Machine Learning Engineer Nanodegree

Capstone Project: Supervised Learning for Fraud Detection

Robert Heyse
September 28, 2019

I. Definition

Project Overview

Fraud is causing companies, and private persons, to lose billions of dollars, with the amount of companies exposed to fraud increasing continuously.[1] Fraud prevention is one important measure to combat this problem, however this might fail since fraudsters can adapt to different prevention systems. Therefore, methods to detect fraud once prevention has failed are necessary. Machine learning is an effective technology to detect fraudulent activities such as money laundering, credit card fraud, computer intrusion, or online payment fraud.[2] In this project, I will focus on detecting the latter.

Statistical fraud detection methods can be distinguished between supervised and unsupervised techniques. In supervised methods, labelled data of fraudulent and non-fraudulent transactions are used to build and train a model that predicts whether a given transaction is fraudulent. Unsupervised methods, on the other hand, seek accounts, customers or transactions that are dissimilar from the norm and flag them for closer inspection. Some techniques that have been historically applied to detect fraud range from linear discriminant analysis, to tree-based algorithms, to neural networks.[3]

In this project, I will apply supervised learning to build a model that predicts whether a given transaction is fraudulent.

The dataset for this project is obtained from the IEEE-CIS Fraud Detection Competition on Kaggle.[4] The data for the competition has been provided by Vesta, one of the leading e-commerce payment solution providers, guaranteeing more than $18B in transactions

---

[1] https://en.wikipedia.org/wiki/Data_analysis_techniques_for_fraud_detection

[2] http://metalab.uniten.edu.my/~abdrahim/ntl/Statistical%20Fraud%20Detection%20A%20Review.pdf

[3] http://metalab.uniten.edu.my/~abdrahim/ntl/Statistical%20Fraud%20Detection%20A%20Review.pdf

[4] https://www.kaggle.com/c/ieee-fraud-detection/data

annually. It comes from real-world e-commerce transactions and contains a wide range of features from device type to product features.

Problem Statement

Fraud detection in online payment transactions is a key problem for payment service providers. The ambition of payment service providers when implementing a fraud detection system is to identify and flag all fraudulent transactions, while keeping the false positive rate low. By flagging all fraudulent transactions, businesses will minimize their fraud loss and increase their revenue. A high false positive rate would have a negative effect on customer experience, as transactions might be blocked or an additional effort to verify a transaction might be required.[5]

To solve the problem at hand, I will apply supervised learning algorithms on the Vesta dataset. The labeled data in the training set will serve to train the algorithm. The input of the algorithm will be a combination of the categorical and numerical features in the dataset. The output of the algorithm will be the probability that a given online transaction is fraudulent. I will use an evaluation metric to measure the performance of different algorithms against each other.

The strategy to reach this solution will be to use exploratory data analysis to get a better understanding of the features, followed by data preprocessing and feature engineering, followed by training different models on the data and using them for predictions.

Metrics

It is important to implement a machine learning model that correctly identifies fraudulent transactions, while keeping the false positive rate low. The evaluation metric will therefore be the area under the ROC curve (AUROC), which will take both the true positive rate and the false positive rate into account.[6]

When plotting the ROC curve, the true positive rate is plotted on the y axis against the false positive rate on the x axis. The true positive rate in this problem is the percentage of fraudulent transactions that have been correctly identified as such. The false positive rate in this problem is the percentage of non-fraudulent transactions that have been identified as fraudulent. A diagonal ROC curve represents a random process, and would be equivalent to an AUROC of 0.5. The maximum achievable AUROC would be 1. I will keep this in mind when interpreting the evaluation results.

---

[5] https://www.kaggle.com/c/ieee-fraud-detection/overview/description
[6] https://en.wikipedia.org/wiki/Receiver_operating_characteristic

II. Analysis

Data Exploration

The dataset for this project is obtained from the IEEE-CIS Fraud Detection Competition on Kaggle.[7] The data for the competition has been provided by Vesta, and comes from real-world e-commerce transactions and contains a wide range of features from device type to product features.

The data is broken into two files, `identity` and `transaction`, which I am later joining on `TransactionID`. Not all transactions have corresponding identity information.

The following features are part of the `transaction` table:[8]

- TransactionDT: timedelta from a given reference datetime (not an actual timestamp)
- TransactionAMT: transaction payment amount in USD
- ProductCD: product code, the product for each transaction
- card1 - card6: payment card information, such as card type, card category, issue bank, country, etc.
- addr: address
- dist: distance
- P_ and (R__) emaildomain: purchaser and recipient email domain
- C1-C14: counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.
- D1-D15: timedelta, such as days between previous transaction, etc.
- M1-M9: match, such as names on card and address, etc.
- Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations.

Categorical Features:

- ProductCD
- card1 - card6
- addr1, addr2
- Pemaildomain Remaildomain
- M1 - M9

In the `identity` table, network connection information, and digital signature associated with transactions are found. They are collected by Vesta's fraud detection system and digital security partners. The following categorical features are part of the `identity` table:[9]

- DeviceType

[7] https://www.kaggle.com/c/ieee-fraud-detection/data
[8] https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203
[9] https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203

- DeviceInfo
- id12 - id38

The transaction data is labelled with the variable `isFraud`, indicating whether a transaction has been reported as a fraud or not. I will use the dataset to build a supervised learning model that predicts the probability whether a given online transaction is fraudulent.

To start off the data exploration, i am visualizing some statistics on the joined data set. The table below shows 16 columns, together with their data type, missing, as well as unique values.

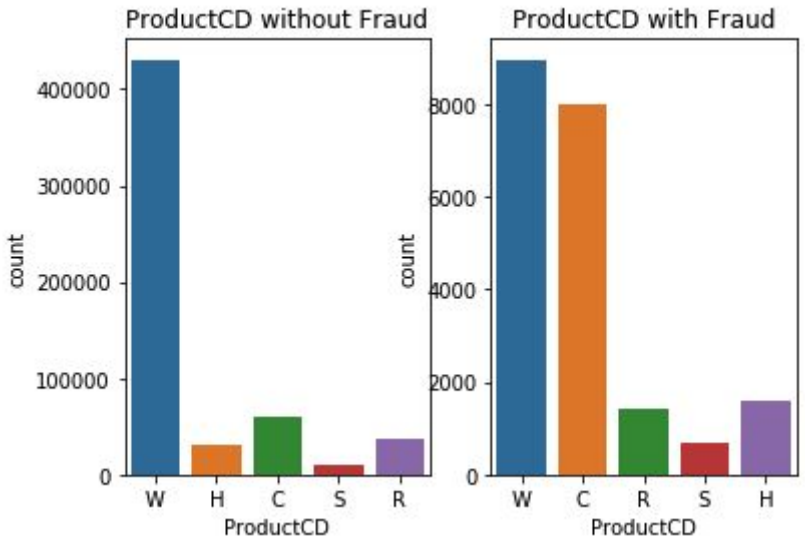
	Name	dtypes	Missing	Uniques	First Value	Second Value	Third Value
0	TransactionID	int32	0	590540	2987000	2987001	2987002
1	isFraud	int8	0	2	0	0	0
2	TransactionDT	int32	0	573349	86400	86401	86469
3	TransactionAmt	float16	0	8195	68.5	29	59
4	ProductCD	object	0	5	W	W	W
5	card1	int16	0	13553	13926	2755	4663
6	card2	float16	8933	500	NaN	404	490
7	card3	float16	1565	114	150	150	150
8	card4	object	1577	4	discover	mastercard	visa
9	card5	float16	4259	119	142	102	166
10	card6	object	1571	4	credit	credit	debit
11	addr1	float16	65706	332	315	325	330
12	addr2	float16	65706	74	87	87	87
13	dist1	float16	352271	2412	19	NaN	287
14	dist2	float16	552913	1699	NaN	NaN	NaN
15	P_emaildomain	object	94456	59	NaN	gmail.com	outlook.com
16	R_emaildomain	object	453249	60	NaN	NaN	NaN

This gives us insights on what kind of data preprocessing we will need to apply later on. As the table shows, there are both numerical and categorical features in the data set. We will use techniques such as label encoding or one hot encoding to make the categorical features digestible for the algorithms to be applied.
Moreover, there is a high amount of missing values for many variables. We need to consider that some algorithms cannot process those. Later on, we will have to fill those missing values or eliminate certain features altogether.

There are certainly some outliers in the data. I will refrain from removing them as I want to find anomalies in the data.

Exploratory Visualization

In the graph below, I visualized the feature ProductCD with regards to transactions marked as fraud and those that are not fraudulent. While it is not clearly explained what ProductCD signifies, we know that it is a banking service, and not a physical product. From the graphs, we can see that there ProductCD = C stands out when it comes to fraudulent transactions. It looks like this has a positive influence on the probability that a transaction is fraudulent.



Algorithms and Techniques

I will apply Light GBM and XGBoost, two supervised learning algorithms, on the dataset at hand. The labeled data in the training set will serve to train the algorithm. The input of the algorithm will be a combination of the features described above.

XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems, and is often seen in kaggle competitions as the winning algorithm. It is an implementation of gradient boosted trees designed for speed and performance.[10] It has multiple hyperparameters, which can be tuned and optimized. A list of XGBoost hyperparameters can be found here: https://xgboost.readthedocs.io/en/latest/parameter.html

Light GBM is a gradient boosting algorithm, based on decision trees, which can be even faster than XGBoost. Due to its leaf-wise growth, which makes it stand out among other boosting algorithms, a high accuracy can be achieved. Light GBM is very fast when

10 https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

compared to XGBoost and is therefore well-suited for large datasets.[11] Light GBM has many parameters, which can be tuned as described in the official documentation: https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html

It might be necessary to modify certain features, to combine or to split them. The techniques used for data preprocessing are described in the section *Data Preprocessing* of this paper.

The output of the algorithm will be the probability that a given online transaction is fraudulent.

I will use the AUROC metric to measure the performance of the different algorithms against each other.

Benchmark

There is a wide range of supervised learning algorithms that can be applied for fraud detection purposes.[12] One popular method that is being applied in the field are decision trees. Decision trees are easy to interpret, they are able to handle categorical features, do not require feature scaling, and are able to capture non-linearities and feature interactions . Tree ensemble methods, such as random forests, are often among the top performing algorithms for classification tasks.[13] Decision trees and their results are clearly measurable. Their output can be categorical or probabilistic. Their performance can be measured with the AUROC metric.

For this problem, I would like to use the solution published by another kaggle user here: https://www.kaggle.com/yoongkang/beginner-s-random-forest-example

In this solution, a random forest model has been used on the data to achieve an AUROC score of 0.9158. I will measure my models against this score in order to determine the best model.

III. Methodology

Data Preprocessing

After merging the transactions and identity datasets, I reduced the memory of the dataset by transforming the data types in the numerical columns of the dataset.

---

[11] https://towardsdatascience.com/lightgbm-vs-xgboost-which-algorithm-win-the-race-1ff7dd4917d

[12] https://arxiv.org/pdf/1611.06439.pdf

[13] https://pages.databricks.com/rs/094-YMS-629/images/financial-fraud-detection-decision-tree.html

As a next step, I am creating bins for the e-mail addresses in the columns P_emaildomain and R_emaildomain, as they can be merged into categories, which improves the model score.

For XGBoost, it isn't always necessary to deal with missing values. XGBoost can decide for each sample, which is the best way to impute missing values.[14] In the Light GBM algorithm, this can lead to overfitting, as NaN values get special treatment at every node. I am circumventing this issue by converting all NaN values to negative values lower than all non-NaN values.[15]

For the categorical variables, I am employing label encoding. Label encoding converts the categorical columns to integers, which makes it possible to for the algorithms to deal with them.

Additionally, I created feature aggregations on top features. Providing the algorithm with group statistics allows it to determine if a value is common or rare for a specific group.[16]

In the last iteration, I added the decimals of the transaction amount as a feature. Moreover, I added the day of the week and the hour of the day as features.

Implementation

The workflow structure I employed when solving this problem is as follows:

1. Data loading and overview: I got a first glance at the data, checked how many columns there are, checked for missing values, ...
2. Data exploration: I analyzed different features and their distributions
3. Feature engineering: I chose to engineer new features or change the existing ones
4. Data preparation: I need to bring the data to the right input format for modelling.
5. Model training: I trained different models on the data
6. Prediction: I used the trained models to do predictions on the data
7. Model evaluation: I evaluated the models based on the AUROC metric.

In the first step, I got an understanding of the data. The joined training dataset is quite large and complex, with 590540 rows and 434 columns. The high amount of features makes the data exploration, feature engineering and data preparation parts complex and require a lot of work.

In the data exploration part, I visualized different features using the matplotlib and seaborn libraries, in order to get a better understanding of what they mean. Besides the feature

[14] https://towardsdatascience.com/xgboost-is-not-black-magic-56ca013144b4
[15] https://www.kaggle.com/c/ieee-fraud-detection/discussion/108575#latest-635072
[16] https://www.kaggle.com/c/ieee-fraud-detection/discussion/108575#latest-635072

descriptions in the data exploration part of this paper, I didn't receive any information on what the features signify. Feature selection was therefore a challenge.

I described the process of feature engineering and data preparation in the Data Preprocessing part of this paper. This is a crucial part of the project, as it probably has the highest influence on the end results.

Finally, I trained XGBoost and LGB models on the preprocessed data. I fitted the models to the training data, and used them to make predictions on the test data.

In order to evaluate the models, I submitted the fraud predictions of the test data set to the official kaggle competition. Based on the AUROC score I received, I took decisions on how to further refine my models.

Refinement

For my first two iterations, I ran a XGBoost and LGB model on the same preprocessed data. I received AUROC scores of 0.9281 and 0.9110, respectively. While the score of the XGBoost model is higher than the benchmark score of the random forest classifier (0.9158), the score of the LGB model is slightly below. The scores are quite close to one another, so it wasn't clear that I could drop one of the models in order to focus on the other.

In order to refine the models, I made a decision to focus on feature engineering rather than hyperparameter optimization. The reason for that is that hyperparameter optimization is computationally expensive and takes a lot of time, while feature engineering has a higher influence on the final score than hyperparameter optimization. I took the hyperparameters for my models from published notebooks, where they have been optimized by other kagglers that are participating in the competition.

After using feature aggregations, e-mail bins, and label encoding in my first iteration, I started adding more engineered features to the project. In the second iteration, I added the decimals of the transaction amount as a feature. Moreover, I added the day of the week and the hour of the day as features. This improved my model score by 0.0019 to 0.9129.

Finally, I could improve the LGB model score to 0.9408 by using k-fold cross validation to train the model.

IV. Results

Model Evaluation and Validation

The final model was derived by comparing the AUROC scores of the different model iterations and improving the models based on this. The final model is the model with the

highest AUROC score, namely a LGB model, trained with k-fold cross validation, that resulted in a AUROC score of 0.9408.

The final parameters and features of the model are appropriate, as they have been evaluated through an iterative approach.

K-fold cross validation did not only improve the model score, but also served as a sensitivity analysis. Cross validation is primarily used in Machine Learning to estimate the performance of a model on unseen data.[17]

In order to train the LGB model, I used a 5-fold cross validation, since this has been shown empirically to yield test error rate estimates that suffer neither from high bias nor from very high variance. This means that the training sample was split into 5 approximately equally sized parts. The first fold is treated as a validation set, and the remaining k-1 folds as training set. For each fold, the validation set gets rotated and the remaining four data sets become the new training set.

The standard deviation of the AUROC score in the cross validation folds is 0.01. This means that the model generalizes well to unseen data. My conclusion is that the model is robust and the results found from the model can be trusted.

Justification

The final score of my LGB model is 0.9408, in comparison to the benchmark score of 0.9158. This means that my model is beating the benchmark by 0.025. Even though the benchmark model is based on less engineered features and on a simpler algorithm, this increase in AUROC score justifies the usage of the LGB model.

I would argue that the final solution is significant enough to have solved the problem. A perfect model would have a AUROC score of 1. My model has a 94.08% chance that it will correctly distinguish transactions between fraudulent and non-fraudulent. It would be interesting to benchmark this score to industry standards in order to make a better evaluation.

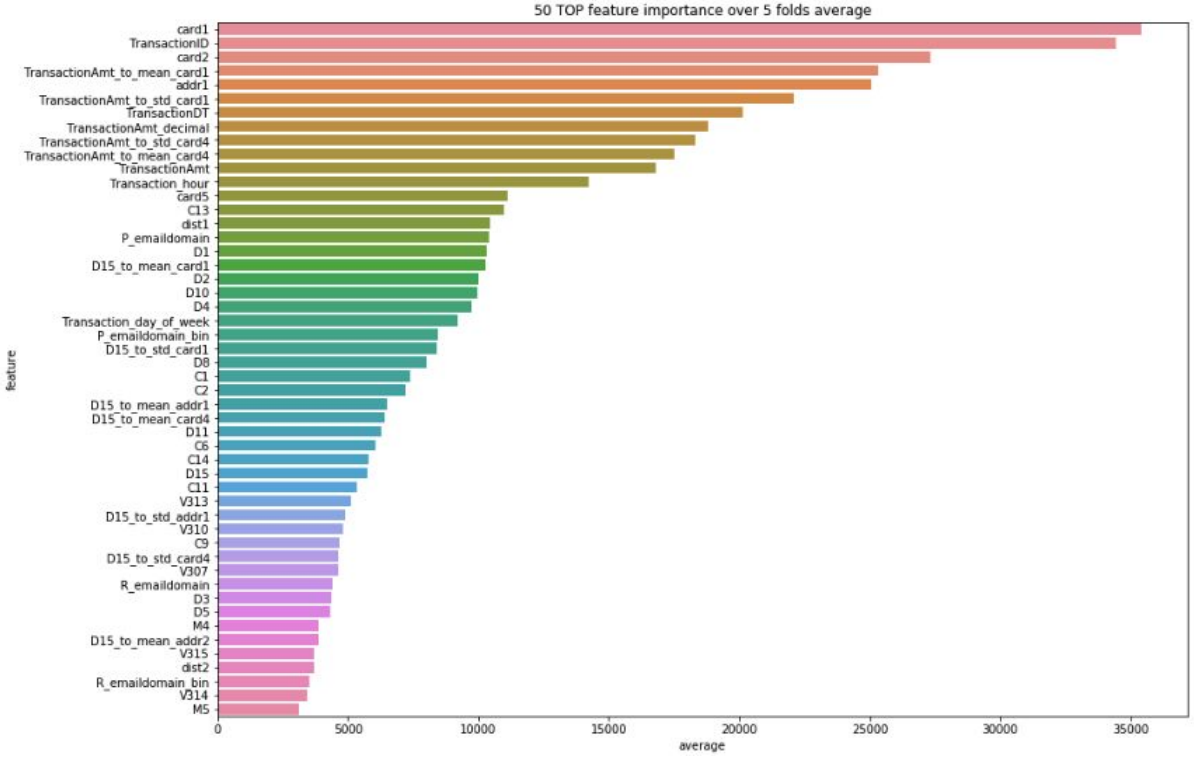
V. Conclusion

Free-Form Visualization

As a result of the final LGB model, I created a visualization of the 50 Top feature importances over the 5 folds average. It can be seen that the card features have a high feature importance, especially card1 and card2. Moreover, the engineered feature aggregations of TransactionAmt to card variables have a high influence on the prediction

[17] https://machinelearningmastery.com/k-fold-cross-validation/

outcome. This is a good way to test hypothesis made on different features from the Data exploration phase.



Reflection

The end-to-end problem solution can be summarized by the steps explained in the implementation section of this paper:

1. Data loading and overview: I got a first glance at the data, checked how many columns there are, checked for missing values, ...
2. Data exploration: I analyzed different features and their distributions
3. Feature engineering: I chose to engineer new features or change the existing ones
4. Data preparation: I brought the data to the right input format for modelling.
5. Model training: I trained different models on the data
6. Prediction: I used the trained models to do predictions on the data
7. Model evaluation: I evaluated the models based on the AUROC metric.

I found the feature engineering section the most interesting, since this requires a lot of creativity and good understanding of the data. Moreover, it has a high influence on the final result of the model. In a kaggle competition, a great feature engineering can make a submission stand out among others.

The most difficult part of the project was certainly to fully understand the data. There were more than 400 features in the Vesta data. To add to that, there were only basic descriptions of the features available. It was therefore hard to tell what the different features signify, and

what kind of influence they could have on whether a transaction is fraudulent or not. I could have made better decisions in the feature selection if I knew more about the features.

The final model and solution fits my expectations to the problem. A similar model and approach could be used in a general setting to solve these types of problems.

Improvement

One way to improve my final solution would be to use model ensembling. Ensemble models combine the decisions from multiple models to improve the overall performance. I could use this technique to combine my LGB and XGBoost models, for example. One simple ensembling technique for classification problems is max voting. Here, multiple models are used to make predictions for each transaction in the test dataset. The final result will be the mode of all predictions.[18]

Furthermore, the score could have probably been improved by doing a more sophisticated hyperparameter tuning. Bayesian optimization is a model-based method for finding the minimum of a function and can be applied to hyperparameter tuning, with better performance than random search. It uses a continually updated probability model to focus on promising hyperparameters by reasoning from past results.[19] Therefore, my suggestion for improvement would be to apply bayesian hyperparameter tuning to the final LGB model.

Finally, one way to further improve the model would be to engineer and test more features, by combining them, splitting them, creating aggregations or group statistics, for example.[20] It would have certainly helped to receive better explanations of the features in the Vesta dataset.

If my final solution would be used as a benchmark, I am quite certain that a better solution exists. As of 09/29/2019, the best score on the leaderboard of the kaggle competition is 0.9678.[21]

---

[18] https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

[19] https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a

[20] https://www.kaggle.com/c/ieee-fraud-detection/discussion/108575#latest-635072

[21] https://www.kaggle.com/c/ieee-fraud-detection/leaderboard