $x_1 < 10$, it then decides if it moves to the "false" node or the "true" node. This process continues until it reaches the bottom node or also called leaf. Here it will be classified and can give an output based on which leaf the sample ended at. In order to find a good Decision Tree, the algorithm must decide if the tree should split into two branches and what feature should cause the split. To do this we use a Gini index, see formula 1. The p in the formula is the probability of it being classified to a particular class. The Gini index tells us how important each feature is. This let's us find the best divides for the branches. An example of an algorithm that calculates this, is the CART-algorithm[1], which can be used both for classification and regression.

$$G = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{1}$$

See graph 6 for an example of a Decision Tree.

Random Forest

Decision Trees have some drawbacks that sometimes make it difficult to obtain a stable predictive performance. This is especially the fact that Decision Trees are really prone to overfitting data. It depends a lot on many factors such as the size and balance of the data set, number of features and if features are continuous or categorical and so forth. This can be avoided by using other methods such as boosting, bagging and random forests. What these other methods have in common is that they create several trees instead of one, and improves the performance of the model.

Random forest is a form of bagging, and it creates a jungle or forest of trees, in order to avoid overfitting to the training set. The drawback of creating a multitude of trees with this algorithm, instead of using the Decision Tree, is that we again return to a black box. It is not as easy to interpret the model as for the diagram created from the decision tree.

The random forest method creates several bootstrapped training samples and then builds trees from those training samples. A difference from the decision tree to random forest is when creating the splits in the trees. When creating a split, we randomly choose m of the total p features, and then choose one of these features to create the new branches. Normally we choose the sub-sample m to be

$$m = \sqrt{p}.$$

The reason for choosing only between m of the p features is that there might be some predictors that are more important than the others, causing the splits to often use the same predictors, thus making many of the generated trees similar. Limiting the choice of feature for splitting then generates a forest of trees that are different, increasing the search in the fitness space of the data set. data set

Methods in Scikit-Learn

For this whole project, we decided on using `Scikit-Learn`'s machine learning methods. The methods in `Scikit-Learn` are tested and proven, so that the the work in this project goes to finding good parameters and comparing the results of the different methods.

We used `Scikit-Learn`'s `LogisticRegression`, `MLPClassifier`[2], `DecisionTreeClassifier` and `RandomForestClassifier`. Each of these methods have a multitude of different parameters that we can adjust to create a good classifier, so we need a method for choosing the values for the parameters.

Grid search is one method for finding parameter-values that optimize the score of our choosing. Grid search is a very simple algorithm that goes structurally through every parameter chosen by the user, creates a model and predicts the output. It then lets us see what parameters caused the best score. The `GridSearchCV` method we chose in `Scikit-Learn` uses cross-validation to improve the scores. This method also allows us to choose the score function we'd like to optimize.

[1] Classification and Regression Trees

[2] MLP: Multi Layered Perceptron. A version of neural networks.

Testing the models

In order to test our models, we use multiple score functions to determine different properties of each of the models.

	Actual 1	Actual 0
Predict 1	TP(True Positive)	FN(False Negative)
Predict 0	FN(False Positive)	TN(True Negative)

Table 1: Confusion Matrix to display the different outcomes possible when comparing the model to the actual result.

We will look into an accuracy function as a way to determine how accurate it is to predict both legitimate transactions and frauds. See formula 2. This will measure the general performance of the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

We can also look at Precision which is defined as how well it is to find the legitimate transaction. See formula 3. This will ignore it's ability to find frauds, but focus entirely on the legitimate transactions.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

We also have a measure of it's ability to detect frauds, called Recall. See formula 4. Much like Precision, it focus only on type of prediction and ignore the other.

$$\text{Recall} = \frac{TN}{TN + FN} \tag{4}$$

3 Results

Data processing

For the data set we chose, most of the features are made up from anonymous data, due to privacy of the users, so we don't get any information on what those features mean in a practical sense. There are 30 features in total, and the only two features that aren't anonymous data is the "Time" feature and the "Amount" feature. The time feature just states when, in time,the transactions took place after the first transaction. The amount feature says how big the transactions were.

Although 28 of the features are unknown, the description specified that the unknown parameters are the result of a PCA (Principal Component Analysis) dimensionality reduction. This means that the only features that hasn't been transformed by PCA are the "Time" and "Amount" features. Since 28 of the 30 features are transformed, we decided not to perform any more PCA reduction on the features.

Because of the unknowns in the features, we decide to not focus on which parameter is most important, but on the performance of the methods.

The data set has a total of 284 807 transaction, with only 492 of them being fraudulent, as you can see in figure 1. This fact makes this data set highly imbalanced, where 99.827 % of the data set are zeros, e.g. regular non-fraudulent transactions.

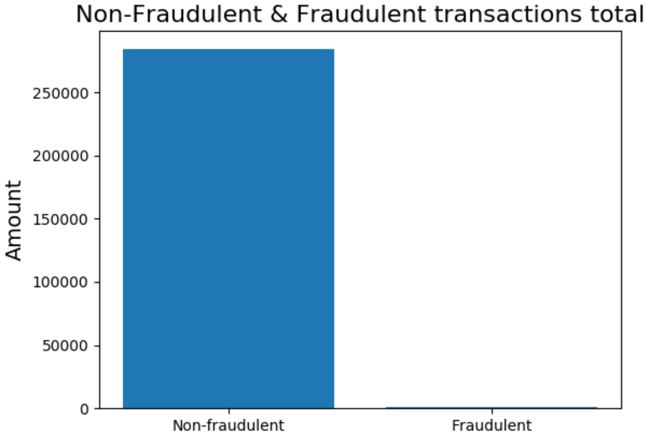

Figure 1: Here you can see the distribution of non-frauds and frauds in the data set. As you can see the data set is imbalanced, which can make it difficult for the methods to process.

To deal with the highly imbalanced classes, we make use of undersampling techniques in order to even out the two classes. We used a simple randomized undersample where we just randomly picks out the same amount of non-fraudulent samples as there are fraudulent samples. With this we end up with a much smaller, but more balanced data set, as you can see in figure 2. This data set now only has 984 samples, and a possible solution to the now small data set, would be to not undersample with a 1:1 ratio, but rather 1:2, 1:4, or 1:10. This would give us more data to work with, while we still have enough fraudulent transactions in the data set.

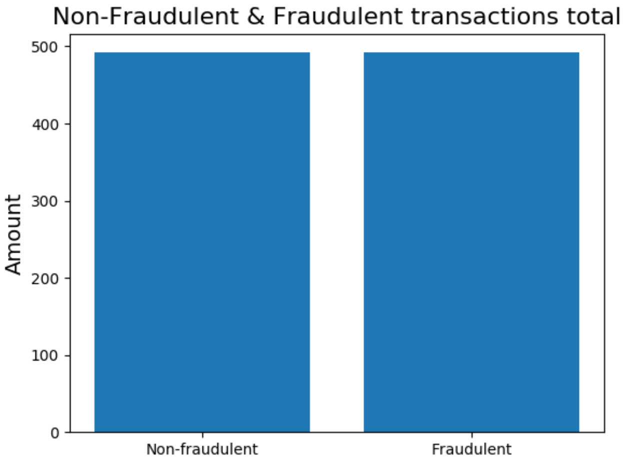

Figure 2: Here we have performed undersampling on our data with a ratio 1:1 and display that amount is equalized

We continued to look deeper at the data set, and made two distribution plots for the "Time" and "Amount" features, as you can see in figures 3 and 4. This plot creates a histogram to give us more info on how the inputs are distributed. In the time-distribution plot, we can tell now it took place in a 48-hour window and we can see there are two big "centers" in time where most of the transactions took place during the day.

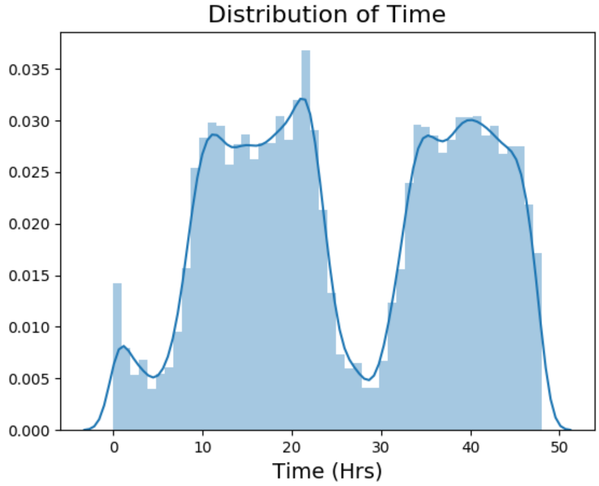

Figure 3: Here we see the distributions of transactions as functions of time. We see the two peaks in the middle of the day, which showed that most the transaction during the day.

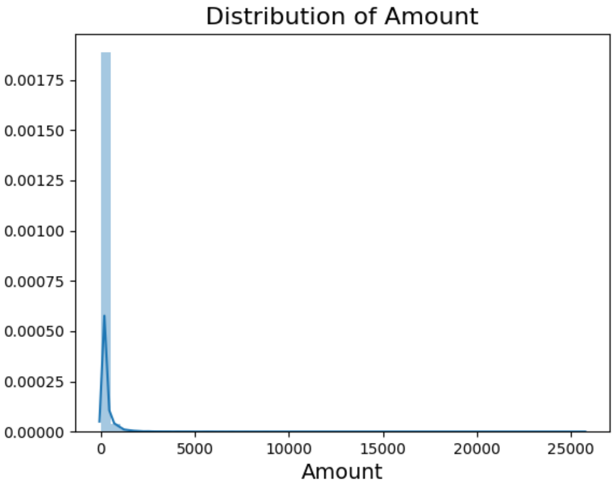

Figure 4: Distribution of the amount. We have most the amounts below 250, but have few higher amounts, the highest being 25691. The data set doesn't specify which currency is used, but considering this being in Europa, we can guess this is in Euros.

Finally, we used Scikit-Learn's StandardScaler to scale the data. Time and amount were the only features that needed scaling.

We plotted the performance on decision tree as to see how the scores behaved at different ratios. This can be seen in graph 5. We choose to do this with decision tree, because it's a lighter method compared to the others, and we wanted to get a grasp of different ratios affected the score functions.

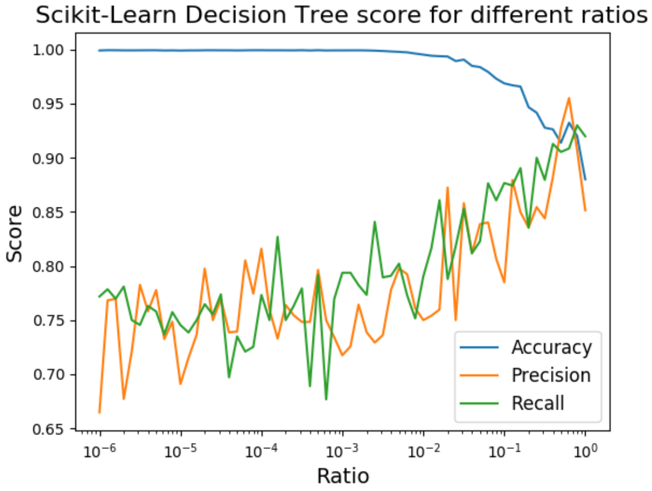

Figure 5: plotted the accuracy, precision and recall score as function of the ratio between the number of frauds and non-fraudulent transactions. The smallest data set with a 1:1 ratio is at the far right, and the data set is increasing as the graph moves to the right.

We see two interesting areas in graph 5 at ratio

1:100 and higher, where it seems to be quite constant, and ratio 1:1, where the accuracy and precision + recall meet. From ratio 1:100 and higher, it seems saturated, we decide to look at ratio 1:100, as it's the smallest ratio that still seem saturated. We are also interested in the ratio 1:1, where the recall and precision is at it's best. These are the ratios we will look into with our methods in our reports, and we present the scores for these ratios in table 2 and 3.

In this project, we have decided to focus on the Recall score. The reason for this is that we think that it's more important to reduce the number of frauds being classified as legitimate (False Positives), than classifying a legitimate transactions as fraud (False Negatives). We think that our algorithm is best used as a way to identify frauds, but should be checked by someone or something else. Because of this, our model should be used to bring fourth candidates for the investigators, where it will tag too many transactions, but few frauds will be missed by the model. Therefore we will focus on creating a model where the recall is at it's best, but not ignoring accuracy and precision. We have adjusted the grid search to find the best value of recall when we have looked for the best model.

Logistic Regression

The first machine learning method we tried for this data set is logistic regression.

To start, the logistic regression in `Scikit-Learn` takes a lot of parameters, but we choose to optimize the parameters "penalty" and "C", while using the solver "liblinear". The penalty can be chosen to be "l1", "l2", "elasticnet" or "none", and it specifies which norm to use in the penalization. By using the "liblinear" solver, we can use "l1" or "l2" penalty. The parameter "C" specifies the strength of the penalization, or regularization. More accurately, it is the inverse of the regularization strength.

With our grid search method, we found that the best parameters for the logistic regression are `C = 0.01` and `penalty = "l1"`. This gave a score of

CV Recall Score: 0.9496.

Neural Network

The second method we used is the neural network in `Scikit-Learn` called `MLPClassifier`.

The parameters of the `MLPClassifier` are many and sometimes difficult to understand. For our neural network we chose to use an adaptive learning rate, with an initial rate of 0.001, and a tolerance of 10^{-4}. The parameters we optimize with our grid search is the activation function, the regularization parameter "alpha", the number of neurons and layers, the maximum iterations in the algorithm and finally which solver to use. Most of the parameters we used are self-explanatory, but some are not. The solver, for example, is the solver which decides the weight optimization, using different versions of gradient descent and so forth.

The grid search found the best parameters to be the sigmoid/logistic activation function, `alpha = 0.1`, 4 hidden layers with 30 neurons in each, a maximum iteration of 500 and the solver "lbfgs". With these parameters we got a score of

CV Recall Score: 0.9555,

Decision tree

We also tried using a decision tree for classification because of its simplicity and interpretability. The exact method we use is `Scikit-Learn`'s `DecisionTreeClassifier`. It also has a bunch of different parameters to choose from, and we ended up optimizing the criterion of a split, the maximum depth of the tree, the leaf node minimum samples, the minimum number of samples to create a split and the number of features for creating a split. When using the grid search we found the best parameters to be the splitting criterion Gini, maximum depth of 20, maximum features of 30, minimum sample leaf of 1 and a minimum samples for split of 2. This gave us the score of

CV Recall Score: 0.9367.

The decision tree we ended up with for the parameters above, can bee seen in figure 6

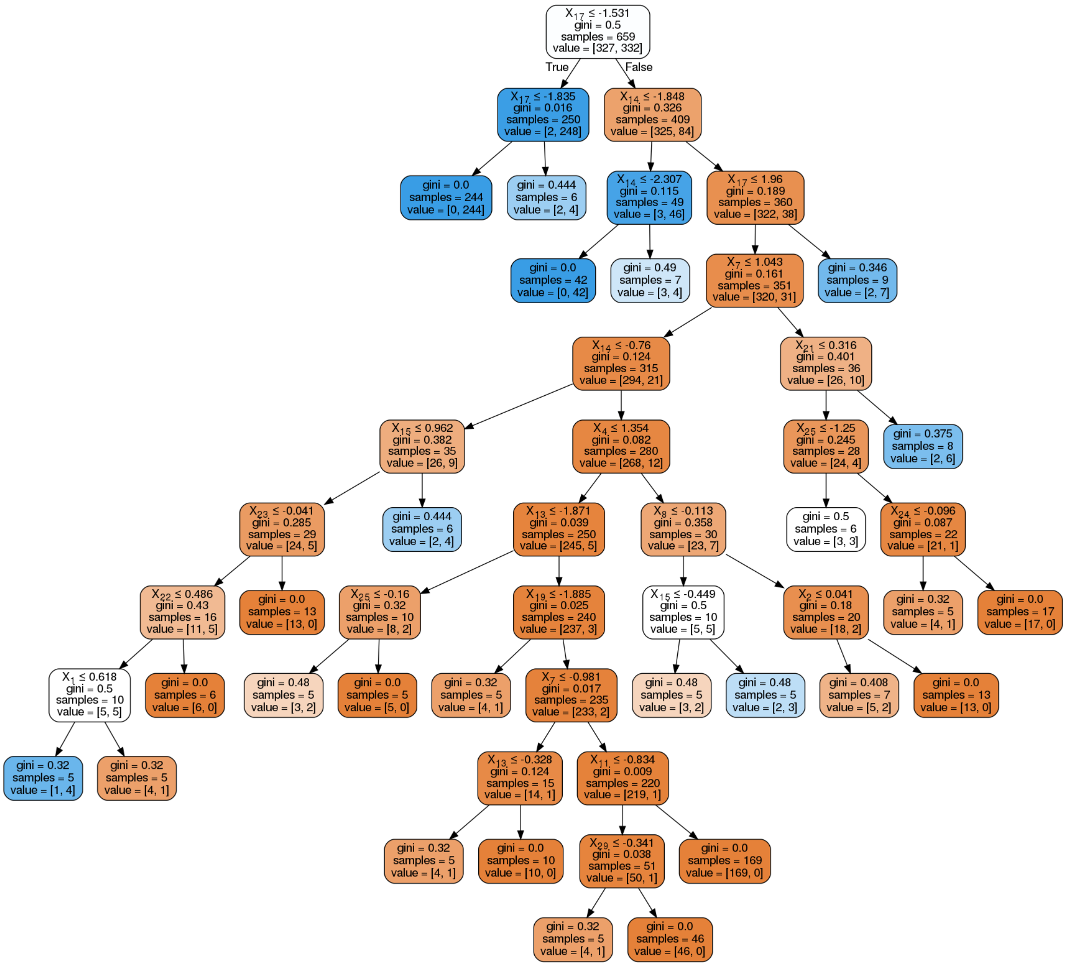
$X_{17} \leq -1.531$ gini = 0.5 samples = 659 value = [327, 332]
True
False
$X_{17} \leq -1.835$ gini = 0.016 samples = 250 value = [2, 248]
$X_{14} \leq -1.848$ gini = 0.326 samples = 409 value = [325, 84]
gini = 0.0 samples = 244 value = [0, 244]
gini = 0.444 samples = 6 value = [2, 4]
$X_{14} \leq -2.307$ gini = 0.115 samples = 49 value = [3, 46]
$X_{17} \leq 1.96$ gini = 0.189 samples = 360 value = [322, 38]
gini = 0.0 samples = 42 value = [0, 42]
gini = 0.49 samples = 7 value = [3, 4]
$X_{7} \leq 1.043$ gini = 0.161 samples = 351 value = [320, 31]
gini = 0.346 samples = 9 value = [2, 7]
$X_{14} \leq -0.76$ gini = 0.124 samples = 315 value = [294, 21]
$X_{21} \leq 0.316$ gini = 0.401 samples = 36 value = [26, 10]
$X_{15} \leq 0.962$ gini = 0.382 samples = 35 value = [26, 9]
$X_{4} \leq 1.354$ gini = 0.082 samples = 280 value = [268, 12]
$X_{25} \leq -1.25$ gini = 0.245 samples = 28 value = [24, 4]
gini = 0.375 samples = 8 value = [2, 6]
$X_{23} \leq -0.041$ gini = 0.285 samples = 29 value = [24, 5]
gini = 0.444 samples = 6 value = [2, 4]
$X_{13} \leq -1.871$ gini = 0.039 samples = 250 value = [245, 5]
$X_{8} \leq -0.113$ gini = 0.358 samples = 30 value = [23, 7]
gini = 0.5 samples = 6 value = [3, 3]
$X_{24} \leq -0.096$ gini = 0.087 samples = 22 value = [21, 1]
$X_{22} \leq 0.486$ gini = 0.43 samples = 16 value = [11, 5]
gini = 0.0 samples = 13 value = [13, 0]
$X_{25} \leq -0.16$ gini = 0.32 samples = 10 value = [8, 2]
$X_{19} \leq -1.885$ gini = 0.025 samples = 240 value = [237, 3]
$X_{15} \leq -0.449$ gini = 0.5 samples = 10 value = [5, 5]
$X_{2} \leq 0.041$ gini = 0.18 samples = 20 value = [18, 2]
gini = 0.32 samples = 5 value = [4, 1]
gini = 0.0 samples = 17 value = [17, 0]
$X_{1} \leq 0.618$ gini = 0.5 samples = 10 value = [5, 5]
gini = 0.0 samples = 6 value = [6, 0]
gini = 0.48 samples = 5 value = [3, 2]
gini = 0.0 samples = 5 value = [5, 0]
gini = 0.32 samples = 5 value = [4, 1]
$X_{7} \leq -0.981$ gini = 0.017 samples = 235 value = [233, 2]
gini = 0.48 samples = 5 value = [3, 2]
gini = 0.48 samples = 5 value = [2, 3]
gini = 0.408 samples = 7 value = [5, 2]
gini = 0.0 samples = 13 value = [13, 0]
gini = 0.32 samples = 5 value = [1, 4]
gini = 0.32 samples = 5 value = [4, 1]
$X_{13} \leq -0.328$ gini = 0.124 samples = 15 value = [14, 1]
$X_{11} \leq -0.834$ gini = 0.009 samples = 220 value = [219, 1]
gini = 0.32 samples = 5 value = [4, 1]
gini = 0.0 samples = 10 value = [10, 0]
$X_{29} \leq -0.341$ gini = 0.038 samples = 51 value = [50, 1]
gini = 0.0 samples = 169 value = [169, 0]
gini = 0.32 samples = 5 value = [4, 1]
gini = 0.0 samples = 46 value = [46, 0]

Figure 6: Here you can see how a decision tree check data at each node and redirects the data to one of the two branches all the way down to the final node called the leaf. This decision tree that was used in our project.

Random Forest

Lastly, we made a model using a random forest method. This method is part of `Scikit-Learns` ensemble methods, and is called `RandomForestClassifier`. As all the other methods, it takes a lot of parameters and can be very much tuned. For optimization with grid search, we chose the parameters for maximum tree depth, maximum features considered for a split, minimum samples required for split, minimum samples at a leaf node, splitting criterion, and the number of estimators, i.e. the number of trees in the forest. When using the grid search we found the parameters that optimized the recall score are the splitting criterion Gini, a maximum depth of 15 nodes, maximum features of 30, minimum samples for a leaf of 1, a minimum samples for split of 10 and using 10 decision trees. With these parameters we got a score of

CV Recall Score: 0.9371.

Comparison

Method	CV Recall Score	Test Recall Score	Test Precision Score	Test Accuracy
Logistic Regression	0.9496	0.9226	0.8994	0.9138
Neural Network	0.9555	0.9355	0.9062	0.9231
Decision Tree	0.9367	0.9250	0.8970	0.9108
Random Forest	0.9371	0.9138	0.9464	0.9262

Table 2: Here we have presented the scores for the different methods with a ratio of 1:1. When running these methods Neural Network and Random Forest took the longest running grid search on, and decision tree and logistic regression needed less time.

Method	CV Recall Score	Test Recall Score	Test Precision Score	Test Accuracy
Logistic Regression	0.7882	0.8373	0.9329	0.9977
Neural Network	0.8342	0.8434	0.8434	0.9968
Decision Tree	0.8416	0.7927	0.9848	0.9978
Random Forest	0.8537	0.7953	0.9577	0.9975

Table 3: Here we have presented the scores for the different methods with a ratio of 1:100. When running these methods Neural Network and Random Forest took the longest running grid search on, and decision tree and logistic regression needed less time. Since this was a bigger data set than the other ratio in table 2, this took also longer to run.

4 Discussion

We decided that the main focus of this projects was to test different machine learning methods for a specific dataset and try to find which parameters optimized the score for each of the methods. As stated earlier in the report, we decided to optimize the method parameters to maximize the recall score in (4). The recall score basically tells us how many of the total fraudulent transactions our prediction models manages to classify correctly. We decided its more important to classify all of the fraudulent transactions and maybe then some, rather than classifying all the non-fraudulent transactions correctly.

The method we used to optimize this recall score was the grid search method, which uses brute force, to test all the different permutations for the parameters. This grid search can be very time-consuming for a large number of parameters, and there exists another method called randomized search, that lets you go through a bigger variety of parameters to obtain a good score without being so time-consuming.

Logistic Regression

Neural Network

Neural networks are good at finding a good model for complex data, but not always as good, when using simple data, compared to the other simpler methods. We

This is a lot of parameters, and our grid search used a lot of time to find the best parameters. We really suggest using the randomized search for the optimization of the neural network.

One downside of the `MLPClassifier` from `Scikit-Learn` is that you can not choose different activation functions for different layers, and it would be interesting to test with different activation functions using for example `TensorFlow`.

Decision Tree

6 5

Random Forest

Comparison

We see in table 2 and 3 a comparison of methods. As stated before we are here interested in a good Recall value, which will be our main focus. We have Logistic Regression

5 Conclusion

We have compared four methods in this paper with focus on their ability to detect fraudulent transactions. To find the best model we used a score function called Recall. We tested these methods for two data set, one where the was the same amount of fraudulent and legitimate transactions and one where the legitimate was 100 times bigger than the frauds.

For both data sets, we have ranked method as such:

1. Neural Network
2. Logistic Regression
3. Random Forest + Decision Tree

Neural Network was however quite a time-consuming model to train although it was stable, so Logistic Regression could be preferable in some cases.

We saw that although Random Forest is known to be better at overfitting than Decision Tree, we see that in ratio 1:100, it did not perform better than Decision Tree in this case here, when the data set is quite unbalanced. We have put them in equal third place, because although Random Forest performed better than Decision tree, it took a lot longer, so we ranked them equally in this project.

There are many machine learning methods that we haven't tested for this e.g. Boosting methods and Bayesian algorithms and we suggest trying them in order to see if they perform better than the method tested here.

We also recommend trying the same methods with more data, so we can train and test our models better.

We also suggest testing Randomized search CV from `scikitlearn`, in order to find a better model and do a more thorough search of the fitness space.

We used `Scikitlearn` in this project, but `TensorFlow` is worth looking into, where one can adjust the activation function for the different methods.

References

[1] Lecture Notes made by Morten Hjorth-Jensen in class FYS-STK3155/4155 at UiO https://compphysics.github.io/MachineLearning/doc/web/course.html

[2] Hastie Tibshiran Friedman; The Elements of Statistical Learning; Second edition; Springer 2009

[3] Géron, Aurélien; Hands-On Machine-Learning with Scikit-Learn & TensorFlow; First edition; O'Reilly 2017

[4] Yeh, Lien; The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients;

[5] Grøndalen Bitney; Project Classification for the subject FYS-STK3155/4155 https://github.com/michaesb/machine_learn_2/blob/master/report2_fysstk_magnubgr_michaesb.pdf

Due to request from data holder Machine Learning Group - ULP, we have cited the works below.

[6] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

[7] Dal Pozzolo, Andrea; Caelen, Olivier; Le Borgne, Yann-Ael; Waterschoot, Serge; Bontempi, Gianluca. Learned lessons in credit card fraud detection from a practitioner perspective , Expert systems with applications,41,10,4915-4928,2014, Pergamon

[8] Dal Pozzolo, Andrea; Boracchi, Giacomo; Caelen, Olivier; Alippi, Cesare; Bontempi, Gianluca. Credit card fraud detection: a realistic modeling and a novel learning strategy , IEEE transactions on neural networks and learning systems,29,8,3784-3797,2018,IEEE

[9] Dal Pozzolo, Andrea Adaptive Machine learning for credit card fraud detection ULB MLG PhD thesis (supervised by G. Bontempi)

[10] Carcillo, Fabrizio; Dal Pozzolo, Andrea; Le Borgne, Yann-Aël; Caelen, Olivier; Mazzer, Yannis; Bontempi, Gianluca. Scarff: a scalable framework for streaming credit card fraud detection with Spark , Information fusion,41, 182-194,2018,Elsevier

[11] Carcillo, Fabrizio; Le Borgne, Yann-Aël; Caelen, Olivier; Bontempi, Gianluca. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization , International Journal of Data Science and Analytics, 5,4,285-300,2018,Springer International Publishing

[12] Bertrand Lebichot, Yann-Aël Le Borgne, Liyun He, Frederic Oblé, Gianluca Bontempi Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection , INNSBDDL 2019: Recent Advances in Big Data and Deep Learning, pp 78-88, 2019

[13] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Frederic Oblé, Gianluca Bontempi Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection Information Sciences, 2019

Acknowledgements

Thanks to Morten Hjorth-Jensen and student teachers in FYS-STK3155/4155 for help during this project. We also thank the Machine Learning Group at the University of Luxemburg, Brussel for providing the data.