Capstone Project
Machine Learning Engineering
Nanodegree

Christopher Kim
October 16, 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 applied 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.

To build a machine learning model, I will begin with examining and exploring the datasets that will be the input to the learning algorithms, to get a sense of what kinds of information is included in the dataset. Then I will apply transformations on the dataset to be used as features for training and testing the machine learning model. Finally, I will remove any columns that would not provide good information as features for the learning model.

I will separate the column that has the label values into a new label dataset and drop the column from the dataset for the training input. Once input and the label datasets are separated, I will split the datasets into a training set, a validation set and a test set, and I will use only the training set for training models, and validation set for only evaluating the performance of the learning model and tuning the hyper-parameters. The test set will be held out for the final model to test the model with the data the model has never seen.

Once the datasets have been prepared, I will process the data through a benchmark learning model to get the baseline metrics on the model for comparison with the final model. Using the same dataset that was used to train the benchmark model, I will apply the final learning algorithm to get the metrics to compare with the benchmark model.

With the metrics gathered by training and validating the learning model, I will evaluate how good the model is. The initial learning model will probably not produce the best results in prediction on the validation dataset. I will apply tuning on the hyper-parameters and use the same training set to train and validate with the same validation dataset to find a model that produces the best results on the validation dataset for the selected metrics.

The learning model with the hyper-parameter that produces the best metric measurements will be the solution for the project. I will use the test data set that was held out, to get the final performance metrics on the model.

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. 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.

Confusion Matrix

	Positive	Negative
True	True Positive	True Negative
False	False Positive	False Negative

The learning model predicts each transaction to be fraud or not, and the predicted result is compared to the truth which the original data has been labeled. The matrix values are identified as below:

True Positive: count of fraud transactions correctly predicted to be fraud
True Negative: count of normal (not fraud) transactions correctly predicted to be normal
False Positive: count of normal transactions incorrectly predicted to be fraud

False Negative: count of fraud transactions incorrectly predicted to be normal

Based on the information gathered in the Confusion Matrix, Precision, Recall and F1 score as below:

Precision = True Positive / (True Positive + False Positive)
Recall = True Positive / (True Positive + False Negative)

Using the two measures, Precision and Recall F1 is calculated:

F1 = 2 * Recall * Precision / (Precision + Recall)

Precision score tells us about when it predicts as a fraud, how often it is correct. Recall score gives us an idea about when it's actually a fraud, how often the model correctly predicts the transaction as fraudulent. 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.

The competition submissions are evaluated on AUC(area under curve) on the ROC(receiver operation characteristics)[6] curve. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at distinguishing between transactions as fraud or normal. 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.

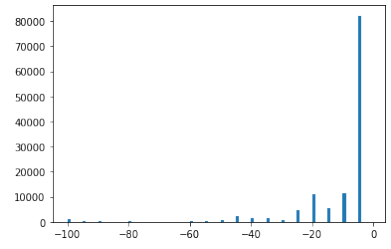
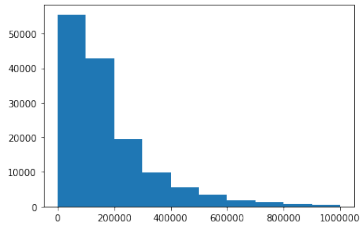
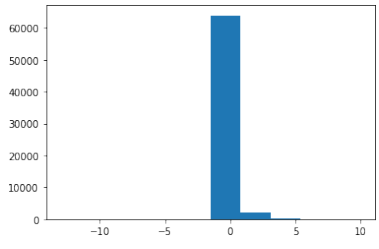
```
Id_07
 NaN      0.991271
 0.0      0.000693
 16.0     0.000415
...
```

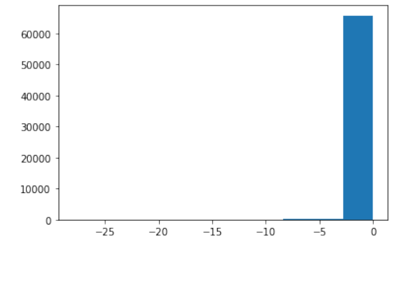
Numerical Value Distributions in the Identity data

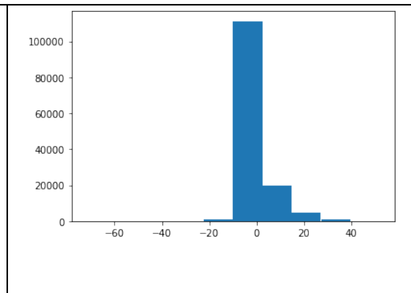
```
train_identity[['id_01', 'id_02', 'id_03', 'id_04', 'id_05', 'id_06', 'id_09', 'id_10', 'id_11']].describe()
```

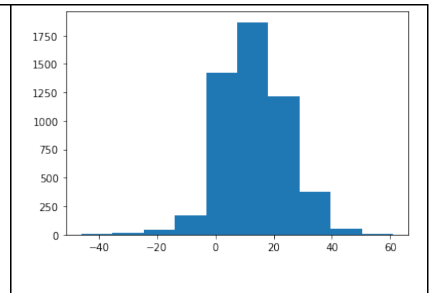
	id_01	id_02	id_03	id_04	id_05	id_06	id_09	id_10	id_11
count	144233.000000	140872.000000	66324.000000	66324.000000	136865.000000	136865.000000	74926.000000	74926.000000	140978.000000
mean	-10.170502	174716.584708	0.060189	-0.058938	1.615585	-6.698710	0.091023	-0.301124	99.745325
std	14.347949	159651.816856	0.598231	0.701015	5.249856	16.491104	0.983842	2.789446	1.127602
min	-100.000000	1.000000	-13.000000	-28.000000	-72.000000	-100.000000	-36.000000	-100.000000	90.000000
25%	-10.000000	67992.000000	0.000000	0.000000	0.000000	-6.000000	0.000000	0.000000	100.000000
50%	-5.000000	125800.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.000000
75%	-5.000000	228749.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	100.000000
max	0.000000	999595.000000	10.000000	0.000000	52.000000	0.000000	25.000000	0.000000	100.000000

The columns id_01, id_04, id_06 and id_10 have negative values. The column id_02 has values range from 1 to 1,000,000.

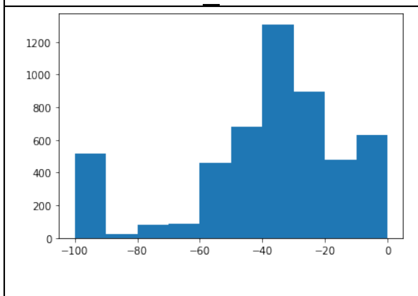
id_01	id_02	id_03
		
id_04	id_05	id_06



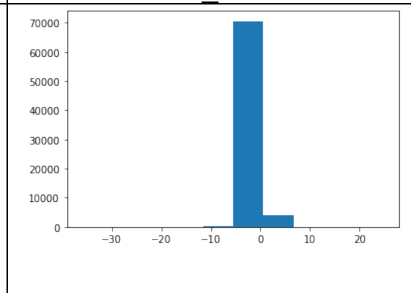




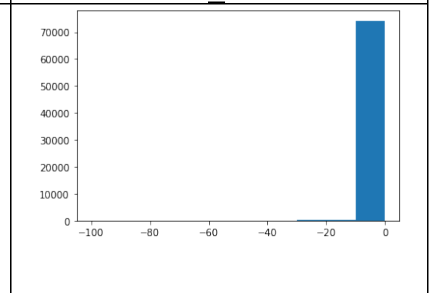
Id_08



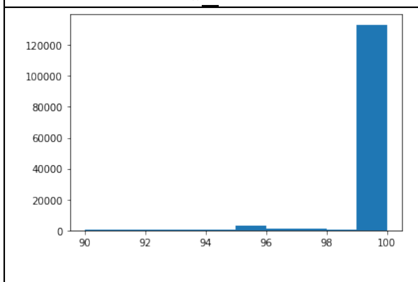
Id_09



Id_10



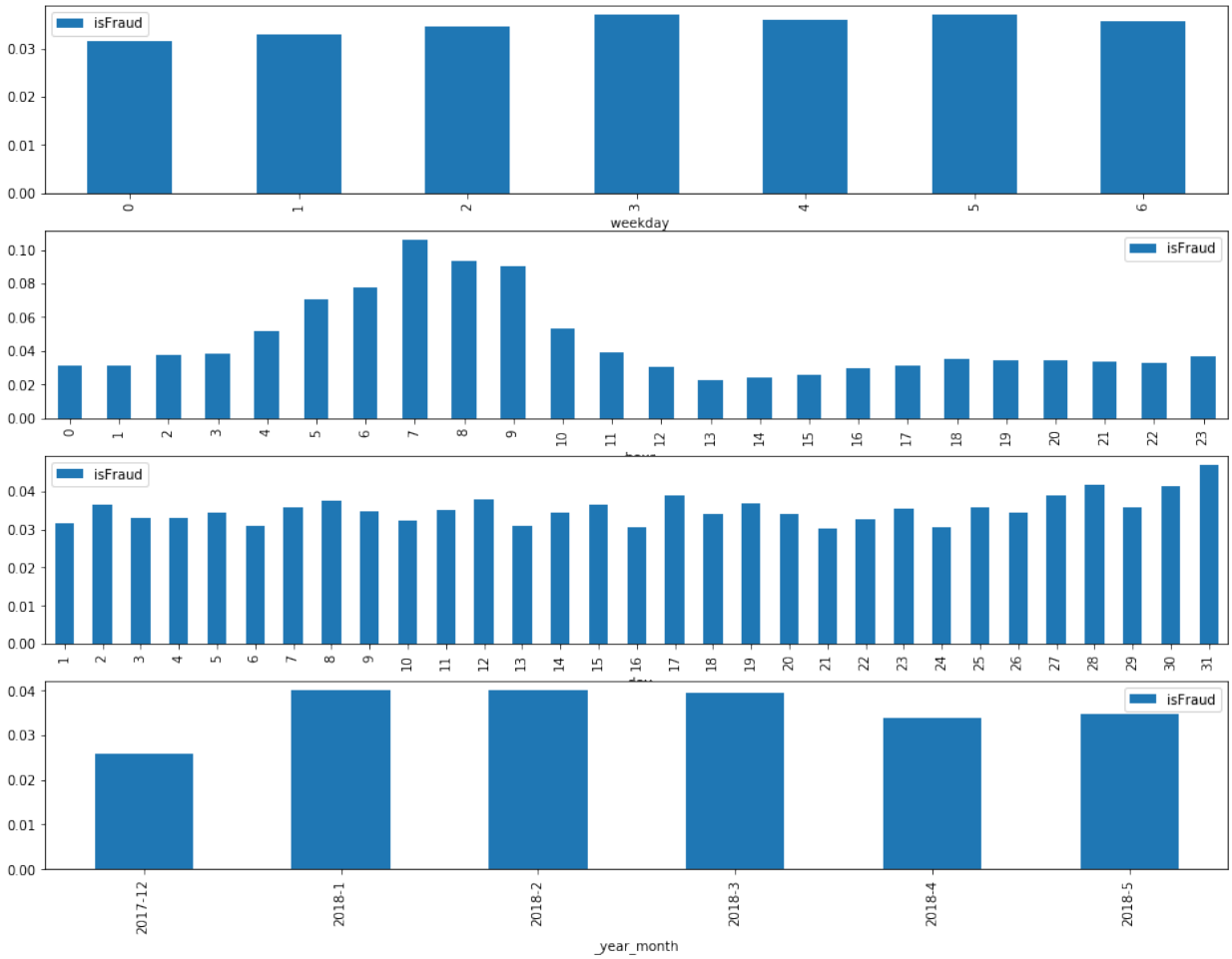
id_11



The values in the columns id_03, id_04, id_05, id_09 and id_10 have the distribution skewed to 0, and id_11 has distribution skewed to 100.

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[7] is a simple tree-based decision-making algorithm to use for this problem. A tree can be *"learned"* by splitting the source set into subsets based on an attribute value test[8]. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

For this problem, the feature value that will separate the transaction record between fraud or not will be split. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. As the dataset has over 400 hundred features, it is very suitable to use the Decision Tree for the learning algorithm for the fraud detection. In general decision tree classifier has good accuracy and can be a good model to benchmark.

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[9] and LightGBM[10].

I researched online to find the comparison between the algorithms[11]. 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.

LightGBM is a Gradient Boosting[12] algorithm. The Gradient Boosting Classifier is an algorithm that iterates many decision-making stages to improve the predictions on whether a transaction is a fraud or normal with certain features. For example, it guesses that a transaction that happened on a certain day of week would be fraud or not, then compare the predictions with the true answers, and get the differences.

Next, it will try to improve the prediction by using another feature like card type would be fraud or not. Then combine the decision conditions together to improve the prediction, and repeat these steps using other features many times until predetermined number of times like 100 times, or performance on the correct prediction doesn't improve.

At the end, the combination of decision-making criteria becomes the final condition that will answer for the new data.

LightGBM is designed to be distributed and efficient with the following advantages[10]:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

All of the advantages listed above are good reasons to choose LightGBM for this project.

For LGBM algorithm the following parameters[14] can be tuned[15] 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 validation. 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 2394, false positive: 2235
false negative: 1848, true negative: 111631
Precision: 0.517, Recall: 0.564
F1 score: 0.540

CPU times: user 1min 30s, sys: 837 ms, total: 1min 31s
Wall time: 1min 30s
```

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[14]. 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. The solution has been implemented using Python 3.7 on a Jupyter Notebook.

2. Python packages Pandas, Numpy, Matplotlib and Scikit-Learns are used to implement the solution.

3. The CSV formatted data files, identity data and transaction data, are imported using Pandas read_csv function.

4. Using a function that converts data types to reduce memory usage is applied to the Pandas data frame to reduce memory usage by about 50%.

5. The two datasets, identity data and transaction data have been joined on TransactionID column.

6. Removed the original identity and the transaction data and called the garbage collection routine to save memory.

7. Prepared data sets for input to the learning algorithms by applying feature engineering mentioned in the section above. (Engineered transaction date, simplified OS and user agent values)

8. Applied the Label Encoder on the category columns.

9. The dataset has been split into the training set, validation set and test set to evaluate the learning model.

10. 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.

11. Using the same training data set to train the LightGBM model, and the validation data set has been applied to the model to predict. With the results of prediction and the validation 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.

12. Evaluated the result of the model and applied the tuning on the hyper-parameters to improve the performance of the model.

13. Once the model has been finalized with the optimal parameters, I tested the model with the test dataset that has been held out to measure the practical performance of the model.

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 0.971, F1 score of 0.701 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 0.549 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 99% 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.

   For better accuracy the parameter

   ```
   'num_leaves': 256 (default=31),
   'min_data_in_leaf':10 (default=20),
   'num_leaves' : 256 (default=31),
   ```

   To handle overfitting

   ```
   'max_depth': 128 (default=-1, no limit),
   'bagging_fraction': 0.85 (default=1.0),
   'bagging_freq': 10(default=0, disabled),
   'feature_fraction': 0.9(default=1.0)
   ```

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 99% 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 `0.983`. 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 `0.596`, 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
       }
```

Validation Result

```
Confusion Matrix
true positive 1863, false positive: 56
false negative: 1530, true negative: 91038
Precision: 0.971, Recall: 0.549
F1 score: 0.701

Test 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,
       }
```

Validation Results

```
Confusion Matrix
true positive 2022, false positive: 36
false negative: 1371, true negative: 91058
Precision: 0.983, Recall: 0.596
F1 score: 0.742

Test AUC: 1.000
```

Test Results

```
Confusion Matrix
true positive 2521, false positive: 39
false negative: 1721, true negative: 113827
```

```
Precision: 0.985, Recall: 0.594
F1 score: 0.741

Test AUC: 1.000
```

LightGBM model, features with less than 99% 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,
       }
```

Validation Results

```
Confusion Matrix
true positive 2050, false positive: 40
false negative: 1343, true negative: 91054
Precision: 0.981, Recall: 0.604
F1 score: 0.748

Training AUC: 1.000
```

Test Results

```
Confusion Matrix
true positive 2580, false positive: 51
false negative: 1662, true negative: 113815
Precision: 0.981, Recall: 0.608
F1 score: 0.751

Training 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. 0.517 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 0.564 for Decision Tree model and 0.596 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 0.540, and the LGBM scored 0.742. This shows the LGBM model is significantly better model to detect the fraud than the benchmarked Decision Tree model.

The LGBM model has been trained and validated using separate train and validation datasets, and the parameters have been tuned to get better prediction results. Once the model has been tuned, I have used the test dataset that has been held out to get the final evaluation of the model.

I also tested the model with columns with null values in more than 99% of the rows removed, and the model also resulted very close to the other model. The model resulted even better score for Recall and F1 with the columns removed.

Here's the comparison of the validation and test metrics.

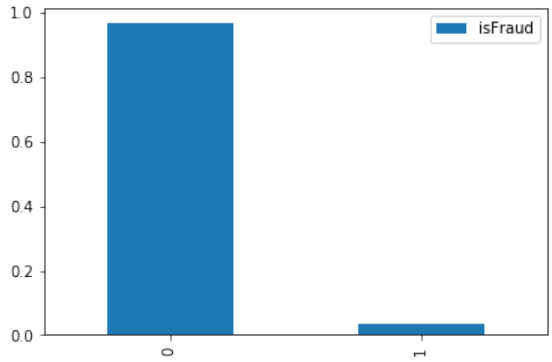
Validation	Test	Test (feature with 99% null removed)

`Precision: 0.981,`
`Recall: 0.604`
`F1 score: 0.748`
	`Precision: 0.983,`
	`Recall: 0.596`
	`F1 score: 0.742`
		`Precision: 0.981,`
		`Recall: 0.608`
		`F1 score: 0.751`

This shows the model performs pretty robust on the unseen data.

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.

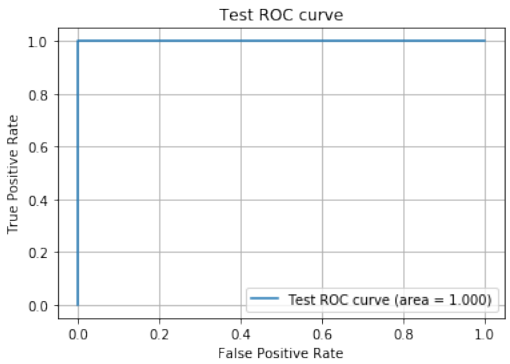
The test result of the selected LGBM model is evaluated using the Confusion Matrix.

Confusion Matrix on Test Data

	Fraud (True)	Normal (False)
Positiv	2521	39
Negative	1721	113827

```
Precision: 0.985, Recall: 0.594
F1 score: 0.741
```

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 0.594. 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[16], 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://en.wikipedia.org/wiki/Decision_tree
8. https://www.geeksforgeeks.org/decision-tree/
9. https://xgboost.readthedocs.io/en/latest/
10. https://lightgbm.readthedocs.io/en/latest/
11. https://medium.com/kaggle-nyc/gradient-boosting-decision-trees-xgboost-vs-lightgbm-and-catboost-72df6979e0bb
12. https://en.wikipedia.org/wiki/Gradient_boosting
13. https://lightgbm.readthedocs.io/en/latest/Parameters.html
14. https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html
15. https://www.kaggle.com/gemartin/load-data-reduce-memory-usage
16. https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203#latest-643955