Introduction to Machine Learning (SS 2023)
Programming Project

Author 1
Last name: Achatz
First name: Finn
Matrikel Nr.: 12120212

Author 2
Last name: Kirchmair
First name: Lukas
Matrikel Nr.: 11946551

Author 3
Last name: Kurz
First name: Andreas
Matrikel Nr.: 12116476

I. Introduction

Since the begin of transactions there are people that try to take advantage Fraud detection is a critical task in various industries, including finance and banking. With the advancement of machine learning techniques, it has become possible to automate the process of identifying fraudulent transactions. In this report, we explore the application of machine learning, specifically neural networks, also with data augmentation, and logistic regression, to classify fraudulent transactions. As a transaction dataset for fraud detection we got a dataset consisting of 227'845 samples, with 394 instances of fraudulent transactions. The datasat has 30 features which include one for the time of the transaction and also the amount of the transaction. The rest of the features are anonymized. The dataset is highly imbalanced, with only 0.17% of the transactions being fraudulent.

II. Implementation / ML Process - Neural Network

For the neural network we removed the time column, because fraud can happen anytime and is not dependent on the time. So we had 29 features and the output class.

We used a CNN with the torch library and as activation function we used the ReLU function. We decided on a neural network, with 1 hidden layer with 50 nodes. The output is then determent with a sigmoid function on the output, that has to be bigger then 0.5.

The neural network is very suitable for the classification, because you can use a neural network with one output neuron. Which is perfect for binary classification.

Out hyper parameters for the convolution neural network where: learning rate, momentum, iterations, number of hidden layers, hidden layer sizes and an output threshold.

The learning rate is how fast the neural network should converge, the higher the faster is the training, but if it is too high the model will not be accurate anymore.

Momentum is used by the optimizer of the torch library and if we lowered the value it seemed that the network got a little bit more random on how the results where, if we lowered it it once got a 50% accuracy and then it got a 99% accuracy without changing anything else, so we decided to leave it at 0.9.

The iterations are trivial, those are the number of times we update the weights and biases in training.

The number of hidden layers is a trivial attribute, but not so trivial to select, so we just did some trial and error and came to the conclusion to use 3 hidden layers because we still have 29 features and that is not little.

The hidden layer sizes, we decided to have them go down as we go to the output layer.

The output threshold is used to determine on what value of the sigmoid function it should say that it is fraud.

We played around with all hyper parameters and the best result we got with those:

1) Learning rate: 0.01
2) Momentum: 0.9
3) Iterations: 250
4) Hidden Layers: 3
5) Hidden Layer sizes: 50, 25, 12
6) Output threshold: 0.5

III. Result - Neural Network

Our neural network performance is very good on paper, if we only see the accuracy of the fraud detection, because it it 99.8%. But the problem is that it does not detect any frauds, it just says that the input is always not fraud and therefor it is 99.8% of the time correct, because the data is so biased on the non frauds.

That is why we decided also use some data manipulation that we talk about in point IV and V. There we use over sampling and under sampling.

IV. Implementation / ML Process - Neural Network with Over- and Under-sampling

In this section we used once again the same classification method, since we have a binary classification problem, but tried to improve the performance of our neural network by using oversampling and under sampling. Oversampling is done by increasing the representation of the minority class (in this case fraudulent transactions) by duplicating existing

samples. under-sampling, on the other hand, involves reducing the representation of the majority class (non-fraudulent transactions) by removing samples. In the following section we will analyze the performance of our neural network with said data transformations. As base we started with the same neural network as in the previous section. The major difference is that we applied oversampling and under sampling to our data set before training the neural network. This gave us an additional hyper parameter called fraud ratio, this is the ratio of frauds we want to have in the data set after applying over- or under sampling to the training set, the other hyper parameters are still the same.

We set the fraud ratio to 0.2 so we have 20% frauds to test, this is around 100x more frauds then it would have without the data sampling. On oversampling this causes the test data to grow from 170,000 to 205,000 and on under-sampling to shrink from 170,000 to 137,000.

V. Result - Neural Network with Over- and Under-sampling

The results on the test set that was never seen of the neural network before is insanely good for both under and over sampling, with a 99.94% accuracy and a 80% fraud detection.

But what was kind of weird is, that if we sample new test data from our data set and test on that we detect way more or no frauds and then the accuracy drops significantly. The model is saved properly and also loaded properly, but for some reason that we cannot understand this problem occurs. It is more drastic the higher we pump our fraud ratio, so it has to do something with the sampling. This is a weird problem, because this means that the model predicts on the training data more frauds then on the actual test data that it has never seen before, because if we sample again we get also training data in our test samples.

Also one can see in the Fig.1, the accuracy ranges from 96.5% to 99.9% if we vary the fraud ratio. The best performance was observed with a fraud ratio between 0.002 and 0.010 with over and undersampling alike.

VI. Implementation / ML Process - Logistic Regression

To train our logistic regression model we used the sklearn library. We first had to split our data in a trining and test set, which consistet of 77% and 33% of the data respectively. For the random state we already know that there is only one value that provides the best results, which is 42. (Common knowledge) We then trained our model with the training data and tested it with the test data. We also used the sklearn library to calculate the accuracy of our model. Logistic regression is a classification algorithm that is used to predict the probability of a categorical dependent variable. In our case the dependent variable is the class of the transaction, which is either fraudulent or not fraudulent. Since logistic regression is a commonly used algorithm for binary classification, we can already assume that it is suitable for our task. Logistic regression provides coefficients for each feature,

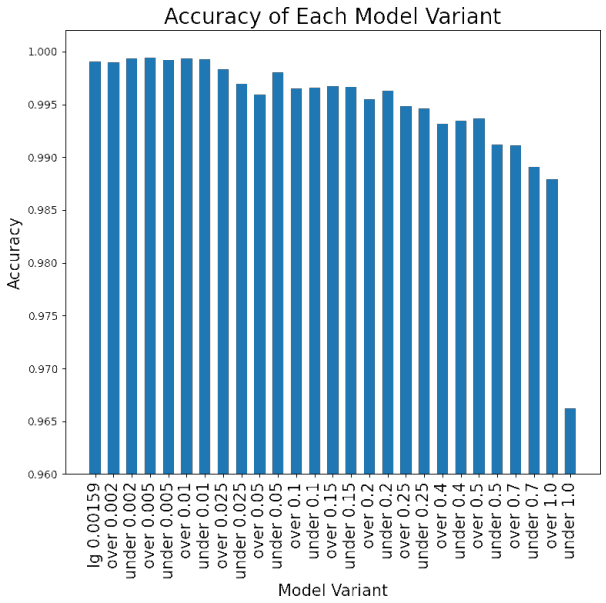

Fig. 1. Results of a neural network with data augmentation.

which can be used to determine the importance of each feature. This can be useful for fraud detection, since it can help to identify the most important features for fraudulent transactions.

The only hyperparameters we changed were the random state, which we set to 0 to get a reproduceable result, and the max_iter which we set to 1000. For all the other used the default hyperparameters for our logistic regression model. The default values for the hyperparameters are the following.:
dual = False, tol = 1e-4, C = 1.0, fit_intercept = True, intercept_scaling = 1, class_weight = None, random_state = 0, solver = 'lbfgs', max_iter = 1000, multi_class = 'auto', verbose = 0, warm_start = False, n_jobs = None, l1_ratio = None.

VII. Result - Logistic Regression

After training our model with the set parameters we tested it on a test set that consisted of 75'189 total transactions, with 75'070 non-fraudulent and 119 fraudulent transactions. From those 75'040 were predicted correctly as non fraudulant and 78 were predicted as fraudulent. This results in an accuracy of 99.906%. See Fig.2.

Column1	Total	Predicted	Correct	Incorrect	Accuracy
Num Non Frauds	75070	75081	75040	41	99,9600%
Num Frauds	119	108	78	30	65,5462%
Num Data	75189	75189	75118	71	99,9056%

Fig. 2. Results of the logistics regression.

VIII. Implementation / ML Process - Random Forest

The data was again split into train and test data sets, using the according sklearn function. The test size was chosen to be 33% of the given dataset.

Random Forests combine multiple decision trees, aggregating their results. Thus, they are a non-parametric classification algorithm. As impurity measure, we used the Gini index.

Aggregating the results of multiple decision trees, results in a lower risk of overfitting. This should be beneficial for this task, as the number of frauds in the dataset is quite small and there is the risk of the model always predicting non-fraud.

Here, we used a Random Forest Classifier from the sklearn ensemble library. The default parameters resulted in quite good predictions, so we ended up using them. By default, it uses the Gini impurity measure as split criteria. The number of trees is set to 100. Not defining a maximal depth results in quite accurate predictions, but takes some time to train. Using this configuration lead to an accuracy 99.92%. To further reduce the number of miss-classified frauds, different weights were assigned to the 2 classes. Assigning a higher weight to the frauds (here, $w(\text{fraud}) = 10, w(\text{non-fraud}) = 1$), ended up being the best. This slightly increases the number of miss-classified non-frauds, but also decreases the number of missed frauds by about 5%, Then, the oversampling already implemented was used. Increasing the ratio of frauds in the dataset again decreases the number of frauds miss-classified, while increasing the miss-classified non-frauds. In a real environment, this compromise should be taken, as it is much more important to classify frauds correctly, than mistakenly classifying a valid transaction as fraud.

IX. Result - Random Forest

The best fraud detection ratio (number of frauds in test set / number of frauds detected $= 84.6\%$) was achieved by setting no maximal depth and 100 decision trees, adding oversampling (fraud ratio $= 10\%$) and giving the frauds a weight of 10 in the training process. The result is displayed in Figure 3, the over all accuracy is 0.9995%.

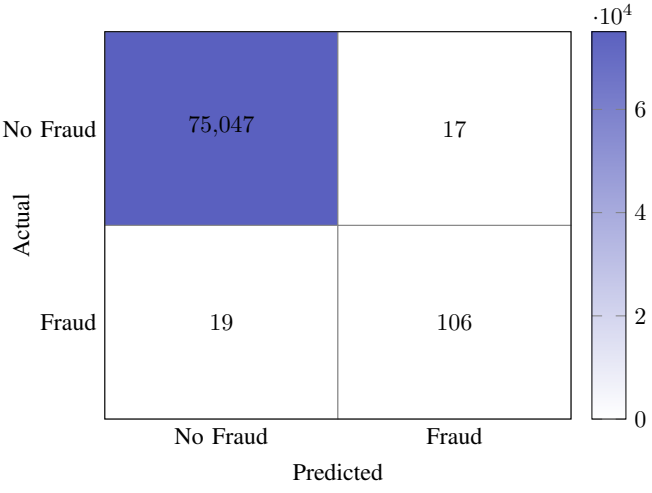

Fig. 3. Confusion matrix of the best random forest implementation

X. Results - Summarized

From the tree models we used (neural networks, logistic regression, random forests), random forests performed the best in terms of over all accuracy (99.96%) and fraud detection ratio (84.6%).

XI. Discussion

In analyzing the results of our implemented models, several observations can be made. The neural network initially showed a high accuracy of 99.8%, but it failed to detect any fraud cases due to the heavy class imbalance in the dataset. Although oversampling and undersampling techniques were applied to address this issue, the model still exhibited inconsistencies in fraud detection when tested on new data. Further investigation is required to understand and rectify this discrepancy in order to ensure the reliability of the model.

During the implementation process, we also experimented with logistic regression and random forests. Their performance outclassed that of the neural network, achieving accuracies of 99.906% and 99.92% respectively. Within those two models, random forests performed the best in terms fraud detection, with a fraud detection ratio of 84.6%. This is likely due to the fact that random forests are less prone to overfitting than neural networks, and they are able to handle imbalanced datasets better than logistic regression.

To further improve the performance of our models, several strategies can be considered. First, obtaining more data, especially fraudulent cases, would help improve the models' ability to detect fraud. Additionally, exploring advanced techniques such as anomaly detection algorithms or ensemble methods could potentially enhance the accuracy and robustness of the fraud detection models.

XII. Conclusion

In our evaluation of fraud detection models, we implemented neural networks, logistic regression, and random forests. Despite the imbalanced nature of the dataset, the models showed promising results.

The neural network achieved an accuracy of 99.8% but struggled to detect fraud due to the heavy class imbalance. By applying oversampling and undersampling techniques, the fraud detection rate improved to 80%. However, there were inconsistencies when testing the model on new data, requiring further investigation.

Logistic regression achieved an accuracy of 99.906% on the test set, while random forests outperformed the other models with an accuracy of 99.92% and a fraud detection ratio of 84.6%. These results demonstrate the effectiveness of machine learning techniques in fraud detection tasks.

Throughout this project, we learned the importance of handling imbalanced data and the challenges it poses for accurate fraud detection. We also gained insights into various machine learning algorithms and their strengths and limitations in addressing fraud detection problems. Future work should focus on refining the models performance and addressing the issues encountered in order to develop more robust and reliable fraud detection systems.