Machine Learning Engineer Nanodegree

Capstone Project

Nitin Mahajan
June 24th , 2018

I. Definition

Project Overview

Credit Card fraud losses are a huge problem. The Nilson Report, a publication covering global payment systems, reported recently that global card fraud losses equaled $22.8 billion in 2016, an increase of 4.4 percent over 2015. That amount does not include costs incurred by retailers, card issuers, and acquirers for their operations and chargeback management.

By 2021, card fraud worldwide is expected to reach a total of $32.96 billion. It is important that card companies can identify fraudulent card transactions so that customers are not charged for items that they did not purchase. Better tactics and technology to combat criminals attempting such transactions will define fraud fighting for merchants in coming years.

Fraud detection is one of the most important applications for machine learning in finance. Institutions are now using machine learning to help detect frauds. Supervised learning is an area of machine learning where based on historical data, which is labelled the machine learns the pattern in the data. Based on the learning, the machine is then able to make predictions for the labels of unseen & unlabelled data.

This project will detail the process of building a supervised learning machine learning model to classify fraudulent and non-fraudulent transactions. The data has for the project has anonymized credit card transactions labelled as fraudulent or genuine. The project data has been obtained from Kaggle.

Problem Statement

The goal of this project is to classify the transactions into fraudulent or genuine transactions. Our main focus would be to identify the fraudulent transactions as these are the transactions we would want to separate from the genuine transactions. Below steps would be involved to solve this binary classification problem-

1. Download the data from Kaggle
2. Data exploration would be done to understand the data better and do any further processing as required
3. Train multiple classifiers i.e. –
    a. Logistic Regression
    b. Adaboost

    c. Ridge Classifier
    d. Light Gradient Boost
4. Select the best model by comparing the Area under precision recall curve(AUPRC) and further try improving the results.

Metrics

The datasets contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

Given the imbalance ratio, we would be using Area Under the Precision-Recall Curve (AUPRC) to measure accuracy. The closer to 1 the AUPRC is, the better the model is. A model with AUPRC score of 1 implies a perfect classifier.

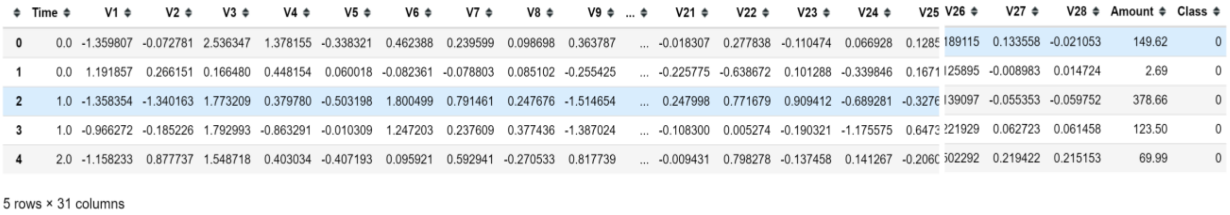
Precision is the proportion of test cases predicted to be fraud that were indeed fraudulent (i.e. the true positive predictions), while recall or sensitivity is the proportion of fraud cases that were identified as fraud. This precision-recall curve(AUPRC) tells us the relationship between correct fraud predictions and the proportion of fraud cases that were detected (e.g. if all or most fraud cases were identified, we also have many non-fraud cases predicted as fraud and vice versa).

II. Analysis

Data Exploration

The dataset for the project is a set of credit card transactions over a period of 2 days with each transaction labelled as ‘fraudulent’ or ‘genuine’. To maintain confidentiality this data is anonymized & the original features were not provided.

The data has only numerical input variables which are the result of a PCA transformation. Again, due to confidentiality issues, there is no background information available for this data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Below is a snapshot of first 5 rows of data –



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.1285	189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.1671	125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.3276	139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.6473	221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.2060	502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The time between first and last transaction is 172,792 seconds or 48 hours. So the data provided has the credit card transactions which occurred over a period of 2 days.

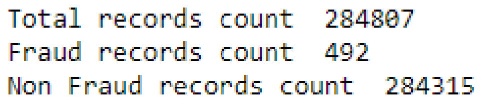
The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. The average transaction amount for fraudulent & genuine transactions is 122.21 and 88.29 respectively.

Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Exploratory Visualization

Distribution of Class

Figure-1 below, shows the distribution of the class i.e. no. of fraudulent transactions vs others. It can be clearly seen that the class i.e. 1 or the fraudulent transactions is the minority class and the dataset is highly unbalanced.



```
Total records count  284807
Fraud records count  492
Non Fraud records count  284315
```

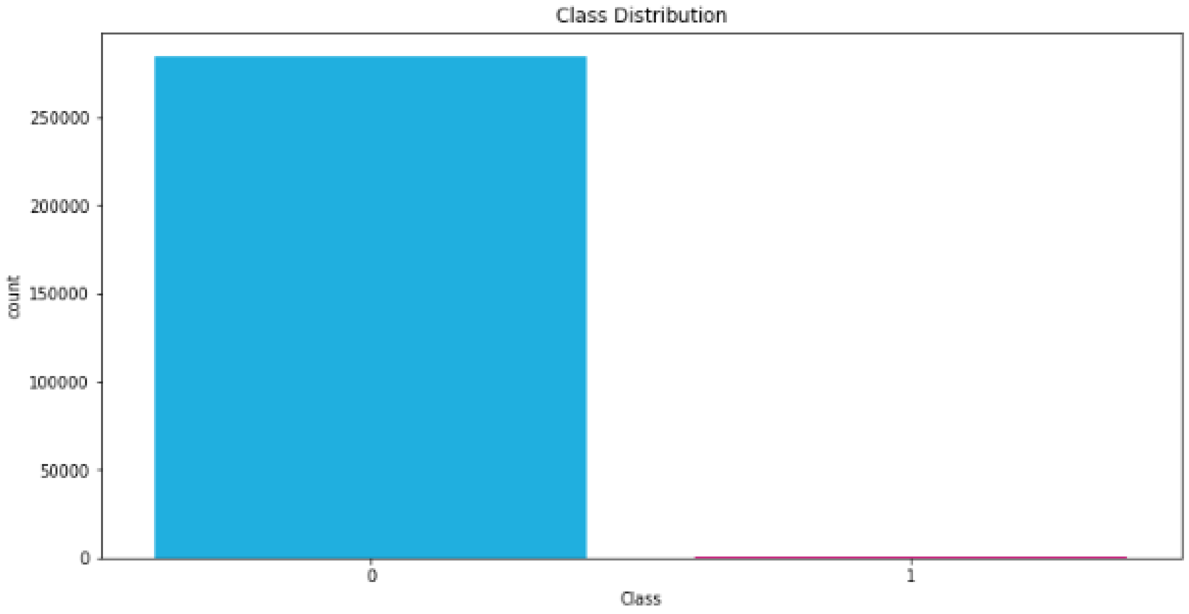


Figure-1

Explore Amount

Figure-2 below shows the spread of transaction amounts for fraudulent and normal transactions. Most of the Fraudulent transactions are low value as compared to non-fraudulent or normal transactions. The maximum amount for a fraudulent transaction is 2,125.

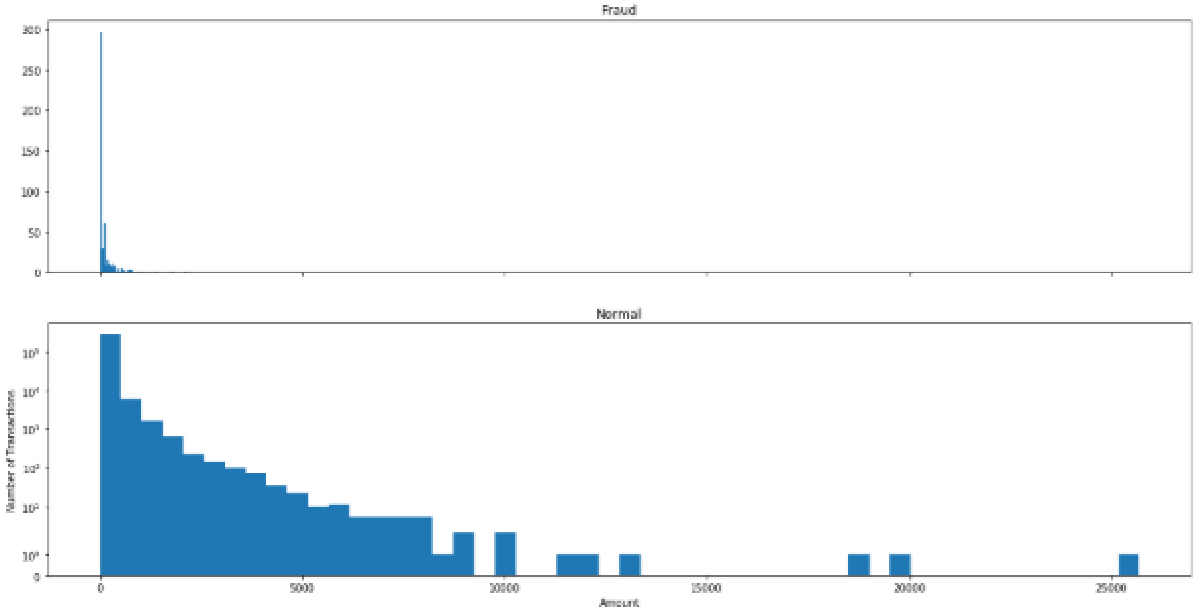


Figure-2

Explore Time vs Amount

Figure-3 below shows the spread of amount values for fraudulent & normal transactions over a period of 48 hours. The way the normal transactions occur over a period of 48 hour looks cyclical but not for Fraud transactions. There is no other significant reading from the below plot.

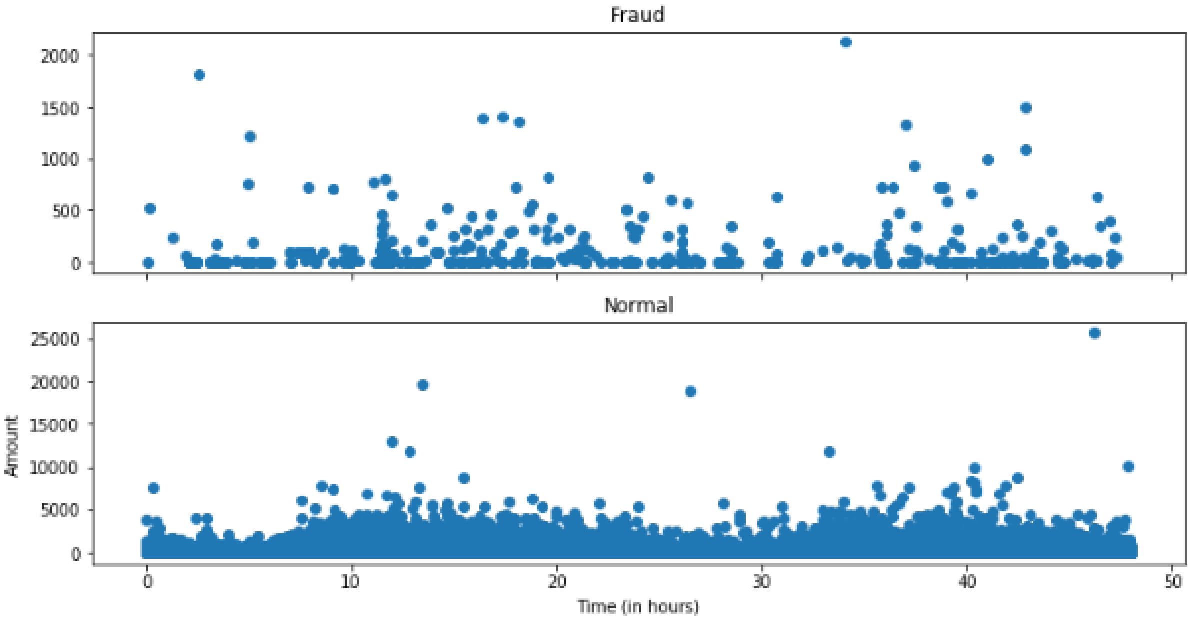


Figure-3

Algorithms and Techniques

This is a binary classification problem with a highly unbalanced dataset. Based on the nature of problem 4 machine learning algorithms namely, Logistic regression, Adaboost, Ridge Classifier and Light gradient boosting algorithms were tested.

Logistic Regression

Logistic regression is named for the function used at the core of the method, the logistic function also called as the sigmoid function. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

The coefficients or the weights of the logistic regression algorithm must be estimated from the training data. This is done using Maximum-likelihood estimation (a common learning algorithm used by a variety of machine learning algorithms).

The best coefficients would result in a model that would predict a value very close to 1 (e.g. non-fraud) for the default class and a value very close to 0 (e.g. fraud) for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients (Beta values) that minimize the error in the probabilities predicted by the model to those in the data (e.g. probability of 1 if the data is the primary class).

We should use Logistic regression as our problem is a binary classification problem where we think the features are somewhat related to the classes. Logistic Regression is also one of the simple classification techniques and would serve as a good baseline model to compare with more sophisticated models like Light Gradient boost, Adaboost.

Adaboost (Adaptive Boosting)

Boosting is a general ensemble method that creates a strong classifier from a number of weak classifiers. AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem.

AdaBoost is best used to boost the performance of decision trees on binary classification problems. We have a problem of binary classification and this model, which is much more sophisticated than the logistic regression model would can give a good metrics to compare with the baseline model.

This is binary classification is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly, or a maximum number of models are added.

In brief Adaboost is an ensemble machine learning technique which uses multiple weak learners to make one strong learner.

Ridge Classifier

Ridge regression is a method that seeks to reduce the mean square error, by adding some bias and, at the same time, reducing the variance. It learns weights & bias, adds a penalty for large variations in weights parameters. This penalizing of the parameter is called as regularization which prevents overfitting by restricting the model, typically to reduce the complexity.

The influence of the regularization term is controlled by the α parameter. Higher α means more regularization and simpler model. We would be using the default alpha value for our model. This is a regression which would be used as a classifier by using the threshold or rounding off.

This model should be used for this project as it performs as this model is very simple & would help generalize better by avoiding overfitting and we would be comparing the performance against other complex models.

Light Gradient Boosting or LIGHT GBM

LightGBM is a fast, distributed as well as high-performance gradient boosting framework that makes the use of a learning algorithm that is tree-based, and is used for ranking, classification as well as many other machine learning tasks. While other algorithms grow trees horizontally, Light GBM grows tree vertically meaning that Light GBM grows tree leaf-wise while other algorithms grow level-wise. It, in order to grow, will choose the leaf that has a max delta loss. When growing the same leaf, Leaf-wise algorithm can reduce more loss when compared to a level-wise algorithm.

Light GBM is relatively a new algorithm and can handle large size of data and takes lower memory to run. Light GBM has light in it' name to signify high speed of the algorithm. Based on above should be worthwhile to try this new algorithm.

Oversampling Techniques

We have a binary classification task in hand where one class dominates the other. Due to this our model could simply classify all transactions as non-fraudulent and still be 99% correct. There are different ways in which we come overcome this problem like –

- Tweak class weights to apply more weightage to
- Synthesis of new minority class instances
- Oversampling of minority class
- Undersampling of majority class

In this project we have applied class weights on some of the models and observed improvements and we used SMOTE for synthesizing new minority class instances.

SMOTE- Synthetic Minority Over-sampling Technique

SMOTE synthesises new minority instances between existing (real) minority instances.

It acts from perspective of the existing minority class and creates new instances very near/similar to them. The SMOTE samples are linear combinations of two similar samples from the minority class. It does not change the expected value of the (SMOTE-augmented) minority class and it decreases its variability. For SMOTE the features and classes are passed and it returns oversampled data which is balanced and can be further used for classification.

Benchmark

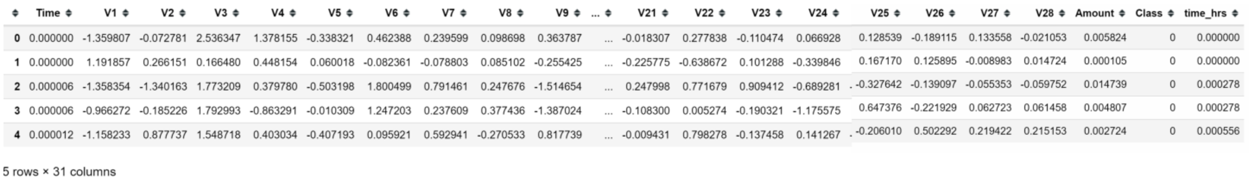
The biggest challenge with this project is highly unbalanced data and accuracy score or the confusion matrix is not helpful metrics for such data. For this imbalanced dataset the metrics, area under the precision recall curve(AUPRC) would be used.

For benchmark, a simple classification model like Logistic regression model was run as the benchmark model and it produced an AUPRC of 0.52. This would be compared against different models with a goal of making sure that any models producing results are worth considering only if the AUPRC score is greater than 0.52.

III. Methodology

Data Preprocessing

The **first step** taken in pre-processing was Normalization. All fields except Amount and Time have been transformed using PCA. Amount and Time were transformed/scaled and below is the top 5 rows of data after scaling these fields.



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class	time_hrs
0	0.000000	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	0.005824	0	0.000000
1	0.000000	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	0.000105	0	0.000000
2	0.000006	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	0.014739	0	0.000278
3	0.000006	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	0.004807	0	0.000278
4	0.000012	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	0.002724	0	0.000556

5 rows × 31 columns

Since the dataset was highly imbalanced, the **next step** was to balance the dataset SMOTE algorithm was applied. SMOTE or synthetic minority oversampling technique is an approach which works from the perspective of existing minority instances and synthesises new instances at some distance from them towards one of their neighbours.

The training data was resampled using SMOTE and test data was kept untouched.

Class Distribution	***Fraudulent***	***Non-Fraudulent***
Before Oversampling	492 (0.173%)	284807 (99.837%)
After Oversampling(SMOTE)	199013 (50%)	199013 (50%)

Table-1

Implementation

Implement multiple models (Initial run- unbalanced dataset)

Below are the steps that were taken for the implementation

1. Test 4 different models on the data(unbalanced) split into training and testing set.
    a. Logistic Regression
    b. Adaboost
    c. Ridge Classifier
    d. Light Gradient Boost(LIGHT GBM)

2. Apply class weights to minority class i.e. fraudulent transactions for the 2 classifiers only (i.e. Logistic Regression and Ridge classifier) and compare the performance (details in refinement section)
3. Next the dataset was oversampled using SMOTE and
4. This dataset was further divided into 2 sets (using K Fold) to test the 4 models.
5. Apply class weights to minority class i.e. fraudulent transactions for 2 classifiers (i.e. Logistic Regression and Ridge classifier) and compare the performance

   As mentioned above the implementation was done by training 4 different algorithms and comparing their performance using the AUPRC metric. Below table compares the performance of the 4 algorithms. This was done by dividing the data into training and testing sets where the data kept for testing was 30% of overall data.

Note: No oversampling done

Model	***AUPRC***	***Precision***	***Recall***	***Train time***	***Test Time***
Adaboost	0.7302	0.7578	0.6879	62.89	0.72
Ridge Classifier	0.7508	0.7778	0.3971	0.33	.01
Logistic Regression	0.7706	0.8521	0.6950	4.76	.01
Light Gradient Boosting	0.8830	0.9426	0.8156	5.76	.4208

Table-2

It was very evident from initial results that the 'Light Gradient Boosting' or LIGHT GBM has the best metrics for the AUPRC score of 0.8830. The model gives the best precision of .9426 at 0.8156 recall. All the 3 metrics are by far the best for the model when compared to others.

Overcome the imbalance – Apply oversampling

We have already tried penalizing the dominating class by using class weights. We can now further try and improve results by using resampling the training data using oversampling or under-sampling techniques. I'm focussing on resampling by oversampling the minority class by using a technique called SMOTE (Synthetic Minority

Over-Sampling Technique) as explained in the section above.In order to make sure that the models are robust and results are not by chance the test data was divided into 2 folds. Below are the results for various models after oversampling the training data

Model	***AUPRC***	***Precision***	***Recall***	***Train time***	***Test Time***
Adaboost	0.9962 0.9960	0.9743 0.9736	0.9579 0.9559	40.82 40.73	1.31 1.32
Ridge Classifier	0.9767 0.9757	0.9833 0.98.25	0.8360 0.8356	0.21 0.19	.01 .01
Logistic Regression	0.9889 0.9886	0.9697 0.9671	0.9100 0.9096	3.19 4.21	.01 .01
Light Gradient Boosting	**0.99998** **0.99992**	**0.99910** **0.99908**	**1.0** **1.0**	**7.54** **7.78**	**1.10** **1.15**

Table-3

Refinement

Applying class weights (before oversampling)

Class weights were applied on 'Logistic regression' and 'Ridge Classifier' to further improve the performance by putting more weight or emphasis on the class 1 or the frauds. Below table has updated metrics for these 2 classifiers only.

Both the models had a better overall AUPRC score. For Ridge classifier there was an improvement in all the 3 metrics i.e. precsion, recall and AUPRC. There was an improvement in recall but a drop in precision for the Logistic regression. Even then the performance of 'Light Gradient Boosting' was far better than any of the algorithms.

Model	***AUPRC***	***Precision***	***Recall***	***Train time***	***Test Time***
Adaboost	0.7302	0.7578	0.6879	62.89	0.72
Ridge Classifier	0.7528	0.8095	0.4822	0.33	.01
Logistic Regression	0.7731	0.7667	0.8156	4.79	.01
Light Gradient Boosting	**0.8830**	**0.9426**	**0.8156**	**5.76**	**.4208**

Table-4

Applying class weights (with oversampled data, from SMOTE)

With the dataset balanced, class weights were applied to the Logistic regression and the Ridge classifier. The overall AUPRC improved for both the models.

Model	***AUPRC***	***Precision***	***Recall***	***Train time***	***Test Time***
Adaboost	0.9962 0.9960	0.9743 0.9736	0.9579 0.9559	40.82 40.73	1.31 1.32
Ridge Classifier	0.9687 0.9675	0.8956 0.8930	0.9083 0.9081	0.24 0.20	.01 .01

Logistic Regression	0.9893 0.9889	0.9394 0.9383	0.9351 0.9355	2.87 3.16	.01 .01
Light Gradient Boosting	**0.99998** **0.99992**	**0.99910** **0.99908**	**1.0** **1.0**	**7.54** **7.78**	**1.10** **1.15**

Table-5

IV. Results

Model Evaluation and Validation

Comparing the 4 algorithms, **Light Gradient Boosting(LIGHT GBM)** algorithm has outperformed all the other models namely -Logistic Regression, Adaboost and Ridge Classifier for all the metrics namely precision, recall and the AUPRC.

In order to make sure that the model performance did not detoriate with the change in training/testing data, multiple(2) samples of training/testing data were crated and the models were tested. The comparison of the different tests validate the robustness of our model.

Adaboost model has a lower but a comparable AUPRC score of 99.96% but it takes much longer to train/test compared to the LIGHT GBM. LIGHT GBM produces better results in 1/20th of the time when compared to Adaboost. So, working on Adaboost to further refine the results is not reasonable.

Justification

The best model i.e. LIGHT GBM model on the oversampled data has a precision of 99.99% at 100% recall with AUPRC score of 99.99% which is very good and a significant improvement when compared to the benchmark model which has an AUPRC score of of 52%(77% precision at 63% recall).

V. Conclusion

Free-Form Visualization

Below figure illustrates the improvement in overall performance as compared to the benchmark model. The figure on left shows the area under the precision recall curve for the benchmark model and the one on right is for the selected LIGHT GBM model.

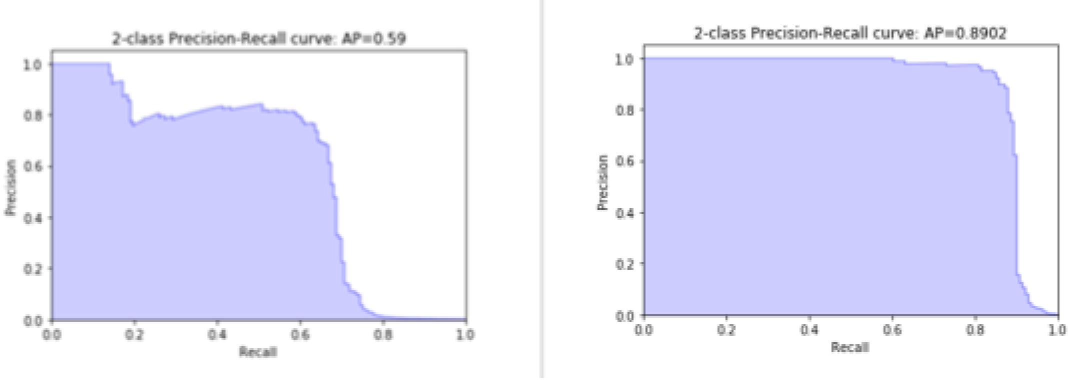


Figure-4

LIGHT GBM Model predicted 85,250 non-fraudulent transactions correctly and incorrectly predicted 52 fraudulent as non fraud. 15 of non Fraudulent transactions were wrongfully identified as fraud transactions and 126 of the fraudulent transactions were correctly identified.

The model gave a precision of 71% while recalling 89% of the fraudulent transactions on the test data.

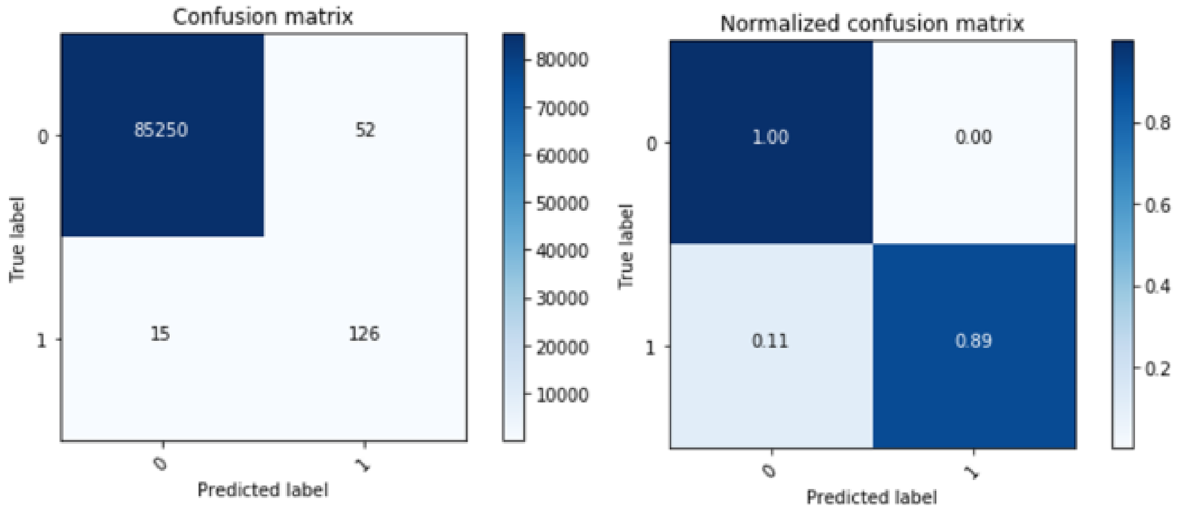


Figure-5

Reflection

While looking for a good use case for the capstone project I found this project as one of the challenging projects and an important problem to solve in the real world. Working in the banking industry also motivated me to select this project.

The data was easily available from Kaggle and nearly all the data was already transformed using PCA which reduced the work or analysis that could be done on various features. Time and Amount were the only two fields which were later transformed/scaled later.

I started working on the project by doing some exploration to find a relation between time, amount and different classes. It was observed that there was some cyclic behaviour of non-fraud transactions during the period of 48 hours but this cyclic behaviour was not visible for Fraud transactions. It was also observed that generally Fraud transactions were of lower values when compared to the genuine transactions.

Then I ran multiple models and compared their performance with each other and the benchmark. Overall Adaboost and LIGHT GBM models gave better results when compared to Ridge classifier and logistic regression. LIGHT GBM model outperformed Adaboost in all aspects i.e. precision, recall and AUPRC score. LIGHT GBM is much faster to train and gives results much sooner than the Adaboost.

Applying class weights to logistic regression did give some boost in performance but then also the performance was not better than Adaboost or LIGHT GBM.

In my view the major challenge for this project was the imbalanced classes which was very well taken care by the oversampling technique called SMOTE. Initially, I did the oversampling incorrectly by applying SMOTE to test data as well and my results were overfitting. I only saw the real performance of the model once I kept the test data separate (i.e. without applying oversampling).

Post oversampling the same algorithms performed much better. I also tried applying class weights to some of the models post-oversampling (although I thought it would not make a positive impact) and there was some improvement observed in performance of Logistic regression but again it was not a significant improvement.

Improvement

- A definite and further improvement would be possible if we have better description of the fields in order to do more meaningful analysis of the data.
- Feature selection may be done and focus on the most relevant features may help improving the results
- More data and more examples of frauds would help in improving the results and we can try and apply deep learning techniques to further try and improve results

References

https://medium.com/data-science-group-iitr/logistic-regression-simplified-9b4efe801389

https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/

https://www.coursera.org/lecture/python-machine-learning/linear-regression-ridge-lasso-and-polynomial-regression-M7yUQ?authMode=signup

https://www.techleer.com/articles/489-lightgbm-a-light-gradient-boosting-machine/

https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-106

http://rikunert.com/SMOTE_explained