Credit card fraud detection with Machine Learning

Artyom Harutyunyan[1]

Yerevan State University, Yerevan

Introduction

Payments fraud represents a significant and growing issue in the United States and abroad. There was more than $8 billion in fraud over U.S. non-cash payment systems in 2015, a 37 percent increase from 2012. Financial institutions and payment operators are increasingly relying on machine learning algorithms to develop efficient and effective fraud detection systems. In this project, I implement and asses the performance of various machine learning models, including logistic regression, random forest and gradient boosting, using a rich dataset from Kaggle. The dataset contains approximately 300000 credit card transactions occurring over two days in Europe.

The key problem with payments fraud data is class imbalance. In the dataset almost 99.8 percent of transactions labeled as legitimate, while 0.2 percent as fraudulent. That wide gap, make it difficult for standard techniques to learn to distinguish between two classes. In similar cases, when data is extremely imbalanced, traditional accuracy metric become not effective. For achieving good results, in this project I focused on exploring different methods for improving performance metrics for imbalanced classification.

Related Academic Work

There is a many academic works related with finding solutions for problems regarding fraud detection in the payments industry. Bhattacharyya et. al. note that while fraud detection algorithms widely used by financial institutions the breadth of studies on the use of machine learning techniques for payment fraud detection is limited, possibly due to the sensitive nature of the data. Their study concluded that random forests, though not widely deployed, may outperform more traditional methods. Roy found that network size is the strongest driver of a neural network's performance for fraud classification. Chaudhary note that no single algorithm is pareto optimal across all performance metrics; each has a unique set of strengths and weaknesses. Class imbalance in Machine Learning is a wide area of research. Various techniques are deployed to address this problem, including oversampling the minority class, undersampling the majority class, generating synthetic samples of the minority class,

[1] Master of Applied Statistics and Data Science Master program from Yerevan State University: Faculty of Mathematics and Mechanics

and adjusting the relative cost of misclassifying the minority and majority class. Japkowicz and Stephen conclude that the effects of class imbalance depend on the severity of the imbalance, sample size and the classifier used, noting that sampling techniques may negatively affect the performance of certain classifiers. Cui highlight research suggesting that oversampling can lead to overfitting, while added noise from synthetic data generation can reduce predictive performance

Data and Features

The dataset[2] 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 and 31 features. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, the original features and more background information about the data is not available. 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, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

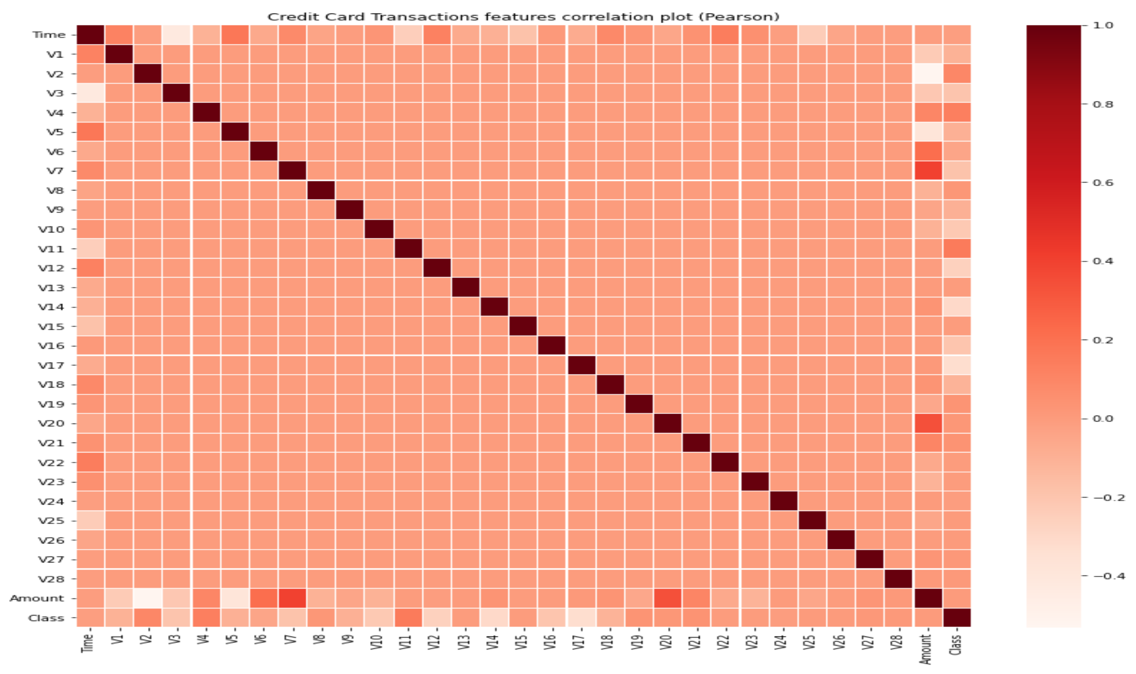
Table 1: Class distribution

	Number	percent
All transactions	**284.807**	**100**
Fraud	492	0.173
Nonfraud	284135	99.827

‘Time’ and ‘Amount’ features originally were in different scales than other features. For scaling this features I used Robust Scaler which is robust to outlier. This scaler removes the median and scales the data according to the quantile range. I prefer this method to Standard Scaler because outliers can often influence the sample mean / variance in a negative way. In such cases, the median and the interquartile range often give better results.[3]

[2] https://www.kaggle.com/mlg-ulb/creditcardfraud
[3] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

A correlation analysis show that class label correlated with some PCA features and PCA features is uncorrelated with each other because of their orthogonality and already scaled with zero mean. In overall dataset didn't need more preprocessing.



Methods

I used Python programming language[4] for all parts of the analysis. My primary models included Logistic Regression, Random Forest and gradient boosting. Also I provided some EDA where I used DBSCAN clustering method for anomaly detection. In analysis was used synthetic oversampling method – ADASYN (Adaptive Synthetic Sampling). So, dataset is very imbalanced and model's accuracy score show incorrect result. For example, Logistic Regression that work well on binary classification tasks show almost 99.9 percent accuracy despite incorrectly classifying many fraudulent transactions. For that reason, I focused on specific classification performance metrics such as Matthews correlation coefficient, precision, recall, confusion matrix and Average Precision (Summary of Area under Precision Recall Curve). The main goal to get well performance on this metrics, especially on precision and recall.

Models

Logistic Regression

Logistic regression is a widely used discriminative learning algorithm that uses a hypothesis of the form:

[4] See my Python code https://github.com/artyom-dev/ML-Project-

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

In its simplistic form, logistic regression fits a linear decision boundary in the feature space to model a binary dependent variable. Parameters are fit by maximum likelihood, where the log likelihood is given by:

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

Random Forest

Tree-based models stratify the predictor space into simple regions that can be used to classify the dependent variable. Although simple tree methods are generally not competitive with other machine learning techniques, combining a large number of trees using methods such as bagging, boosting, and random forests often results in significant improvements in prediction accuracy, at the expense of some loss interpretation.

Light GBM

Light GBM[5] is a gradient boosting framework that uses tree based learning algorithm**.** Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm. Light GBM is prefixed as 'Light' because of its high speed. Light GBM can handle the large size of data and takes lower memory to run. Another reason of why Light GBM is popular is because it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development.

ADASYN

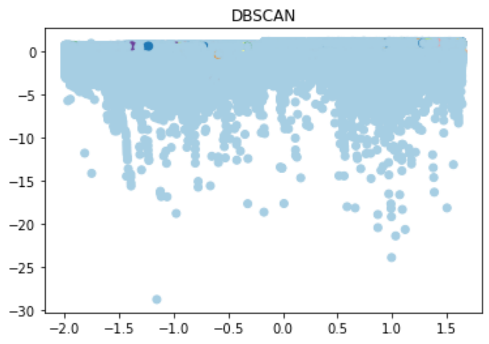
(ADASYN) sampling approach for learning from imbalanced data sets. The essential idea of ADASYN is to use a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn compared to those majority examples that are easier to learn. As a result, the ADASYN approach improves learning with respect to the data distributions in two ways: (1) reducing the bias introduced by the class imbalance, and (2) adaptively shifting the classification decision boundary toward the difficult examples. Simulation analyses on several machine learning data sets show the effectiveness of this method across five evaluation metrics

[5] https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

Experiments

Firstly, I divided data on train 80 / test 20 proportion. After it I try to understand behavior of frauds and whether there are significant difference between legitimate and fraud transaction's behaviors ? (whether they are outliers ?). For that, I provided training on DBSCAN (Density-Based Spatial Clustering of Applications with Noise. Finds core samples of high density and expands clusters from them. Good for data which contains clusters of similar density[6]) model and visual it's result:

Density of DBSCAN



As hyperparameters I choose distance between two samples = 0.4 and the number of samples in a neighborhood = 20. In result density we can see that there is no datapoints outside of density. In this step we can conclude that fraud's transaction behavior not very differs from legitimate transaction behavior.

As my main performance metric I choose Matthew's correlation coefficient which is show very explanable results of model in case of imbalance datasets. The Matthews correlation coefficient is used in machine learning as a measure of the quality of binary and multiclass classifications. It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. The statistic is also known as the phi coefficient.[7]

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

Using and comparing other performance metrics and algorithms mentioned above I trained each model in three ways.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html

1) On Original imbalanced data
2) On oversampled data with ADASYN
3) On imbalanced data with tuned Class Weights

After it I test each model on cross validation with 5 folds. The best model was chosen for final Test data which had remained untouched throughout the training process. During Class Weight tuning I used ratio of Negative to Positive classes and assign it's value to minority class. Unfortunately, because of long training time of Random Forest (several hours) I didn't tried tune many hyperparameters but I try some to prevent overfitting (max_depth). In case of Light GBM, since it has trained very fast I tried to tune it with some hyperparameters for faster training time and avoiding overfitting. For that reason, I tried different values of num_leaves, learning_rate and max_depth hyperparameters. The best model based on performance metric was model with num_leaves = 100 and max_depth = 10. I trained Logistic Regression with default hyperparameters.

Results

After training and testing on cross validation I get the following results:

Logistic Regression

	On original imbalanced data	**On Oversampled data with ADASYN**	**On Imbalanced Data with tuned Class Weights**
Precision	0.6269035532994924	0.8897058823529411	0.06281893366179835
Recall	0.8884892086330936	0.6142131979695431	0.9060913705583756
MCC	0.745969890933559	0.7388786033224832	0.2351631736187096
ROC AUC SCORE	0.8133836301038968	0.8070406507145946	0.9413376690471201
Average Precision	0.77	0.73	0.74

Random Forest

	On original imbalanced data	**On Oversampled data with ADASYN**	**On Imbalanced Data with tuned Class Weights**
Precision	0.9409937888198758	0.9435736677115988	0.8646408839779005
Recall	0.7690355329949239	0.7639593908629442	0.7944162436548223
MCC	0.8504537052661266	0.8488021324840862	0.8285020827766065
ROC AUC SCORE	0.8844759992596831	0.8819401264693659	0.8971004063194556

Average Precision	0.99	0.96	0.97

Light GBM

	On original imbalanced data	On Oversampled data with ADASYN	On Imbalanced Data with tuned Class Weights
Precision	0.600451467268623	0.2593068035943517	0.8962536023054755
Recall	0.6751269035532995	0.5126903553299492	0.7893401015228426
MCC	0.6360299501273342	0.3631183210193419	0.8408442961226863
ROC AUC SCORE	0.8371743569826061	0.7550767726019061	0.89 4590912837209
Average Precision	0.75	0.35	1.00

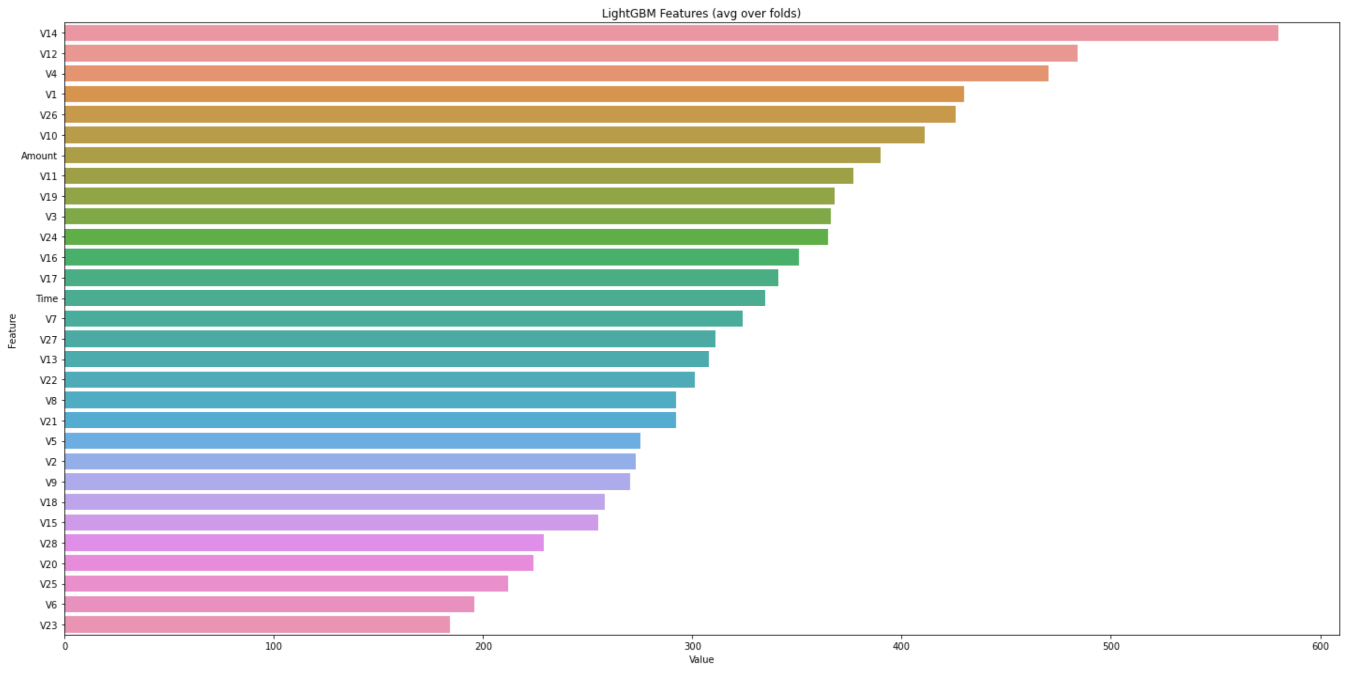
Conclusion

For the final choice of model, we must pay attention to models with higher precision and recall. Only 3 models – Random Forest on Original dataset / ADASYN and Light GBM with tuned Class Weight show productive results. As a finalist, I chose Light GBM, because it has shown a high recall, precision as well as fast training time. Also it has good MCC result, which testifies right prediction on minority class. Random Forest also show good result but because of low training time and lower recall than Light GBM I decide do not continue with it.

Prediction on Test data with Light GBM with tuned Class weight show the following result:

Precision	0.9746835443037974
Recall	0.7857142857142857
MCC	0.8749276812909632
ROC AUC SCORE	0.892839557038347
Average Precision	0,85

Important features by Light GBM is the following:



So, Light GBM show good result on Test data. If you have enough time for training or GPU, you can try a random forest that can improve this indicator or different hyperparameter tuning with Light GBM .

References

Siddhartha Bhattacharyya , Sanjeev Jha , Kurian Tharakunnel , J. Christopher Westland, Data mining for credit card
fraud: A comparative study, Decision Support Systems, v.50 n.3, p.602-613, February, 2011.

Roy, A., Sun, J., Mahoney, R., Alonzi, L., Adams, S., and Beling, P. Deep learning detecting fraud in credit card transactions. Systems and Information Engineering Design Symposium (SIEDS) (2018), pp. 129–134.

Japkowicz, N., and S. Stephen. “The Class Imbalance Problem: A Systematic Study.” Intelligent Data Analysis, vol. 6,
no. 5, pp. 429–449.

https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

https://lightgbm.readthedocs.io/en/latest/#:~:text=LightGBM%20is%20a%20gradient%20boosting,training%20speed%20and%20higher%20efficiency.&text=Support%20of%20parallel%20and%20GPU,of%20handling%20large%2Dscale%20data.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html

https://www.kaggle.com/mlg-ulb/creditcardfraud

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html