CREDIT CARD FRAUD DETECTION-CAPSTONE PROJECT REPORT

Abstract

This report describes Capstone project as part of Udacity Machine Learning Engineer Nanodegree course for building machine learning system to detect Credit Card Fraud

1 Contents

1. DEFINITION

a) Project Overview

Every year Trillions and Trillions of dollars of transactions are being done using credit cards thru online/regular purchase mode. As per **Nilson report[1]**, Global Card Fraud losses reached $21.84 Billion in year 2015 and will be close to $32 Billion in 2020. Hence it is extremely important that credit card fraud is detected on time which helps in stopping transaction and minimize financial loss. There has been extensive academic/industry research **(For e.g. Andrea Dal Pozzolo et al., 2015[2][3])** on this subject on using different Machine learning techniques to detect credit card fraud. **Purpose of this capstone project is to find potential credit card fraud transactions using machine learning techniques**

b) Problem Statement

As described in project overview, we need to find if credit card transaction is fraudulent or non-fraudulent. As far as machine learning is concerned, it is supervised binary classification problem (as we have labelled dataset) where we need to flag if transaction is normal (Negative) / fraudulent (positive). Additional thing we need to consider is, it is unbalanced classification problem (Anomaly detection problem) where typically 99% of transactions are non-fraudulent one

c) Metrics

Credit card transactions are highly skewed towards normal transaction(negative->99%) and fraud transactions(positive) are very less (<1%). It is highly critical that as much as possible fraud transactions (positive) are identified, so that most of fraud transactions are detected earlier itself and no financial loss happens. Hence considering the problem nature (unbalanced data), below metrics(stressing on positive classes) will be used to evaluate/identify correct algorithm -

1. Precision Score: (True Positives)/(True Positives + False Positives)
2. Recall score: (True Positives)/(True Positives + False Negatives)
3. F1-Score – (2 * precision*recall)/(precision +recall)
4. Area under precision recall curve – To use Sklearn.metrics. average_precision_score [**8]**
5. Accuracy: (True Positives + True Negatives)/ # of samples (Additional metrics to compare against benchmark but not very useful for unbalanced data)

2. ANALYSIS

a) Data Exploration

For this project – Dataset (Reference - https :// www.kaggle.com /dalpozz / creditcard_fraud)) collected and analysed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be/ARTML) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection will be considered (It is public dataset).

The datasets contain transactions made by credit cards in 1 month 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.

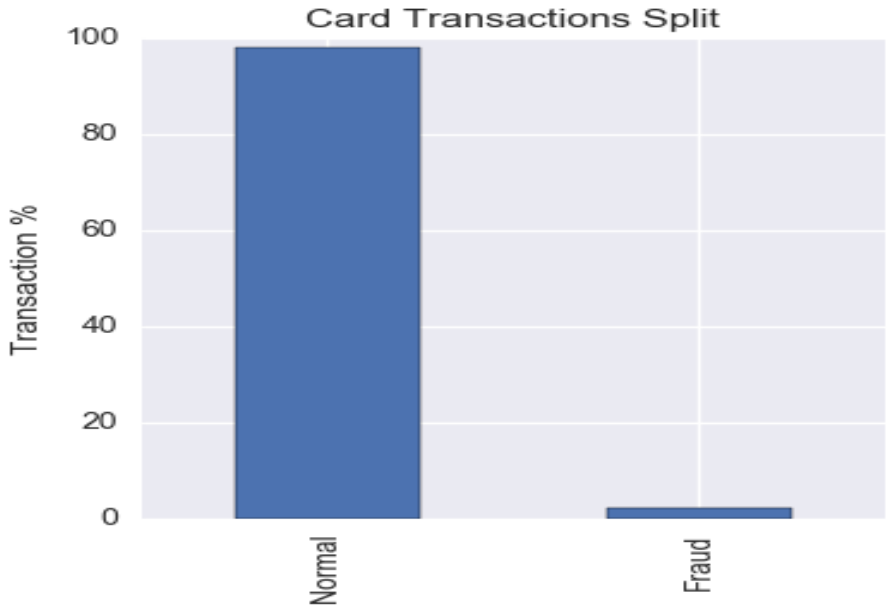
Total no. of transactions	No. of normal transactions	No. of fraud transactions	% of normal transactions	% of fraud transactions
284,807	284,315	492	99.8273%	0.1727%

It contains only numerical input variables which are the result of a PCA transformation. Due to confidentiality issues, original features are not provided. 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'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0.00	-1.36	-0.07	2.54	1.38	-0.34	0.46	0.24	0.10	0.36	...	-0.02	0.28	-0.11	0.07	0.13	-0.19	0.13	-0.02	149.62	0
0.00	1.19	0.27	0.17	0.45	0.06	-0.08	-0.08	0.09	-0.26	...	-0.23	-0.64	0.10	-0.34	0.17	0.13	-0.01	0.01	2.69	0
1.00	-1.36	-1.34	1.77	0.38	-0.50	1.80	0.79	0.25	-1.51	...	0.25	0.77	0.91	-0.69	-0.33	-0.14	-0.06	-0.06	378.66	0
1.00	-0.97	-0.19	1.79	-0.86	-0.01	1.25	0.24	0.38	-1.39	...	-0.11	0.01	-0.19	-1.18	0.65	-0.22	0.06	0.06	123.50	0
2.00	-1.16	0.88	1.55	0.40	-0.41	0.10	0.59	-0.27	0.82	...	-0.01	0.80	-0.14	0.14	-0.21	0.50	0.22	0.22	69.99	0

b) Exploratory Visualization

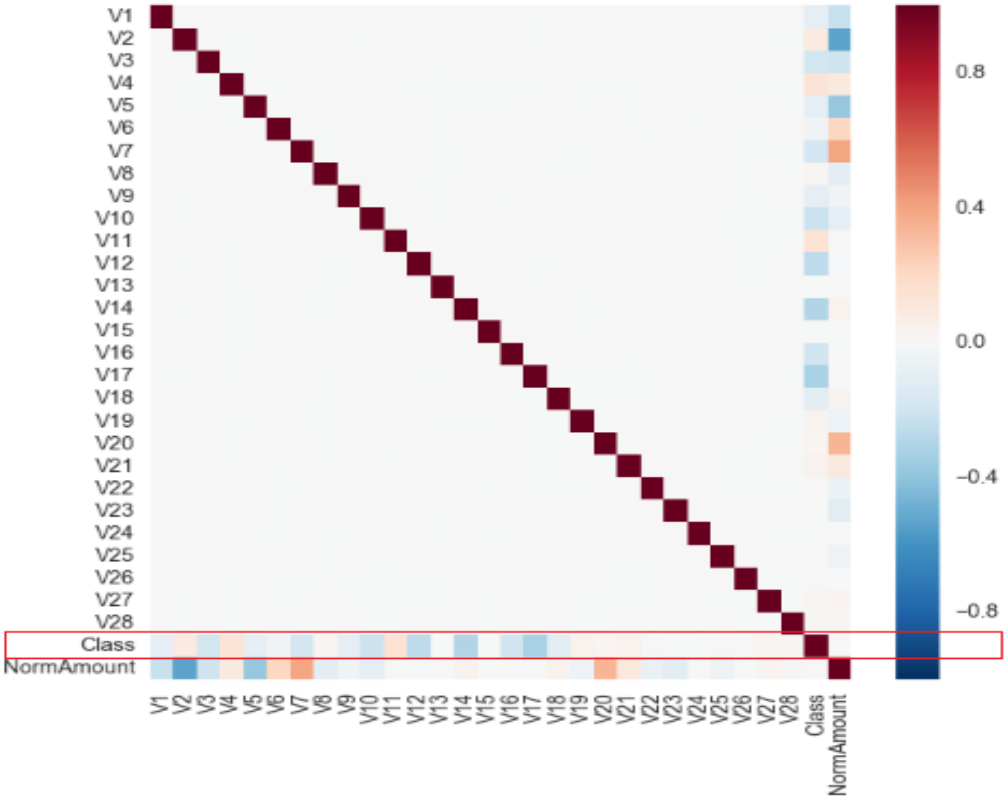
As first step using bar plot we find the number of normal & fraud transactions to confirm imbalanced data. We can clearly see number of fraud transactions are very less (<1%) and normal transactions are greater than 99%



Given the imbalanced dataset, we need to do appropriate data sampling/pre-processing, so that machine learning algorithms don't face any challenge while classifying transactions.

Another important visualization is to look for correlation between class (Target variable) and different predictor variables(V1 to V28, Amount)

Below is the heatmap for correlation between variables -



Based on above heatmap we can see that variables from **V22 to V28**, have negligible correlation with class (target variable). Hence during pre-processing, we can drop this columns for prediction. Additionally don't see any correlations between independent variables(V1 to V28)

c) Algorithms and Techniques

As this is unbalanced classification problem (Anomaly/fraud detection) we would require complex model to get correct predictions. Typically, complex models have high variance and a low bias while simple models have low variance and a high bias. In order to overcome variance/bias problem we can consider ensembles of models. Several works have found that ensembles of models very often outperform a single model (e.g., [4]), because averaging multiple models often reduces the variance of single models.

Well-known ensemble methods are bagging [5] and boosting [6]. So, **Random forest/ Adaboost/ XGBOOST classification algorithms will be considered for this problem**. The model giving best score (Recall/Precision/F1-Score) will finalized as solution. I will also be considering basic Logistic Regression to compare ensemble methods with basic classifier.

Additionally, tried SVM classifier as well (As suggested by Udacity reviewer during review of Capstone proposal document (Step before this project)) but I found algorithm very slow to train (like 15-20

times slower than other methods and when we do GridsSarchCV for tuning purposes, it will be much slower) and metrics score were also not good, hence not considering for this project

To counter unbalanced data, we will use **SMOTE (Synthetic Minority Over -sampling Technique [9]).** SMOTE is an oversampling method, it works by creating synthetic samples from the minor class instead of creating copies, this will help algorithm getting additional training data with positive examples

d) Benchmark

For this particular data set will use basic logistic regression classifier test data scores as Benchmark (Taken from *Credit_Card_Fraud_Detection_Capstone_Proj* Notebook) –

1) Accuracy score – 0.999064
2) Precision Score – 0.695906
3) Recall score – 0.809524
4) F1-Score - – 0.748428
5) Area under precision recall curve – 0.743391

3. METHODOLOGY

a) Data Pre-processing

Based on data exploration, below processing is done

i. In existing data, all columns but Amount column is in normalized state. So Amount column is normalized using StandardScaler() function of sklearn package
ii. **Time** column is sequential number, will not be helpful for prediction. Hence column is dropped
iii. Based on correlation heatmap, we can see that variables from **V22 to V28**, have negligible correlation with **class** (our target variable), hence these columns are dropped

Data was split into train set (70%) and test set (30%).

As mentioned in **Algorithms and Techniques** section, **SMOTE (Synthetic Minority Over -sampling Technique [9])** is used to oversample the training set with fraud transactions (Positive class) by additional 2%(Beyond that ML algorithms were not giving good score)

Post applying **SMOTE** to training dataset, we could see positive class has gone up by 0.1727% to 1.9606% (~11 times increase)

SMOTE Applied Training Data Metrics				
Total no. of transactions	**No. of normal transactions**	**No. of fraud transactions**	**% of normal transactions**	**% of fraud transactions**
202,999	199,019	3980	98.0394%	1.9606%

b) Implementation

As mentioned in *Algorithms and Techniques* Section **Logistic Regression**/**Random forest/ Adaboost/ XGBOOST** are used for initial training on SMOTE sampled data with default parameters.

Only challenge initially was applying default parameters of SMOTE to training data. Initially default parameter of ratio=1(where we get equal distribution for positive and negative class) was applied but algorithms performed badly, after trying multiple ratios, finally settled for .02(2% positive samples of all) based on algorithm performance

Below are the scores obtained

	Training Scores					Testing Scores				
Classifier	**Recall**	**Precision**	**F1**	**Precision-Recall area**	**Accuracy**	**Recall**	**Precision**	**F1**	**Precision-Recall area**	**Accuracy**
Logistic Regression	0.8686	0.9539	0.9093	0.9462	0.9966	0.8095	0.6959	0.7484	0.7434	0.9991
Ada Boost	0.9136	0.9484	0.9306	0.9782	0.9973	0.7891	0.6138	0.6905	0.7682	0.9988
XGBoost	**0.9533**	**0.9786**	**0.9658**	**0.9911**	**0.9987**	**0.8299**	**0.7922**	**0.8106**	**0.8063**	**0.9993**
Random Forest	**0.9965**	**0.9997**	**0.9981**	**1.0000**	**0.9999**	**0.8095**	**0.9370**	**0.8686**	**0.8363**	**0.9996**

Based on above scores(Testing) it is clear that XGBoost and Randomforest classifiers are giving us best scores and will be tuned further for better results

c) Refinement

XGBoost and RandomForest Model parameters were tuned with **Scikit-learn GridSearchCV[10]** with 5 fold cross validation. Parameters tuned are shown below

XG Booster						
Parameters	**Meaning**	**Previous Value**	**Values Tested**	**Performance Improved**	**Best Value**	**Metrics where improvement happened**
max_depth	Maximum tree depth for base learners	10	(8,10)	Yes	8	F1-score ,Precision score,Precision recall area
min_child_weight	Minimum sum of instance weight(hessian) needed in a child	1	(2,4)	Yes	2	F1-score ,Precision score,Precision recall area
reg_lambda	L2 regularization term on weights	1	(0.391, 0.395, 0.399)	No		
n_estimators	Number of trees	100	(150 , 200)	No		

Random Forest Classifier						
Parameters	**Meaning**	**Previous Value**	**Values Tested**	**Performance Improved**	**Best Value**	**Metrics where improvement happened**
max_depth	Maximum tree depth for base learners	10	(25, 30)	No		
min_samples_split	Minimum no. of samples required for tree split	2	(3,4)	No		
min_samples_leaf	Minimum no. of samples required in leaf	1	(4,5)	Yes	5	There is 2% increase in Recall score, but decrease in precision, since Recall is more important here, considering this improvement
n_estimators	Number of trees	10	(12,14,16)	No		

Refer - *Credit_Card_Fraud_Detection_Capstone_Proj.ipynb* for tuning code for XGBooster and *Credit_Card_Fraud_Detection_Capstone_Proj_RandomForest_Tuning.ipynb* for RandomForest

4. RESULTS

a) Model Evaluation and Validation

As mentioned in *Refinement Section* Random forest and XG Booster were tuned using **GridsearchCV[10]**method by varying different parameters with 5-fold cross validation and below are the metrics for pre-tuned and post tuned model for Random forest classifier -

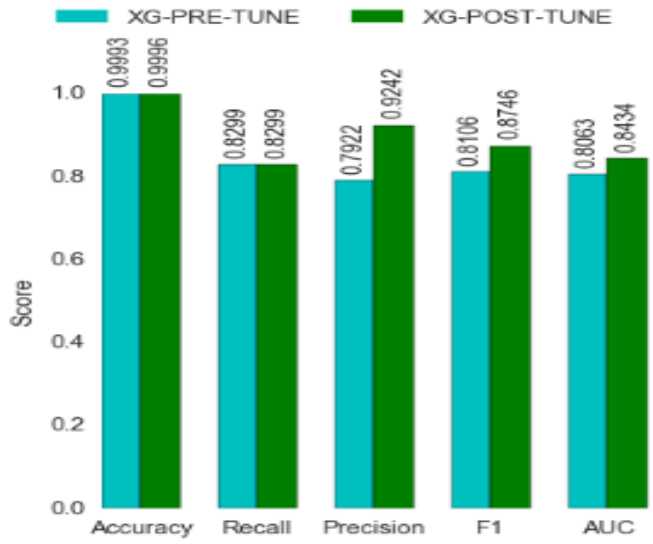
Testing Scores - Random Forest					
Classifier	**Recall**	**Precision**	**F1**	**Area Under Precision-Recall Curve**	**Accuracy**
Random Forest-Pre Tuned	**0.8095**	**0.9370**	**0.8686**	**0.8363**	**0.9996**
Random Forest-Post Tuned	**0.8231**	**0.8832**	**0.8521**	**0.8323**	**0.9995**
Improvement	**3.993%**	**-15.826%**	**-4.857%**	**-1.180%**	**-0.023%**

As metrics preferred is Recall (More positives(Fraud) to be identified) , here tuning helped in improving Recall score by 3.99% for RandomForest classifier(Pls refer notebook -

Credit_Card_Fraud_Detection_Capstone_Proj_RandomForest_Tuning.ipynb where tuning is carried out)

Below is table displaying, pre-and post-tuned metrics for XGBoost

Testing Scores-XGBoost					
Classifier	Recall	Precision	F1	Area Under Precision-Recall Curve	Accuracy
XGBoost-Pre Tuned	0.8299	0.7922	0.8106	0.8063	0.9993
XGBoost-Post Tuned	0.8299	0.9242	0.8746	0.8434	0.9996
Improvement	0.000%	38.834%	18.800%	10.901%	0.076%



Testing Scores Comparison Between XGBoost and Random Forest					
Classifier	Recall	Precision	F1	Area Under Precision-Recall Curve	Accuracy
Random Forest-Post Tuned	0.8231	0.8832	0.8521	0.8323	0.9995
XGBoost-Post Tuned	0.8299	0.9242	0.8746	0.8434	0.9996
XGBoost improvement	2.009%	12.071%	6.604%	3.263%	0.026%

Based on above table/chart we could see significant improvement in Precision, F1 and area under precision recall curve for XGBoost

Finally, above is comparison between XGBoost and RandomForest classifier. **We could see XGBoost has significantly better score than Randomforest classifier, hence we can choose XGBoost as preferred classifier for Credit Card Fraud Detection.**

Final XGB model with parameters (Green colored parameters were obtained through tuning)

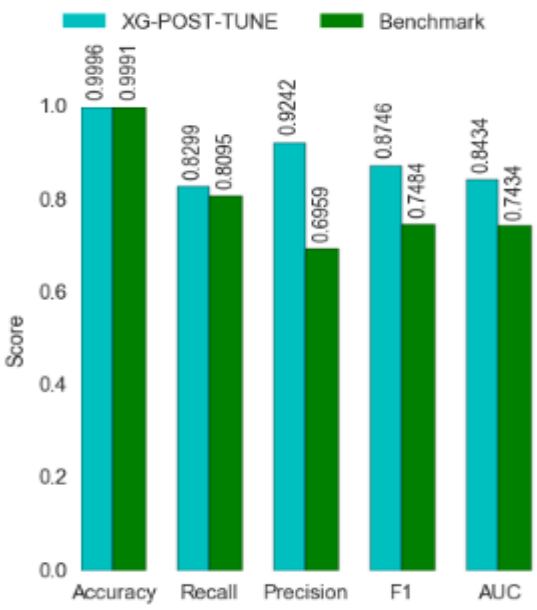
```
XGBClassifier (base_score=0.5, colsample_bylevel=1, colsample_bytree=1,
       gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=8,
       min_child_weight=2, missing=None, n_estimators=100, nthread=-1,
       objective='binary:logistic', reg_alpha=0, reg_lambda=1,
       scale_pos_weight=1, seed=0, silent=True, subsample=1)
```

b) Justification

Initially four classifiers were considered for credit card fraud detection problem namely - Logistic Regression (Used as Project Benchmark),Adaboost, Random forest, XGBOOST. During initial training (As mentioned in Implementation section), found that Randomforest and XGBoost were giving better scores. Hence Random forest and XGBoost were further tuned as discussed in Refinement section. Finally, found that XGBoost was giving best metrics score and selected as preferred classifier for the problem.

Below is comparison between benchmark (Logistic Regression) and XGBoost classifier –

Testing Scores Comparison Between XGBoost and Benchmark Score					
Classifier	**Recall**	**Precision**	**F1**	**Area Under Precision-Recall Curve**	**Accuracy**
Benchmark	**0.8095**	**0.6959**	**0.7484**	**0.7434**	**0.9991**
XGBoost	**0.8299**	**0.9242**	**0.8746**	**0.8434**	**0.9996**
XGBoost improvement	**6.00%**	**67.16%**	**37.10%**	**29.41%**	**0.15%**

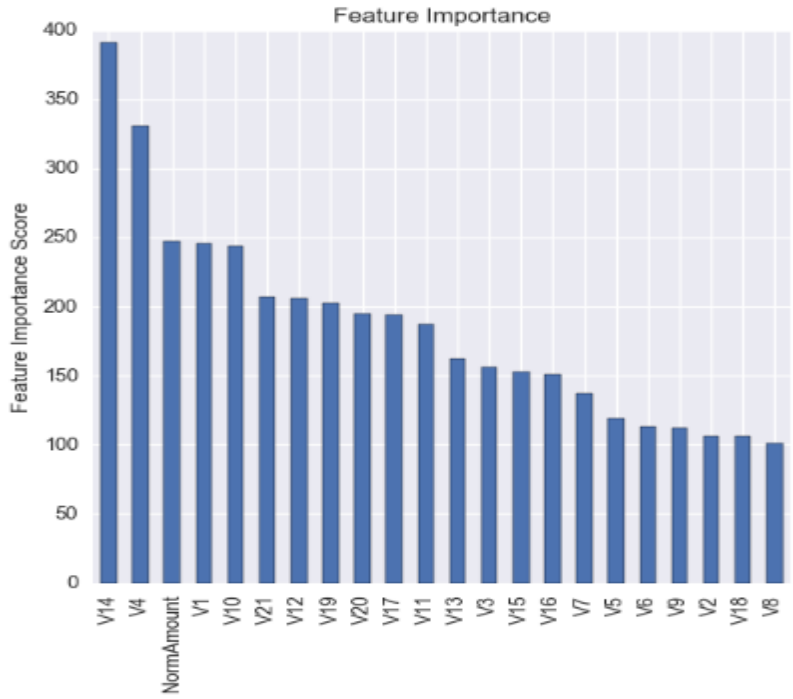


Based on above table it is clear that XGBoost gives us significant improvement in score when compared to benchmark (Also other classifiers like Adaboost, Randomforest), hence justifies in selecting XGBoost as preferred classifier for Credit Card Fraud Detection

5. CONCLUSION

a) Free Form Visualization

Importance of each feature was visualized with the following plot -



To test the performance using important features, the final model was run with dropping last 3 important features(V2,V18,V8). The key metrics of this fewer-feature model was 2-3% less. Therefore, all features were kept and used. (Also, based on initial analysis we had already dropped V22 to V28 columns based on low correlation)

b) Reflection

This problem is highly imbalanced classification problem, where we have more than 99% transactions as normal and <1% as fraud. Data which we got was, PCA applied columns (As original columns were not shared due to privacy reasons)

Given data skewedness, classifiers will tend to prefer normal (negative class) transactions and will have challenge identifying fraud positive classes. Hence to overcome this **SMOTE (Synthetic Minority Over -sampling Technique [9])** method was used to oversample training data with positive classes, so that classifiers have more positive class to work with.

As metrics benchmark, logistic regression metrics was used (As could not find industry wide benchmark) and ensemble methods (Adaboost, RandomForest, XGBoost) were trained on data as ensemble methods tend to predict better for these kinds of problems**[4]**. RandomForest and XGBoost gave good metrics, were further selected for tuning steps

RandomForest and XGBoost were tuned using GridSearchCV 5-fold cross validation data set. Also, I am tuning parameters one by one as tuning multiple parameters at same time was causing program to run for very long time and required lots of memory.

Finally, between 2 classifiers, based on better score, **XGBoost was chosen (Refer *Justification section*) as preferred classifier for the problem.**

c) Improvement

Overall, the final XGBoost model provided good score on solving the challenging problem, at least better than the logistics regression (our benchmark)/Randomforest classifier/AdaBoost classifiers.
As further improvement, we can consider using Deep Learning/Meta classifier model (Like combination of k-nearest neighbour, decision tree, naïve Bayesian algorithms) to predict fraud detections.
Another big improvement can happen via feature engineering. If we are given raw features, we can try out different feature engineering techniques for better result. Currently due to security/privacy concerns, we have been given PCA applied data and original raw features are not there, hence feature engineering scope was less
Finally, during real project scenario, threshold percentage to mark transactions as fraud can be discussed with business team and increase recall score accordingly (Though it might decrease precision)

REFERENCE

1) https://www.nilsonreport.com/index.php
2) Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit Card Fraud Detection with Alert-Feedback Interaction. Submitted to IEEE Transactions on Neural Networks and Learning Systems.
3) Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. Expert Systems with Applications, 41(10):4915-4928, 2014.
4) Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine learning, 36(1-2):105–139, 1999.
5) Leo Breiman. Bagging predictors. Machine learning, 24(2):123–140, 1996.
6) Robert E Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. Annals of statistics, pages 1651–1686, 1998.
7) http://scikit-learn.org /stable /modules /generated / sklearn.model_selection.GridSearchCV .html
8) http://scikit-learn.org /stable /modules /generated / sklearn. metrics. average_ precision_ score. html
9) http://www.jair.org/papers/paper953.html
10) http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html