Capstone Project
Machine Learning Engineering
Nanodegree

Christopher Kim
October 14, 2019

IEEE-CIS Fraud Detection

I. Definition

Project Overview

At any given moment, millions of credit card transactions occur in the world. With the rise of electronic commerce businesses, the credit card transactions became even more common in the past couple of decades.

Recently, we've heard many cases of huge data breach cases involving major businesses. With such a massive number of compromised credit card numbers, fraudulent credit card transactions are not so rare occurrences anymore. In 2018, the Federal Trade Commission processed 1.4 million fraud reports totaling $1.48 billion in losses[1].

Preventing credit card fraud is an essential part of credit card payment processing. Improving fraud detection system will save a lot of money and improve consumers' experience.

In this project, I explored the data set provided by IEEE-CIS Fraud Detection competition[2] on Kaggle and apply feature engineering to prepare for machine learning algorithms. Using the prepared data, I applied machine learning algorithms to train the fraud detection model and evaluate the model for the meaningful metrics.

Problem Statement

Using Machine Learning technique to detect and prevent fraudulent transactions is already saving millions of dollars a year. Researchers from the IEEE Computational Intelligence Society (IEEE-CIS) want to improve this figure, while also improving the customer experience. On the Kaggle platform IEEE-CIS has opened a competition IEEE-CIS Fraud Detection competition[3].

Building a machine learning model that scores high accuracy on detecting fraud in transactions will be the problem to solve.

Metrics

As I explored the provided datasets, it turned out about 3.5% of the dataset is labeled as fraudulent transactions. If the system predicts every transaction to be normal transaction, the accuracy score would be 96.5%. Therefore, scoring high on the accuracy wouldn't necessarily be a good model for the problem solution.

Rather than using the accuracy to evaluate the model, I used the Confusion Matrix[4] and evaluated on Precision, Recall and F1 scores. Recall score gives us an idea about when it's actually true, how often the model predicts true. Precision score tells us about when it predicts true, how often it is correct. F1 score[5] is the harmonic mean of the Precision and the Recall scores. These measures would give much better confidence on the model that detects the fraud in transactions. Therefore, Precision, Recall and F1 scores would be good metrics to measure how well the model performs. I used the metrics to compare the final model against the benchmark model.

The competition submissions are evaluated on AUC(area under curve) on the ROC(receiver operation characteristics)[6] curve I have calculated the AUC on the test results of the selected learning model.

II. Analysis

Data Exploration and Visualization

Two sets of data are provided for the competition. One is the identity data and the other is the transaction data, and they are matched by TransactionID values. The transaction data set has the target label value of 'isFraud' on each row, and not every transaction has a matching row in the identity data set. There are about ¼ number of identity rows to the transaction data set.

Along with the mismatching transaction rows, most of the identity features have null values. Some of the features have null values in 99% of the rows.

However, the data sets have many feature columns, especially in the transaction data has 394 columns, and the identity data set has 41 columns. The identity and the transaction datasets have been joined on the TransactionID column, and the combined dataset has 434 feature columns.

Identity Data

Variables in this table are identity information – network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions. They're collected by Vesta's fraud protection system and digital security partners. (The field names are masked and pairwise dictionary was not provided for privacy protection and contract agreement.)

Categorical Features in the Identity Data:

The following features are identified as categorical features in the Identity data set, and the others are numerical values.

- DeviceType
- DeviceInfo
- id12 - id38

Transaction Data

The data in the Transaction dataset has transaction related columns such as transaction date, amount, product code, payment card information, engineered columns, and etc.

- 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 in Transaction Data:

In the transaction data set, the following features are identified as categorical data.

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

Ratio of normal transaction vs. fraud transactions

```
isFraud
0              0.96501
1              0.03499
```

About 3.5% of the transactions are fraud.

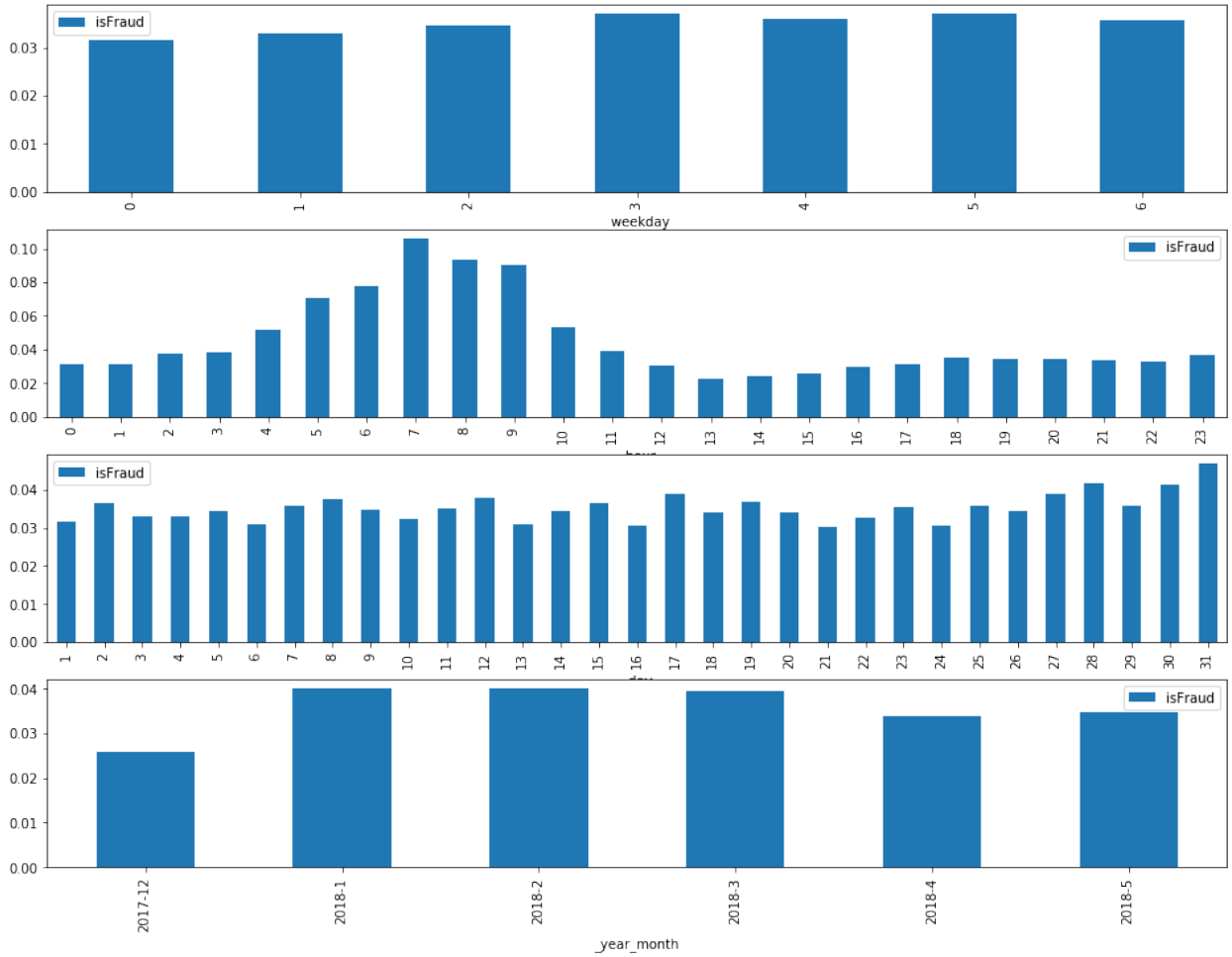
Null values

Some of the identity features such as id_07, id_08, id_21 - id_27 have null values for 99% of the data.

```
Id_07
 NaN     0.991271
 0.0     0.000693
 16.0    0.000415
...
```

Transaction Date

The following graphs show the distributions of the fraud transactions by weekdays, time of day, day of month, and month of year.



Algorithms and Techniques

Determining a card transaction to be fraud or not is a classification problem in supervised machine learning. There are many fine classification algorithms including Support Vector Machine, Decision Tree and many tree-based ensemble methods.

The SVM is a good algorithm for classification, but it takes a long time to train when there are more than handful of features. The Fraud Detection model has over 400 features and it will take too long to train using the SVM algorithm.

The Decision Tree is a simple tree-based decision-making algorithm to use for this problem. I decided to use the Decision Tree model to be the benchmark model to compare the performance of the selected algorithm for the problem.

Ensemble methods such as Random Forest, Bagging and Boosting improve performance on supervised learning. I have had a good result with the Scikit-Learn Gradient Boosting algorithm in one of the earlier projects. That led me to choose one of the Gradient Boosting algorithms. As I was gathering insights on how to solve the problem by examining the kernels that have been submitted to the Kaggle competition, I have noticed two such algorithms have been used by many kernels, which are XGBoost[7] and LightGBM[8].

I researched online to find the comparison between the algorithms[9]. I found that LightGBM paper uses XGBoost as a baseline and outperforms it in training speed and the dataset sizes it can handle. The accuracies are comparable. LightGBM in some cases reaches its top accuracy in under a minute and while only reading a fraction of the whole dataset. This goes to show the power of approximation algorithms and intelligently sampling a dataset to extract the most information as fast as possible. I decided to use the LightGBM for this project.

For LGBM algorithm the following parameters[10] can be tuned[11] for speed, accuracy, and dealing with over-fitting.

```
num_leaves (default=31)
min_data_in_leaf (default=20)
max_depth (default=-1, no limit)
bagging_fraction (default=1.0), bagging_freq (default=0, disabled)
feature_fraction (default=1.0)
max_bin (default=255)
save_binary (default=False)
learning_rate (default=0.1)
num_iterations (default=100)
lambda_l1, lambda_l2, min_gain_to_split (default=0.0)
```

Parameters used in the final model:

```
'learning_rate': 0.03,
'objective': 'binary',
'metric': 'auc',
'max_bin': 256,
'num_leaves': 256,
'min_data_in_leaf': 10,
'bagging_fraction': 0.85,
'bagging_freq': 10,
'feature_fraction': 0.9,
'max_depth': 128,
```

The model performed almost perfectly with the training data. To deal with the over-fitting I have set the `min_data_in_leaf, bagging_fraction, bagging_freq, feature_fraction` and `max_depth` values, which resulted with the best on the test. The bagging fraction and frequency randomly selects part of data, and the feature fraction randomly select part of the features on each iteration. This helped the model to speed up in training and deal with over-fitting.

Benchmark

As mentioned in the section above, I chose to compare the performance of my chosen algorithm LightGBM to the performance of basic decision-making algorithm Decision Tree. I have trained the model using the same set of training data and default hyper-parameters.

Decision Tree model Metrics

```
Confusion Matrix
true positive 2539, false positive: 3203
false negative: 1703, true negative: 110663
Precision: 0.442, Recall: 0.599
F1 score: 0.509

CPU times: user 1min 45s, sys: 945 ms, total: 1min 46s
Wall time: 1min 45s
```

III. Methodology

Data Pre-processing

For training and testing the model, I have applied feature engineering techniques to process the data set to be prepared for the learning model. As mentioned above, the data set has many null values, and they are substituted with mean value of the column.

Date values have been engineered into hour of day, day of week and day of month features. Some categorical features with many variant values such as Operating System and User Agent information are grouped into major categories. Eventually Label Encoding function has been applied to the categorical features to make the values become numerical values to be processed by the algorithm. More details of what I have applied to the data set have been described below.

As I don't have information about what each column data is, I didn't apply transformation on most of the columns. I have dropped only a few columns that are obviously not good features for the learning such as transaction date and transaction ID.

Reduce Memory Usage

When the CSV data files are read into Pandas data frames, the columns with numeric values are assigned with the widest data types in order to preserve the values, which became 64-bit integers and floating point values. However, many of the columns don't require such high precision data types.

During my research, I found many of the Kaggle competitors use some kind of memory saving routines to convert the data types into smaller precision data types in order to save memory space[12]. I have adopted one of the memory-reducing functions to save about 50% of the memory space between two data sets.

```
Memory usage of dataframe is 45.12 MB --> 25.86 MB (Decreased by 42.7%)
```

```
Memory usage of dataframe is 1775.15 MB --> 542.35 MB (Decreased by 69.4%)
```

Null Values

The null values for the numeric features are filled with the mean values of the column in preparation for the modeling.

Transaction Date

As the graph above in the Data Exploration section shows, Transaction Date of the Data Exploration and Visualization section shows, the fraud transactions are not uniformly distributed in each time frame. This makes the transaction time frame be a useful feature for the learning model. I have decided to use the day of week, hour of day and day of month as features for the training.

Since, the data set only includes transactions for 6 months out of a year, the month of year feature cannot be generalized for the rest of the months in a year. Therefore, I didn't include the month of year feature for the learning model.

The original feature, TransactionDT has been dropped from the training features because the specific time wouldn't be a feature found common in the data for the model.

OS Data

An identity feature, id_30, is an identification of device's Operating System with version information. The version information of the OS would vary over the time, and new versions will appear in the future data. Particular version information may not match the version in the learning model that the feature may not be properly evaluated for the prediction.

In order to properly evaluate the feature for the learning model, the values are simplified to 'Windows', 'iOS', 'Mac' and 'Android'.

Browser (User Agent)

The feature, id_31 has browser information with version number and platform information. The reasoning for not to include the version number of the OS is also applicable for the version number and the platform information that are included in the user agent information feature.

The feature values have been generalized into browser names, 'Chrome', 'Firefox', 'Safari', 'Edge', 'IE, 'Samsung', 'Opera' and 'Others'.

Dropped Features

TransactionID and TransactionDate have been dropped from the data set for training and testing the models.

Implementation

The following steps are taken to implement the machine learning models for the Fraud Detection.

1. Prepare data sets for input to the learning algorithms by applying feature engineering mentioned in the section above.

2. Once the data has been prepared, it has been split into the training set (80%) and the test set (20%) to evaluate the learning model.

3. Decision Tree classifier model has been fit with the prepared data set to train the model, and once the model is trained, the trained model has been applied to the test data set to predict the results. Using the prediction results and the labeled test data, the Confusion Matrix, Precision, Recall and F1 Score have been calculated.

4. Using the same training data set to train the LightGBM model, and the test data set has been applied to the model to predict. With the results of prediction and the test set labels, the Confusion Matrix, Precision, Recall, and F1 Score have been calculated to compare with the benchmark model, and also the Receiver Operation Characteristics Area Under the Curve has been calculated.

Refinement

As the data set has about 96.5% of normal transaction data and only about 3.5% of the transactions have been marked as fraudulent, most of the normal transactions have been correctly predicted to be normal. The initial LGBM model I have tried, resulted with the Precision of 0.982, F1 score of 0.735 and the AUC value of 1.0 which looked pretty good. However, the model also resulted with quite a lot of false negatives, and scored 0.587 for Recall, meaning the model fails to detect significant number of fraudulent transactions.

In an effort to improve the metrics, I have tried a few different optimization methods.

1. Feature columns with over 95% of null values in the rows have been dropped from the training and test data sets and applied to the same model.
2. Applied tuning on the hyper-parameters to see if the test metrics resulted with better scores.

IV. Results

Model Evaluation and Validation

The dataset has a lot of null values for many of the features, and when the model is trained and tested with the features having null values in more than 95% of the rows being removed from the feature set, the test results scored very close to the model trained with all features.

The Confusion Matrix shows very well how the model performs. The Precision scores at `0.985`. That means transactions that are identified as fraudulent are very much correctly classified, and normal transactions are hardly incorrectly classified as fraud. However, the Recall scores at `0.638`, that is not too bad, but quite few fraud transactions are not correctly caught as fraud by this model. I have tried to improve the Recall score, but that was the best score I was able to achieve.

The competition objective metrics of AUC-ROC is calculated to be 1.0 for all LGBM models which shows the model can separate the true positives from the false positives, transactions that are incorrectly identified as fraud. Since the competition is evaluated by AUC, the model has achieved very good score.

Initial LightGBM model

```
params={'learning_rate': 0.01,
        'objective': 'binary',
        'metric': 'auc',
        'num_leaves': 256,
        'verbose': 1,
        'random_state': 42,
        'bagging_fraction': 1.0,
        'feature_fraction': 1.0
       }
```

Test Result

```
Confusion Matrix
true positive 2490, false positive: 46
false negative: 1752, true negative: 113820
Precision: 0.982, Recall: 0.587
F1 score: 0.735

Test AUC: 1.000
```

LightGBM model, features with less than 95% null values

```
params={'learning_rate': 0.03,
        'objective': 'binary',
        'metric': 'auc',
        'max_bin': 256,
        'num_leaves': 256,
        'min_data_in_leaf': 10,
        'verbose': 1,
        'random_state': 42,
        'bagging_fraction': 0.85,
        'bagging_freq': 10,
        'feature_fraction': 0.9,
        'max_depth': 128,
       }
```

Test Results

```
Confusion Matrix
true positive 2670, false positive: 41
false negative: 1572, true negative: 113825
Precision: 0.985, Recall: 0.629
F1 score: 0.768

Training AUC: 1.000
```

LightGBM model, optimized

```
params={'learning_rate': 0.03,
        'objective': 'binary',
        'metric': 'auc',
        'max_bin': 256,
        'num_leaves': 256,
        'min_data_in_leaf': 10,
        'verbose': 1,
        'random_state': 42,
        'bagging_fraction': 0.85,
        'bagging_freq': 10,
        'feature_fraction': 0.9,
        'max_depth': 128,
       }
```

Test Results

```
Confusion Matrix
true positive 2705, false positive: 42
false negative: 1537, true negative: 113824
Precision: 0.985, Recall: 0.638
F1 score: 0.774

Test AUC: 1.000
```

Justification

Due to the distribution of the normal vs. fraud transactions being so much skewed toward normal transactions, the accuracy metric on the results wouldn't be a good measure to determine how good the model is. Calculating Precision and Recall metrics would provide much better meaningful measurement on the model along with the AUC-ROC.

For comparison to the benchmark model, I have used the Confusion Matrix to evaluate how well the final model worked. The final LGBM model performed much better on Precision by 0.985 vs. 0.442 of the Decision Tree. The Decision Tree model resulted with a lot of false positive predictions.

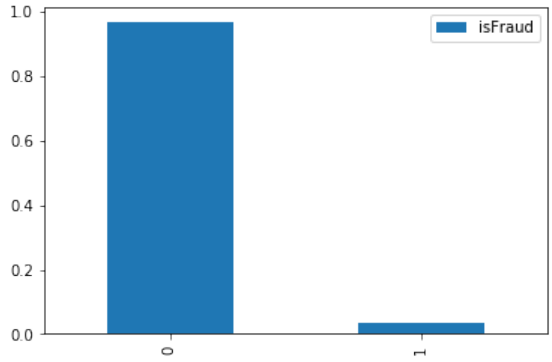
The Recall value on both of the models turned out fairly close 0.599 for Decision Tree model and 0.638 for LGBM. The LGBM model is a bit better on the Recall but both models detected

quite a lot of false negatives, meaning that many of the fraud transactions have not been properly identified as fraudulent. This is a bit of disappointment on detecting fraud to prevent bad transactions.

The F1 score of the Decision Tree model is 0.509, and the LGBM scored 0.774. This shows the LGBM model is significantly better model to detect the fraud than the benchmarked Decision Tree model.

Conclusion

Normal vs. Fraud Transactions



The data set provided for training has about 3.5% of the rows labeled as fraud, and the rest are normal. This means fewer than 4 out of 100 transactions are fraud.

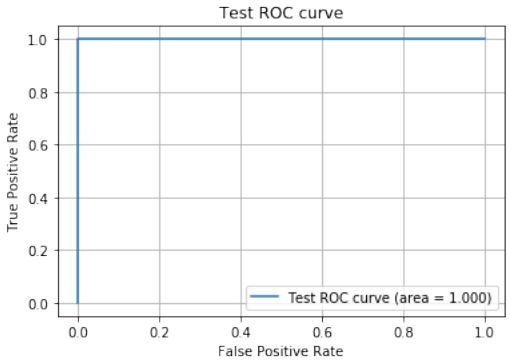
The test result of the selected LGBM model is evaluated using the Confusion Matrix.

Confusion Matrix on Test Data

	Fraud (True)	Normal (False)
Positive	2705	42
Negative	1537	113824

```
Precision: 0.985, Recall: 0.638
F1 score: 0.774
```

Receiver Operating Characteristic Curve of the optimized LGBM model




```
Test AUC: 1.000
```

As shown above, the test result of the optimized LGBM model scored high, 0.985, in Precision that when the model detects a transaction as fraud it is highly likely that it is indeed a fraudulent transaction. That is very good in precisely detecting fraud.

The Recall score of 0.638. That is a little less than 2/3 of the fraudulent transactions are detected correctly within all fraud transactions. In an ideal world, we'd like to detect every fraud transaction correctly, and have the system prevent all bad transactions. However, it is very difficult problem to solve.

The AUC is the metric the competition used for evaluating the models. The AUC is at 1.0. This means it has good measure of separability[5]. The model has achieved very good score for the competition.

Reflection

Working on the project that was presented as a competition on the Kaggle platform, it was very helpful for me to get started. Since the data sets have been provided, I didn't need to source the data for the project, and the competition objective was a very clear problem to solve.

As I downloaded the data sets and started examining them, I found that the data sets have over 430 columns. The number of columns was intimidating to begin exploration on the data, because I wasn't sure how I can explore the data column by column. Finding that most of columns don't have specific names, but they were labeled as generic names such as 'id_01', 'M_02', 'V_003' and etc. also made me wonder how to approach the exploration.

Somewhat overwhelmed to begin with the data exploration, I was fortunate that it was a Kaggle competition. To get an idea of where to begin, I sought a help from the discussion board and the notebooks on the platform. On the discussion board, I found the description on the data set[13], which gave me some understanding of what kinds of data in each column. For the most fundamental exploration of the data, I began with counting values in each column to see what kinds of values there are and what the distributions look like. It turned out many columns have over 75% of rows have null values.

To get better understanding of what the features are and how to use the features for training the learning model, I looked into a few notebooks on the Kaggle, and learned what the other people were doing with the data exploration, and feature engineering, and applied the techniques on the data set where I see fit. I learned a lot by researching on the data exploration and feature engineering through this. The notebooks showed me that visualization on the data helps reveals insights on the data very clearly. This led me to realize that I need to improve my data visualization skills to become proficient in it.

Once feature engineering techniques have been applied to the data set, I executed training and test on the data set using selected learning algorithms.

Improvement

As mentioned above, if I were more proficient in data visualization, I would be more efficient in data exploration and get even better insight on the data set. This will lead me to apply feature engineering that might result in better predictions.

Knowing what algorithms are available and applicable to this kind of problems and applying the algorithms to examine which algorithm results in better performance would lead to a better solution. Also, learning the details of the hyper-parameters of each learning algorithm would let me fine tune the hyper-parameters to let me build even better model.

Neural net based deep learning algorithms might be good candidates for consideration.

References

1. https://www.consumeraffairs.com/finance/identity-theft-statistics.html
2. https://www.kaggle.com/c/ieee-fraud-detection/data
3. https://www.kaggle.com/c/ieee-fraud-detection/overview
4. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
5. https://en.wikipedia.org/wiki/F1_score
6. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
7. https://xgboost.readthedocs.io/en/latest/
8. https://lightgbm.readthedocs.io/en/latest/
9. https://medium.com/kaggle-nyc/gradient-boosting-decision-trees-xgboost-vs-lightgbm-and-catboost-72df6979e0bb
10. https://lightgbm.readthedocs.io/en/latest/Parameters.html
11. https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html
12. https://www.kaggle.com/gemartin/load-data-reduce-memory-usage
13. https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203#latest-643955