Data Exploration Documentation
July 16, 2020

Mobile Payment Fraud Detection using Machine Learning

I. Overviewing the Random Forest Algorithm

Bjarne Gerdes, Raphael Schmidt, Anabel Lilja, Georg Schieck

Fakultät Wirtschaft, Duale Hochschule Mannheim (DHBW), Coblitzallee 1-9, DE-68163 Mannheim
e-mail: `info@dhbw-mannheim.de`

Received July 17, 2020

ABSTRACT

Context. Since mobile payment and credit card fraud is a very costly factor in the payment industry a machine learning algorithm to detect such fraudulent actions is to be developed and used to make accurate predictions.
Aims. To train a machine learning model based on a synthetically generated data set which was simulated based on a sample of real world transactions. Using the trained algorithm enables banks and mobile payment companies to detect fraudulent transactions as soon as they happen so actions against further abuse of the payment medium can be taken.
Methods. Building an initial model using the synthetic data set containing more than six million transactions. Further improving of the initial model using the method of feature engineering. Given the domain knowledge the most informative features are extracted from the raw data to increase the performance of the algorithm. Based on the chosen features an evaluation and necessary adjustments are applied to the built model. Implementation of the optimized model into an API which is connected to a web interface for a final integration into the business process.
Results. Using the final model predictions whether the transaction is or is not fraudulent are made based on the amount of the transferred currency.

Key words. mobile payment fraud – random forest algorithm

1. Introduction & Motivation

Mobile payment fraud is a very costly factor in the payment industry. Especially companies with high transaction volumes suffer immense losses. In 2017 PayPal reported a loss of 1011 Million USD due to Fraud and transaction losses. [2][3]

But what exactly is mobile payment fraud? And why is it relevant? It is defined as a fraudulent transaction that was initiated by a person that was not authorized to do so. The transaction was made on their own account or on someone else's account while the actual accountholder and its provider are not aware of it at the time of its realization. [1]

The relevance of the chosen topic is clear due to the high amount of banking transactions in our modern world. New technologies and an ever-present change in consumer behaviour result in millions of transactions occurring every day. At this point the method of completing transactions by using a credit card or a mobile device is the most predominate way to pay in our everyday life.

While the banks and financial institutes are constantly trying to develop new techniques to detect and even prevent fraud, the fraudsters are also adapting and trying to invent new ways to bypass the protective measurements.

The example mentioned above, about PayPal stating a loss of multiple millions in a single year is a great motivation to invest time and money into developing a machine learning algorithm which enables financial institutes to predict possible fraudulent transactions. By using a powerful and well-optimized machine learning model they should be able to prevent the loss of money and increase their overall customer satisfaction.

2. Related Work

So what exactly did financial institutes and banks do to detect and prevent fraudulent transactions up to this point? What machine learning methods and algorithms were already used to tackle this problem?

Given the urgency and the relevance of the problem at hand there are already a lot of publications and articles about machine learning solutions used for fraud detection. The machine learning algorithms used most commonly in previous research were:

- naive Bayesian,
- decision trees,
- k-nearest-neighbor,
- logistic regression,
- and random forests.

Husejinovic (2020) investigated the problem using a combined approach existing of a naive Bayesian classifier and a C4.5 decision tree. Using precision, recall and PRC area rates as evaluation criteria he claimed to achieve an accuracy of 82.5% for the bagging ensemble learner. [4]

In 2018 a team from Bharati Vidyapeeth'S College of Engineering also went with an ensemble method approach combining the use of a Naive Bayes model and a KNN classifier. The article claims that the task was managed with an overall precision of approximately 90 - 95% while also stating that the problem of fraud detection could only be solved efficiently if multiple algorithms were combined. [5]

M. Suresh Kumar (2019) and his collegues from the Sri Sairam Engineering College proposed a system in which they used a Random Forest Algorithm (RFA) to detect fraudulent transactions in their dataset. The result of the classification was then evaluated using a confusion matrix to achieve an accuracy of the model of approximately 90%. [6]

Earlier in 2018 S. Patil et al. also tackled the problem using a Random Forest Algorithm while also considering the use of a logistic regression model for the classification of fraud detection. Evaluation of the proposed models using a confusion matrix results in a precision of 76% for the logistic regression model and 89% for the Random Forest Algorithm. [7]

3. Data set & Pre-processing

The data set used for the proposed system is based on a real world data set obtained from an unnamed company based in Africa. The company basically offers electronic wallets which enable customers to make mobile and digital transactions. The data set stimulates transactions based on a sample of real transactions from one month of financial logs.

The set was obtained from the website kaggle.com, the world's largest data science community with powerful tools and resources. It allows users to find and publish data sets and even build models in a web-based data-science environment. It provides the opportunity to work alongside other data scientists and machine learning engineers. [9]

The data set is available as an CSV-File and consists of more than six million transactions. It is made out of eleven entities overall which are explained in the following table:

Step	Unit of time in the real world (One step is one hour)
Type	Type of Transaction (i.e. CASH-IN, CASH-OUT, etc.)
Amount	Amount of Transaction in local currency
nameOrig	Customer starting the transaction
oldbalanceOrg	Customers balance before the transaction
newbalanceOrg	Customers balance after the transaction
nameDest	Recipient ID
oldbalanceDest	Recipient balance before the transaction
newbalanceDest	Recipient balance after the transaction
isFraud	Identifies a fraudulent transaction
isFlaggedFraud	Flags illegal attempts to transfer too much money

Table 1: Entities of the data set

To properly train the model the data set was split into a subset for training purposes and a testing-subset used for evaluation. The training-subset consists of 80% and the testing-subset of 20% of the original data set.

To prevent the training-subset from containing information which might affect the test-data that is to be interpreted, the subsets were not created on a random basis but rather picked by hand. This is done to avoid data leakage, which can lead to the creation of an overly optimistic or even completely invalid predictive models. [8]

Since most of the transactions are non-fraudulent the data set is affected by a certain imbalance where one class occurs more often than any other class in the set. This is a typical problem for most real world data science scenarios.

To obtain additional information from the chosen data set the data was pre-processed using Feature Engineering. As a result of pre-processing the following features were left unchanged and used in their original state:

– Amount
– oldbalanceOrg
– newbalanceOrg

The entity called "Type" was furthermore chosen to be a dummy variable since the type of a transaction is not very informative for a prediction of the fraudulent character. Additionally some new features which described the transaction history between the recipient and the customer authorizing the transaction were created and tested on the model. But because he importance of those features were found to be too insignificant they have been discarded again.

The following table displays the features which were obtained by performing transformations on the original data:

PctChangeOrg	Percentage change of the customers balance
HourOfTheDay	Daytime in hours (obtained from the "step" entity)
relativePctTxToDest	Fraction of all transaction made towards this recipient

Table 2: Features obtained after transformation

4. Methods & technologies

4.1. Logistic Regression

The Logistic Regression is a commonly used algorithm to find solutions to a classification problem and was suggested to create a basis for the proposed model. It is used for binary classification of either 1, in this case the fraudulent transaction or 0, the non-fraudulent transaction. [10]

To achieve the classification of either 0 or 1 the logistic regression makes use of the sigmoid function:

$$y^i = \frac{1}{1 + e^{-z}}, \tag{1}$$

which is a type of squashing function, meaning the function will limit the output to a range of 0 to 1.

Furthermore a combination of grid search and cross-validation was used to optimize penalty values and the parameter λ which influence the regularization that is performed during the gradient method.

4.2. Support Vector Machines

The same optimization-parameters were used to perform a additional grid search onto a linear Support Vector Classifier. A Support Vector Machine or SVM for short is a two-class classification model. The SVM's principle is to build a model that assigns new examples to one class or the other given a present set of training data. The resulting model is a representation of the examples as points in space, mapped in a way so that the examples of the separate classes are divided by a clear gap which is chosen to be as wide as possible. [11]

4.3. Random Forest Classifier

Also called the Random Forest Algorithm, belongs to the supervised machine learning methods where the teacher provides the input and the associated labels while training the model. It is a type of ensemble learning, meaning that multiple individual algorithms are joined together to form a more complex and powerful model.

In the case of the RFA we combine multiple algorithms of the same type. The Random Forest consists of an immense amount of individual decision tree classifiers. Each of those decision trees will give its class prediction as an output. The class with the highest score in the end will be the models final prediction. [14]

The Random Forest Classifier was chosen to build a third model to take on the task of fraud detection in the financial sector. In order for the model to work efficiently it was necessary to optimize following parameters:

- maximum tree-depths
- number of maximum features for each tree
- minimum amount of samples for each split and leaf
- and the number of trees

In general there were 95 parameter pairs which were used during the parameter fine-tuning. The best-performing pair of each algorithm was used to finally train a model on the complete data set. There are several advantages when working with a Random Forest Algorithm:

1. Very flexible, can be used for classification and regression.
2. The algorithm is not biased, because there are multiple trees with each tree being trained on a subset of data. So the Random Forest relies on the power of "the crowd" which will reduce the overall bias in the model.
3. It's a very stable algorithm, because even if new data points are introduced to the data set it is very unlikely that those new examples will have any impact on the overall forest. They will only affect a few trees which will hardly influence the final prediction.
4. Generally the Random Forest Algorithm works very well when there are both categorical and numerical features in our data set.

4.4. Technology & Libraries

The following section is used to name and describe all libraries used during this project to build the proposed models and algorithms:

numpy	A general-purpose array-processing package. Provides tools for working with arrays.
pandas	Flexible, easy to use open source tool to analyze and manipulate data.
scikit-learn	Contains a lot of efficient tools used for machine learning and statistical modeling.
joblib	Consists of a set of tools which are used to provide lightweight pipelining in python.
sqlalchemy	The Python SQL toolkit and Object Relational Mapper that provides full power of SQL.

Table 3: Libraries used to build the models

To also enable the customer to quickly check if a new transaction is fraudulent or not a web-interface was built for easy and fast predictions. In order to create this customer-friendly interface the following technologies and libraries were used:

Flask API	A web framework providing the developer with tools to build a web application.
Docker	Provides the ability to package and run an application in a container.

Table 4: Libraries used to build web-application

4.5. Frontend-Backend-Communication

The following schema is used to further explain the communication between the frontend web-interface and the data storage on the backend site.

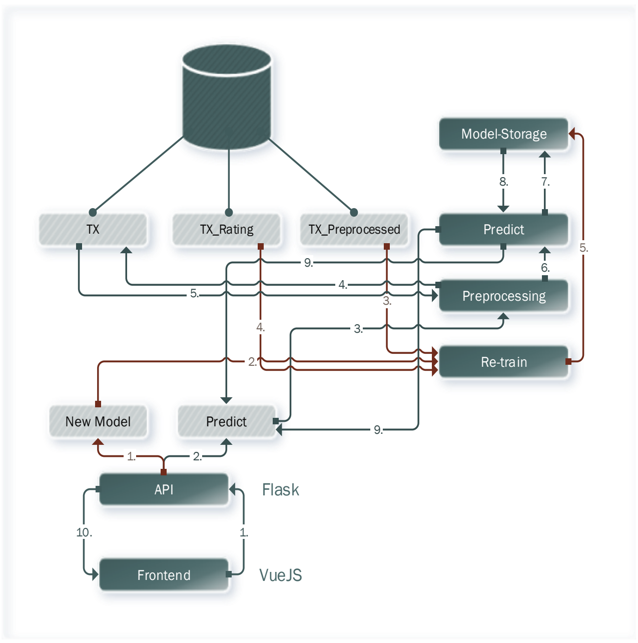

Fig. 1: Architecture of Front- and Backend

Before a thorough explanation of the communication schema the different parts of the architecture should be mentioned and described:

Frontend	Frontend Web-Interface created with the help of vueJS.
API	Built by using the Flask toolkit to initiate the communication.
Database	The Database on the backend-site was built using the open-source postgre database.

Table 5: Elements of the communication architecture

At first the process of making a request for new data - as shown by the dark-grey arrows - is explained following all the steps:

1. A request for a new transaction is made on the frontend web-interface which starts the communication to the Flask-API
2. The API then addresses the Predict-Class to initiate the process of making a new prediction.
3. New transaction is passed on to the pre-processing step.
4. In pre-processing the data will be inserted into our already known data.
5. Database will then forward the updated data set back to the pre-processing class.
6. After pre-processing the information is passed on so a prediction can be made.
7. The prediction based on the new data is stored inside the Model-Storage.
8. Model-Storage back-passes the information to the Predict-Class.
9. Finally the prediction will be relegated back to the API.
10. The API prints out the prediction made on the new transaction to the user on the web-interface.

The second process of creating a new model is described by following the dark-red arrows in the schema shown above:

1. A process for creating a new model is initiated by the Flask-API.
2. The request is forwarded from the New Model-Class to the Retrain-Class.
3. During retraining of the model the data-rating is obtained from the database.
4. Furthermore the pre-processed data which is already known is forwarded to the Retrain-Class.
5. The newly trained model is passed on and will be stored in the Model-Storage.

5. Evaluation

As it was mentioned before the data set for this real world problem is imbalanced due to the non-fraudulent class occurring more often than the fraudulent transactions. To properly evaluate the model with such an imbalanced set of data the F1-Score was chosen to do so. Additionally a cross-validation was performed on two disjoint subsets of the training data. The result of the evaluation is displayed within the following table:

Algorithm	**F1**	**Average Precision**	**AUC-Score**
Random Forest	0.9998	0.9995	$0.\overline{9999}$
Logistic Regression	0.6535	0.4530	0.9634
Support Vector Machine	0.8820	0.7838	

Table 6: Evaluation of the built models

It should be anticipated that after the above seen table of evaluation was acquired it was again checked if the testing- and the training-subset of the data are in fact disjoint. Furthermore the evaluation obtained using the above mentioned methods was compared to the evaluation results other data scientists posted to kaggle.com. [12][13]

Due to the comparison it became clear that the result of the Random Forest Algorithm seems to be plausible. It is also worth mentioning that none of the observed kernels on kaggle.com achieved a better target metric than the Random Forest built during this project.

6. Conclusion

In this last section the evaluation result and the potential of the model will be discussed. For the model to be relevant it needs to be able to properly classify fraudulent transactions so a financial institute can intervene quickly enough before a monetary loss is finalized. Given the feature "ifFlaggedFraud" it is possible to historically compute the success for fraud-prevention.

Especially the monetary potential of the Random Forest Algorithm (RFA) should be mentioned here since it performed extremely well compared to kernels already published by different data scientists around the world.

The total loss induced by fraudulent transactions within the testing-subset comes to the amount of 6.685 billion USD. Overall only 70 million USD were historically blocked which is only 1.06% of the total amount.

Compared to that the proposed Random Forest was able to detect and block transactions with an summed up amount of 6.684 billion USD which represents 99.99% of the overall loss caused by fraud. The monetary value saved by the use of this Random Forest results in the difference between the data and the model which is approximately 6.613 billion USD.

References

[1] Askari.S and Hussain.A, 2017, Credit Card Fraud Detection Using Fuzzy ID3, IEEE, Computing, Communication and Automation (ICCCA), 446-452

[2] https://www.ftc.gov/system/files/documents/reports/consumer-sentinel-network-data-book-2018/consumer_sentinel_network_data_book_2018_0.pdf (Zugriff am 15.07.2020)

[3] https://de.statista.com/statistik/daten/studie/1100324/umfrage/bekanntheit-von-zahlungsverfahren-im-internet-in-deutschland/ (Zugriff am 15.07.2020)

[4] Husejinovic, Admel. (2020). Credit card fraud detection using naive Bayesian and C4.5 decision tree classifiers. 8. 1-5. 10.21533/pen.v%25vi%25i.300.

[5] Kiran, S., Guru, J., Kumar, R., Kumar, N., Katariya, D., Sharma, M.P. (2018). Credit card fraud detection using Naïve Bayes model based and KNN classifier. International Journal of Advance Research, Ideas and Innovations in Technology, 4, 44-47.

[6] M. S. Kumar, V. Soundarya, S. Kavitha, E. S. Keerthika and E. Aswini, "Credit Card Fraud Detection Using Random Forest Algorithm," 2019 3rd International Conference on Computing and Communications Technologies (ICCCT), Chennai, India, 2019, pp. 149-153, doi: 10.1109/ICCCT2.2019.8824930.

[7] Patil, Suraj Nemade, Varsha Soni, Piyush. (2018). Predictive Modelling For Credit Card Fraud Detection Using Data Analytics. Procedia Computer Science. 132. 385-395. 10.1016/j.procs.2018.05.199.

[8] https://machinelearningmastery.com/data-leakage-machine-learning/ (Zugriff am 15.07.2020)

[9] https://www.kaggle.com/ntnu-testimon/paysim1 (Zugriff am 15.07.2020)

[10] O. Adepoju, J. Wosowei, S. lawte and H. Jaiman, "Comparative Evaluation of Credit Card Fraud Detection Using Machine Learning Techniques," 2019 Global Conference for Advancement in Technology (GCAT), BANGALURU, India, 2019, pp. 1-6, doi: 10.1109/GCAT47503.2019.8978372.

[11] G. Cheng and X. Tong, "Fuzzy Clustering Multiple Kernel Support Vector Machine," 2018 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR), Chengdu, 2018, pp. 7-12, doi: 10.1109/ICWAPR.2018.8521307.

[12] https://www.kaggle.com/netzone/eda-and-fraud-detection (Zugriff am 15.07.2020)

[13] https://www.kaggle.com/arjunjoshua/predicting-fraud-in-financial-payment-services (Zugriff am 15.07.2020)

[14] Meenakshi, D., Indira, M. (2019). CREDIT CARD FRAUD DETECTION USING RANDOM FOREST.

Appendix

Locally execute the code

To locally execute the code and make use of the proposed machine learning model:

1. Navigate to the folder containing the dockerfile using a terminal.
2. Execute the command "docker build -t de_api".
3. Use the command "docker run -d -p 1213:1213 de_api" to run the container.
4. The endpoints are: http:0.0.0.0:1213/post/tx
5. And the payload is: Payload: {"amount" : [float], "type": ["PAYMENT", "TRANSFER", "CASH$_{I}N", "DEBIT", "CASH_{O}UT"], "nameOrig" : [nameOrig(ktonr.ausDB)], "nameDest" : [nameDest(ktonr.ausDB)]\}$.
6. Use http:0.0.0.0:1213/post/model as an endpoint to train a new model. This can take some time and will return the duration needed to training.

Use of the server-based version

If instead the version stored on the server is used, the endpoints "http://h2655330.stratoserver.net:1213/post/tx" and "http://h2655330.stratoserver.net:1213/post/model" are needed. The payload remains the same as stated above.

Locally run the website

1. Navigate to the folder containing the package.json using a terminal.
2. Execute the command "npm install" (node.js needs to be installed).
3. Run the website using the command "npm run dev".

Notes:

1. It is recommended to use the website at "http://fraud-detection.schmidt-development.de/".
2. "http://h2655330.stratoserver.net:1213/post/model" will not work on the server because not enough RAM is provided.