# Fraud Detection using Machine Learning

Aditya Oza - aditya19@stanford.edu

***Abstract*— Recent research has shown that machine learning techniques have been applied very effectively to the problem of payments related fraud detection. Such ML based techniques have the potential to evolve and detect previously unseen patterns of fraud. In this paper, we apply multiple ML techniques based on Logistic regression and Support Vector Machine to the problem of payments fraud detection using a labeled dataset containing payment transactions. We show that our proposed approaches are able to detect fraud transactions with high accuracy and reasonably low number of false positives.**

## I. INTRODUCTION

We are living in a world which is rapidly adopting digital payments systems. Credit card and payments companies are experiencing a very rapid growth in their transaction volume. In third quarter of 2018, PayPal Inc (a San Jose based payments company) processed 143 billion USD in total payment volume [4]. Along with this transformation, there is also a rapid increase in financial fraud that happens in these payment systems.

An effective fraud detection system should be able to detect fraudulent transactions with high accuracy and efficiency. While it is necessary to prevent bad actors from executing fraudulent transactions, it is also very critical to ensure genuine users are not prevented from accessing the payments system. A large number of false positives may translate into bad customer experience and may lead customers to take their business elsewhere.

A major challenge in applying ML to fraud detection is presence of highly imbalanced data sets. In many available datasets, majority of transactions are genuine with an extremely small percentage of fraudulent ones. Designing an accurate and efficient fraud detection system that is low on false positives but detects fraudulent activity effectively is a significant challenge for researchers.

In our paper, we apply multiple binary classification approaches - Logistic regression , Linear SVM and SVM with RBF kernel on a labeled dataset that consists of payment transactions.Our goal is to build binary classifiers which are able to separate fraud transactions from non-fraud transactions. We compare the effectiveness of these approaches in detecting fraud transactions.

## II. RELEVANT RESEARCH

Several ML and non-ML based approaches have been applied to the problem of payments fraud detection. The paper [1] reviews and compares such multiple state of the art techniques, datasets and evaluation criteria applied to this problem. It discusses both supervised and unsupervised ML based approaches involving ANN (Artificial Neural Networks), SVM (Support Vector machines) ,HMM (Hidden Markov Models), clustering etc. The paper [5] proposes a rule based technique applied to fraud detection problem. The paper [3] discusses the problem of imbalanced data that result in a very high number of false positives and proposes techniques to alleviate this problem. In [2], the authors propose an SVM based technique to detect metamorphic malware. This paper also discusses the problem of imbalanced data sets - fewer malware samples compared to benign files - and how to successfully detect them with high precision and accuracy.

## III. DATASET AND ANALYSIS

In this project, we have used a Kaggle provided dataset [8] of simulated mobile based payment transactions. We analyze this data by categorizing it with respect to different types of transactions it contains. We also perform PCA - Principal Component Analysis - to visualize the variability of data in two dimensional space. The dataset contains five categories of transactions labeled as 'CASH IN', 'CASH OUT', 'DEBIT', 'TRANSFER' and 'PAYMENT' - details are provided in table I

| Transaction Type | Non-fraud transactions | Fraud transactions | Total |
|---|---|---|---|
| CASH IN | 1399284 | 0 | 1399284 |
| CASH OUT | 2233384 | 4116 | 2237500 |
| TRANSFER | 528812 | 4097 | 532909 |
| DEBIT | 41432 | 0 | 41432 |
| PAYMENT | 2151494 | 0 | 2151494 |
| TOTAL | 6354407 | 8213 | 6362620 |

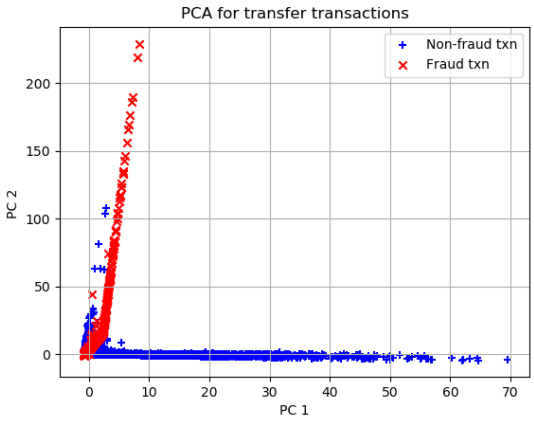
TABLE I: Paysim dataset statistics

Paysim dataset consists of both numerical and categorical features like transaction type ,amount transferred, account numbers of sender and recipient accounts. In our experiments we use the following features to train our models.

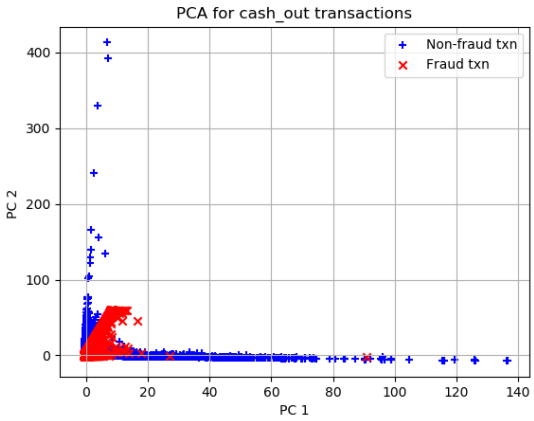
1) Transaction type
2) Transaction amount
3) Sender account balance before transaction
4) Sender account balance after transaction
5) Recipient account balance before transaction
6) Recipient account balance after transaction

The dataset consists of around 6 million transactions out of which 8312 transactions are labeled as fraud. It is highly imbalanced with $0.13$ percent fraud transactions. We display the result of performing two dimensional PCA on subsets for two transaction types that contain frauds - TRANSFER and CASH OUT transactions.

The PCA decomposition of TRANSFER transactions shows a high variability across two principal components for non-fraud and fraud transactions. This gave us confidence that TRANSFER dataset can be linearly separable and our chosen algorithms - Logistic regression and Linear SVM are likely to perform very well on such a dataset. In the results section, we'll see that this is indeed the case.




(a) TRANSFER transactions




(b) CASH OUT transactions

Fig. 1: PCA decomposition of Paysim data

## IV. Method

Our goal is to separate fraud and non-fraud transactions by obtaining a decision boundary in the feature space defined by input transactions. Each transaction can be represented as vector of its feature values. We have built binary classifiers using Logistic regression, linear SVM and SVM with RBF kernels for TRANSFER and CASH OUT sets respectively.

### A. Logistic Regression

Logistic regression is a technique used to find a linear decision boundary for a binary classifier. For a given input feature vector $x$, a logistic regression model with parameter $\theta$ classifies the input $x$ using the following hypothesis $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$ where $g$ is known as Sigmoid function. For a binary classification problem, the output $h_\theta(x)$ can be interpreted as a probability of $x$ as belonging to class 1. The logistic loss function with respect to parameters $\theta$ can be given as

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} log\big(1 + exp(-y^{(i)}\theta^T x^{(i)})\big)$$

### B. Support Vector Machine

Support vector machine creates a classification hyper-plane in the space defined by input feature vectors. The training process aims to determine a hyper-plane that maximizes geometric margin with respect to labeled input data. SVMs optimization problem can be characterized by

$$min_{\gamma,w,b}\frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\epsilon_i$$

$$s.t.\ y^{(i)}(w^T x^{(i)} + b) >= 1 - \epsilon_i,\ \ i = 1, ..., m$$

$$\epsilon_i >= 0,\ \ i = 1, ..., m$$

In this project we use two variants of SVM - linear SVM and SVM based on RBF kernel. An SVM based on RBF kernel can find a non-linear decision boundary in input space. The RBF kernel function on two vectors $x$ and $z$ in the input space can be defined as

$$K(x, z) = exp\bigg(-\frac{||x - z||^2}{2\sigma^2}\bigg)$$

.

### C. Class weights based approach

We assign different weights to samples belonging fraud and non-fraud classes for each of the three techniques respectively. Such an approach has been used to counter data imbalance problem - with only 0.13 percent fraud transactions available to us. In a payments fraud detection system, it is more critical to catch potential fraud transactions than to ensure all non-fraud transactions are executed smoothly. In our proposed approaches, we penalize mistakes made in misclassifying fraud samples more than misclassifying non-fraud samples. We trained our models (for each technique) using higher class weights for fraud samples compared to non-fraud samples.

We fine tune our models by choosing class weights to obtain desired/optimal balance between precision and recall scores on our fraud class of samples. We have chosen class weights such that we do not have more than around 1 percent of false positives on CV set. This design trade off enables us to establish a balance between detecting fraud transactions with high accuracy and preventing large number of false positives. A false positive, in our case, is when a non-fraud transaction is misclassified as a fraudulent one.

## V. Experiments

In this section,we describe our dataset split strategy and training ,validation and testing processes that we have implemented in this work. All software was developed using Scikit-learn [7] ML library.

### A. Dataset split strategy

We divided our dataset based on different transaction types described in dataset section. In particular, we use TRANSFER and CASH OUT transactions for our experiments since they contain fraud transactions. For both types, we divided respective datasets into three splits - 70 percent for training, 15 percent for CV and 15 percent for testing purposes. We use stratified sampling in creating train/CV/test splits. Stratified sampling allows us to maintain the same proportion of each class in a split as in the original dataset. Details of the splits are mentioned in tables II and III.

| TRANSFER | | | |
|---|---|---|---|
| Split | Fraud | Non fraud | Total |
| Train | 2868 | 370168 | 373036 |
| CV | 614 | 79322 | 79936 |
| Test | 615 | 79322 | 79937 |
| Total | 4097 | 528812 | 532909 |

TABLE II: Dataset split details

### B. Model training and tuning

We employed class weight based approach as described in previous section to train our each of our models. Each model was trained multiple times using increasing weights for fraud class samples. At the end of each iteration, we evaluated our trained models by measuring their performance on CV split. For each model, we chose the class weights which gave us highest recall on fraud class with not more than $\sim 1$ percent false positives.

Finally, we used the models trained using chosen set of class weights to make predictions on our test dataset split. In the next section, we elaborate on our choice of class weights based on their performance on CV set. We also discuss their performance on train and test sets.

## VI. Results and Discussion

In this section, we discuss results obtained in training, validation and testing phases. We evaluated performance of our models by computing metrics like recall, precision, f1 score, area under precision-recall curve (AUPRC).

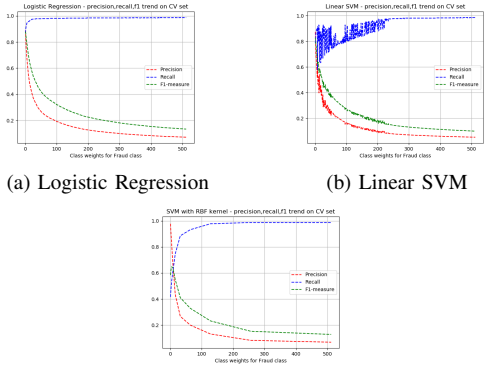
### A. Class weights selection

In our experiments, we used increasing weights for fraud samples. We initially considered making class weights equal to imbalance ratio in our dataset. This approach seemed to give good recall but also resulted in very high number of false positives - $>>1$ percent - especially for CASH OUT. Hence, we did not use this approach and instead tuned our models by trying out multiple combinations of weights on our CV split.

| CASH OUT | | | |
|---|---|---|---|
| Split | Fraud | Non fraud | Total |
| Train | 2881 | 1563369 | 1566250 |
| CV | 618 | 335007 | 335625 |
| Test | 617 | 335008 | 335625 |
| Total | 4116 | 2233384 | 2237500 |

TABLE III: Dataset split details

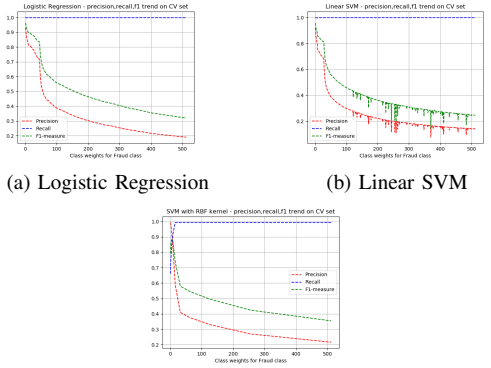
Overall, we observed that higher class weights gave us higher recall at the cost of lower precision on our CV split. In figure 2, we show this observed behavior for CASH OUT transactions.




(a) Logistic Regression (b) Linear SVM

(c) SVM with RBF kernel

Fig. 2: CASH OUT - Precision,Recall, F1 trend for increasing fraud class weights

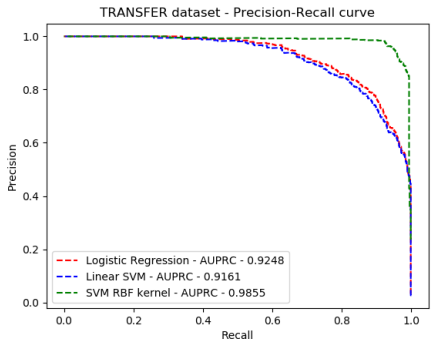



(a) Logistic Regression (b) Linear SVM

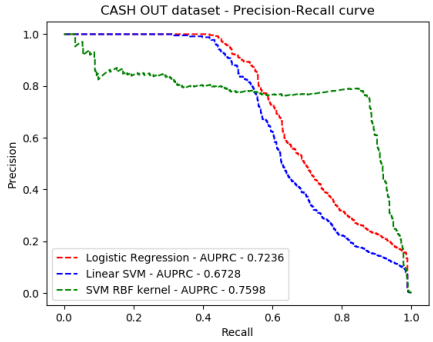
(c) SVM with RBF kernel

Fig. 3: TRANSFER - Precision,Recall, F1 trend for increasing fraud class weights

For TRANSFER dataset, the effect of increasing weights is less prominent, in particular for Logistic Regression and Linear SVM algorithms. That is, equal class weights for fraud and non-fraud samples give us high recall and precision scores. Based on these results, we still chose higher weights for fraud samples to avoid over-fitting on CV set. Figure 4 shows precision-recall curves obtained on CV set using chosen class weights for all three algorithms. Table IV summarizes these results via precision,recall,f1-measure and AUPRC scores. We chose to plot precision/recall curves (PRC) over ROC as PRCs are more sensitive to misclassifications when dealing with highly imbalanced datasets like ours. The final values of selected class weights are mentioned in table V.

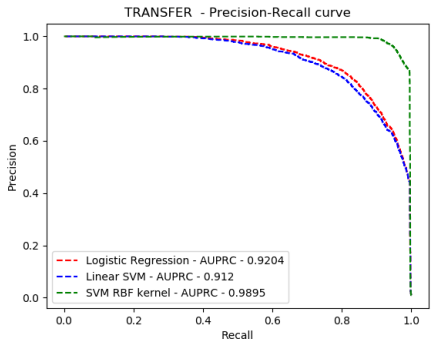



(a) TRANSFER

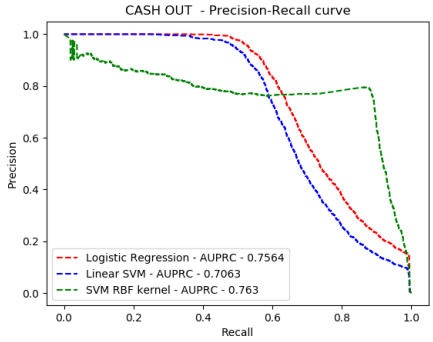



(b) CASH OUT

Fig. 4: CV set - Precision-Recall curve




(a) TRANSFER




(b) CASH OUT

Fig. 5: Train set - Precision-Recall curve

| TRANSFER | | | | |
|---|---|---|---|---|
| Algorithm | Recall | Precision | f1-measure | AUPRC |
| Logistic Regression | 0.9983 | 0.4416 | 0.6123 | 0.9248 |
| Linear SVM | 0.9983 | 0.4432 | 0.6139 | 0.9161 |
| SVM with RBF kernel | 0.9934 | 0.5871 | 0.7381 | 0.9855 |
| CASH OUT | | | | |
| Algorithm | Recall | Precision | f1-measure | AUPRC |
| Logistic Regression | 0.9822 | 0.1561 | 0.2692 | 0.7235 |
| Linear SVM | 0.9352 | 0.1263 | 0.2226 | 0.6727 |
| SVM with RBF kernel | 0.9773 | 0.1315 | 0.2318 | 0.7598 |

TABLE IV: Results on CV set

| TRANSFER | | | | |
|---|---|---|---|---|
| Algorithm | Recall | Precision | f1-measure | AUPRC |
| Logistic Regression | 0.9958 | 0.4452 | 0.6153 | 0.9204 |
| Linear SVM | 0.9958 | 0.4431 | 0.6133 | 0.9121 |
| SVM with RBF kernel | 0.9958 | 0.6035 | 0.7515 | 0.9895 |
| CASH OUT | | | | |
| Algorithm | Recall | Precision | f1-measure | AUPRC |
| Logistic Regression | 0.9847 | 0.1541 | 0.2664 | 0.7564 |
| Linear SVM | 0.9361 | 0.1225 | 0.2119 | 0.7063 |
| SVM with RBF kernel | 0.9875 | 0.1355 | 0.2383 | 0.7631 |

TABLE VI: Results on Train set

| TRANSFER | | |
|---|---|---|
| Algorithm | non-fraud | fraud |
| Logistic regression | 1 | 70 |
| Linear SVM | 1 | 39 |
| SVM with RBF kernel | 1 | 16 |
| CASH OUT | | |
| Algorithm | non-fraud | fraud |
| Logistic regression | 1 | 145 |
| Linear SVM | 1 | 132 |
| SVM with RBF kernel | 1 | 128 |

TABLE V: Class weights

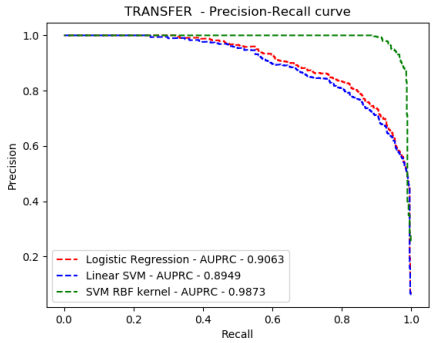
### *B. Results on train and test sets*

In this section, we discuss results on train and test sets using chosen class weights. Figure 5 and table VI summarize the results on train set.

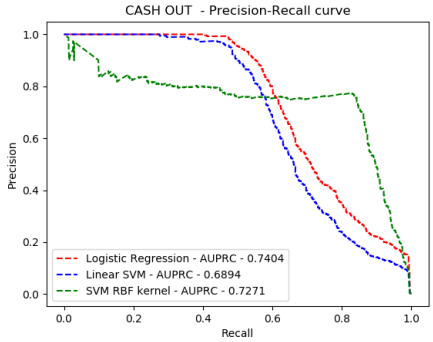
Similarly, figure 6 and table VII summarize the results on test set. We get very high recall and AUPRC scores for TRANSFER transactions with $\sim 0.99$ recall score for all three algorithms. In particular, SVM with RBF kernel gives us the best AUPRC value because it has much higher precision compared to the other two algorithms. Table VIII displays corresponding confusion matrices obtained on test set of TRANSFER. We are able to detect more than 600 fraud transactions for all three algorithms with less than 1 percent false positives. TRANSFER transactions had shown a high variability across their two principal components when we performed PCA on it. This set of transactions seemed to be linearly separable - with all three of our proposed algorithms expected to perform well on it. We can see this is indeed the case.

For CASH OUT transactions, we obtain less promising results compared to TRANSFER for both train and test sets.

Logistic regression and linear SVM have similar performance (and hence similar linear decision boundaries and PR curves). SVM with RBF gives a higher recall but with lower precision on average for this set of transactions. A possible reason for this outcome could be non-linear decision boundary computed using RBF kernel function. However, for all three algorithms, we can obtain high recall scores if we are more tolerant to false positives. In the real world, this is purely a design/business decision and depends on how many false positives is a payments company willing to tolerate.




(a) TRANSFER




(b) CASH OUT

Fig. 6: Test set - Precision-Recall curve

| TRANSFER | | | | |
|---|---|---|---|---|
| Algorithm | Recall | Precision | f1-measure | AUPRC |
| Logistic Regression | 0.9951 | 0.4444 | 0.6144 | 0.9063 |
| Linear SVM | 0.9951 | 0.4516 | 0.6213 | 0.8949 |
| SVM with RBF kernel | 0.9886 | 0.5823 | 0.7329 | 0.9873 |
| CASH OUT | | | | |
| Algorithm | Recall | Precision | f1-measure | AUPRC |
| Logistic Regression | 0.9886 | 0.1521 | 0.2636 | 0.7403 |
| Linear SVM | 0.9411 | 0.1246 | 0.2201 | 0.6893 |
| SVM with RBF kernel | 0.9789 | 0.1321 | 0.2327 | 0.7271 |

TABLE VII: Results on Test set

Overall, we observe that all our proposed approaches seem to detect fraud transactions with high accuracy and low false positives - especially for TRANSFER transactions. With more tolerance to false positives, we can see that it can perform well on CASH OUT transactions as well.

TABLE VIII: Confusion matrices

(a) Logistic Regression

| True \ Pred | - | + |
|---|---|---|
| - | 78557 | 765 |
| + | 3 | 612 |

(b) Linear SVM

| True \ Pred | - | + |
|---|---|---|
| - | 78579 | 743 |
| + | 3 | 612 |

(c) SVM with RBF kernel

| True \ Pred | - | + |
|---|---|---|
| - | 78886 | 436 |
| + | 7 | 608 |

## VII. Conclusion and Future Work

In fraud detection, we often deal with highly imbalanced datasets. For the chosen dataset (Paysim), we show that our proposed approaches are able to detect fraud transactions with very high accuracy and low false positives - especially for TRANSFER transactions. Fraud detection often involves a trade off between correctly detecting fraudulent samples and not misclassifying many non-fraud samples. This is often a design choice/business decision which every digital payments company needs to make. We've dealt with this problem by proposing our class weight based approach.

We can further improve our techniques by using algorithms like Decision trees to leverage categorical features associated with accounts/users in Paysim dataset. Paysim dataset can also be interpreted as time series. We can leverage this property to build time series based models using algorithms like CNN. Our current approach deals with entire set of transactions as a whole to train our models. We can create user specific models - which are based on user's previous transactional behavior - and use them to further improve our decision making process. All of these, we believe, can be very effective in improving our classification quality on this dataset.

## APPENDIX

**Github code link** - https://github.com/aadityaoza/CS-229-project

## VIII. Acknowledgement

I would like to thank Professor Ng and entire teaching staff for a very well organized and taught class. In particular, I would like to thank my project mentor, Fantine, for her valuable insights and guidance during the course of this project.

## References

[1] A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective - *Samaneh Sorournejad, Zojah, Atani et.al - November 2016*

[2] Support Vector machines and malware detection - *T.Singh,F.Di Troia, C.Vissagio , Mark Stamp - San Jose State University - October 2015*

[3] Solving the False positives problem in fraud prediction using automated feature engineering - *Wedge, Canter, Rubio et.al - October 2017*

[4] PayPal Inc. Quarterly results *https://www.paypal.com/stories/us/paypal-reports-third-quarter-2018-results*

[5] A Model for Rule Based Fraud Detection in Telecommunications - *Rajani, Padmavathamma - IJERT - 2012*

[6] HTTP Attack detection using $n-$gram analysis - *A. Oza, R.Low, M.Stamp - Computers and Security Journal - September 2014*
[7] Scikit learn - machine learning library *http://scikit-learn.org*
[8] Paysim - Synthetic Financial Datasets For Fraud Detection *https://www.kaggle.com/ntnu-testimon/paysim1*