Introduction to Machine Learning (SS 2021)
Programming Project

Author 1
Last name: Kruselburger
First name: Patrick
Matrikel Nr.: 11910362

Author 2
Last name: Hauser
First name: Michael
Matrikel Nr.: 01267565

I. Introduction

This report is concerned about credit card fraud detection. Several prediction models from machine learning where used to detect if a transaction is a fraud or not. To get an idea which model is fitting best for this problem, pre-implemented python classifiers were used to see how accurate they are in detecting whether a transaction is a normal payment or a fraud.
Our dataset includes 227845 entries each contains 30 features and one class label, where 1 indicates it is a fraud. The distribution between the 2 classes, fraud and non-fraud, is 0.17% to 99.83%, which shows us that the data is totally unbalanced. Except for the amount and time, we don´t know what the columns are, due privacy reasons. The transaction amount is relatively small. The mean of all the amounts made is approximately 88 USD. For the final submission we have choosen the LR model instead of NN.

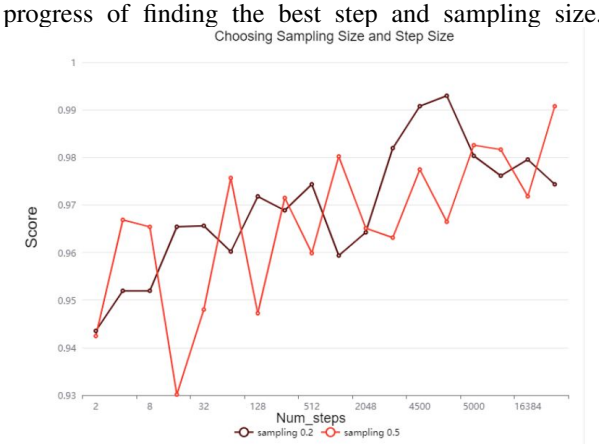
II. Implementation / ML Process

Due the fact that our dataset is imbalanced, only 394 out of 227845 entries are a fraud, we had to think about a way, how we can match up our minority class to the number of samples in the majority class, else it can happen that the machine learning algorithms lead to wrong solution. This is a problem as it is typically the minority class on which predictions are most important. Therefore we tried random undersampling, random oversampling and had a look on SMOTE. With these strategies overfitting and making wrong correlations get minimized. We also used on our dataset the standard scaler, to reshape the values of the features to a common range. This improves the prediction accuracy and speeds up the training. SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b. Additionally, we printed a correlation matrix and found that no feature has a high correlation to the resulting class label, hence we did not select specific features from the dataset.
Logistic regression for classification is commonly used to estimate the probabilities of whether an instance belongs to a particular class, in our case fraud or not. It is basically a binary classifier and so for this problem it fits pretty good. To calculate the probabilities, the logistic regression model computes a weighted sum of input features and bias and passes this through the sigmoid function. By using random undersampling we were deleting examples from the majority class to get a better distribution. The thing what can happen is, you can lose information with this technique if you use it to hard. To make sure that our model is performing good over the entire data set, we used batch gradient descent with 5000 steps, hereby all the training data is taken into consideration to take a single step. Batch gradient descent is great for convex or relatively smooth error manifolds. In this case, we move somewhat directly towards an optimum solution. The cost keeps on decreasing over the epochs. Choosing the parameters was a trial and error approach, with modifying the sample strategy and increase/reducing the ratio of it we gained after several attempts about 1 percent more and submitted it with a value of 0.2. Also interesting was finding the optimal iteration range for the batch gradient descent, therefore we tried also many different values and in the end we have chosen 5000.
Our activation function in the NN is a ReLU (rectified linear activation function), which is used in many types of neural networks, it´s a piecewise linear function that will output the input directly if it´s positive, otherwise, it will output zero. It acts like a linear function, but it´s a nonlinear function. This activation function allows our model to learn faster and perform better than a linear one. For setting good hyper parameters we read a few articles and tried many different combinations. We tried one, two and three hidden layers each tested with different number of neurons. For setting the perfect fitting learning rate we started to test different ones. As we can say now, there are 3 different cases what can

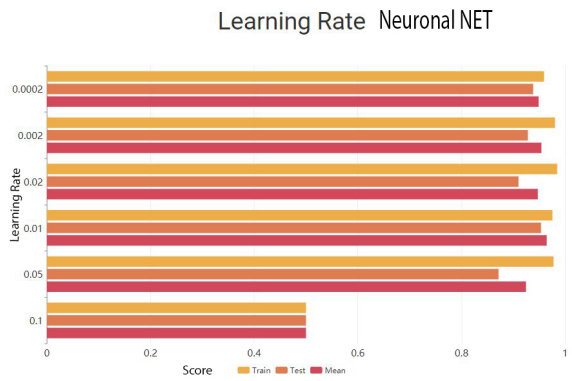
happen - starting from small to high value. With a small learning rate it requires many updates before reaching the minimum point, using a optimal one can reduce the steps to a minima dramatically and picking a too high learning rate can cause drastic updates which lead to divergent behaviors.
Neuronal Networks are fitting in the most problems pretty well and we liked it already in the PS, we chose it as our first model. After trial and error with the prebuild classifiers like decision trees, linear discriminant analysis, support vector machine and k-nearest neighbors we have got the best leaderboard score with logistic regression and so we wanted to implement this as our second model.

III. Results

Our Logistic Regression model performed better than the neural network with a score improvement of about 3%. For multiple runs it received a score of about 96.5% to 99.2%. We used Random undersampling for our logistic regression model to increase the training speed. Since undersampling returns a subset of the training data the resulting score has some variance depending on the selected elements. To reduce this fact we tried to use more data from the majority class. Therefore we set the sampling rate to 0.2 which means 20% of the data is from the minority class. It still has some variance but in average it performed better than the initial 50%. In the graphic you can see the progress of finding the best step and sampling size.



In the NN, we realized that if we did too many epochs or the NN was too small it began to overfit. An indicator to see this is when the train and test set have a big divergence. To overcome this issue we decided on 3 epochs and a learning rate of 0.0002 to find the optimal weights. SMOTE (Synthetic Minority Oversampling Technique) was used to make a good fitting set for our NET as a NN can handle much data. In the end for our submission we used the LR, because after modifying it a bit more we had a better score than the NN.



IV. Discussion

With resampling of the data we had a big progress in our accuracy, as we mentioned before that our data was totally imbalanced and after finding a way to make the existing classes more evenly distributed we made a big step for a good prediction. We tried number of hidden layers from 1 to 3 with different sizes of neurons for the Neuronal Network but our best result was about 95%. Therefore, we switched to logistic regression which performed better. Some mistake was that we didn't save the trained standardscaler with the model, because this is also necessary for the prediction function. If we don´t use the same one as we used before for our training set we had a large miss in the leaderboard score. Without using the standardscaler for prediction we got just a score of about 55%. At the beginning we also tried to reduce the dimensions by using LDA or PCA, but for our model it didn´t make a big difference in the score, so we discarded the idea. For the NN we would try some more combinations of layers and neurons although we already tried many. When reading some articles, we also found that an isolation- or random forest could work very well. But we decided on not implementing that since it was not covered in the lecture. A problem we faced was that we misunderstood the predict function. Initially, we returned the predicted class label instead of the probability being fraud. Fixing this improved our logistic regression model by nearly 8% and we think this could also improve the score of other classifiers like LDA but we ran out of time implementing it.

V. Conclusion

With the seminar and the project we got a small overview of the ML field. The main takeaway for us was the hands on mentality, this means we learned many ML algorithms and concepts what we had to proof in the project that we understood the main facts about it. With not just using implemented classifiers it made it much more detailed, so we had to think for each case why we used this algorithm or have chosen this way instead of another. In this project paper, data pre-processing, normalization and resampling carried out to overcome the problems faced by using an imbalanced dataset.

REFERENCES

[1] Haibo He: Imbalanced Learning: Foundations, Algorithms, and Applications 1st Edition

[2] WWW Reference: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ (26.06.2021, 12:00)

[3] Hala Z Alenzi, Nojood O Aljehane: Fraud Detection in Credit Cards using Logistic Regression, https://thesai.org/Downloads/Volume11No12/Paper_65-Fraud_Detection_in_Credit_Cards.pdf (26.06.2021, 14:00)

[4] WWW Reference: https://towardsdatascience.com/explain-your-machine-learning-with-feature-importance-774cd72abe (26.06.2021, 18:00)

[5] WWW Reference: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/ (28.06.2021, 10:00)

[6] WWW Reference: https://www.kaggle.com/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets (15.05.2021, 11:00)

[7] **Our final code with some more inforamation and some features: https://www.kaggle.com/merase/notebooka3f98eebce?scriptVersionId=67126039**

APPENDIX