# Capstone 1 Project Report

04/21/2018

Ginny Jie Zhu

## Problem description

With ecommerce and online transactions increasing each day, credit card fraud has become a real concern for consumers as well as financial companies. According to Forbes news(1), card fraud losses actually hit $21.84 billion worldwide in 2015, which was truly alarming.

So the question I am going solve here is: how can I maximize the use of transaction data and help accurately detect a potential fraudulent transaction?

## Clients

The real-world clients could be financial companies like Chase, Capital One, American Express, Discover etc.

- For company themselves, the early prediction of potential frauds will help prevent unnecessary financial losses by allowing them to take appropriate early actions.
- For customers, the detection features could be built into credit card apps so that when a potential fraud is detected, the customers could be alerted of the situation by texts or calls. They could then respond and communicate with card companies to solve the issues together.

After all, who doesn't like a reliable model/method that helps guard your bank accounts and prevent unnecessary financial losses?

## Data

The dataset is already available on Kaggle.
([https://www.kaggle.com/dalpozz/creditcardfraud](https://www.kaggle.com/dalpozz/creditcardfraud))

It contains transactions made by credit cards in September 2013 by European cardholders. The transactions all happened within 2 days, where there are 492 frauds out of 284, 807 transactions.(2) It's in the format of .csv tabular data which has the following information:

- V1-V28: The numerical principal components obtained through PCA transformation. Original features and background information are not provided due to confidentiality issues.
- Time: The seconds elapsed between each transaction and the first transaction in the dataset;
- Amount: Transaction amount;
- Class: The classification class. 1: fraud; 0: legal transaction.

Here is what the head of the data looks like:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | |

| | V8 | V9 | ... | V21 | V22 | V23 | V24 | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.098698 | 0.363787 | ... | -0.018307 | 0.277838 | -0.110474 | 0.066928 | |
| 1 | 0.085102 | -0.255425 | ... | -0.225775 | -0.638672 | 0.101288 | -0.339846 | |
| 2 | 0.247676 | -1.514654 | ... | 0.247998 | 0.771679 | 0.909412 | -0.689281 | |

| | V25 | V26 | V27 | V28 | Amount | Class |
|---|---|---|---|---|---|---|
| 0 | 0.128539 | -0.189115 | 0.133558 | -0.021053 | 149.62 | 0 |
| 1 | 0.167170 | 0.125895 | -0.008983 | 0.014724 | 2.69 | 0 |
| 2 | -0.327642 | -0.139097 | -0.055353 | -0.059752 | 378.66 | 0 |

## EDA

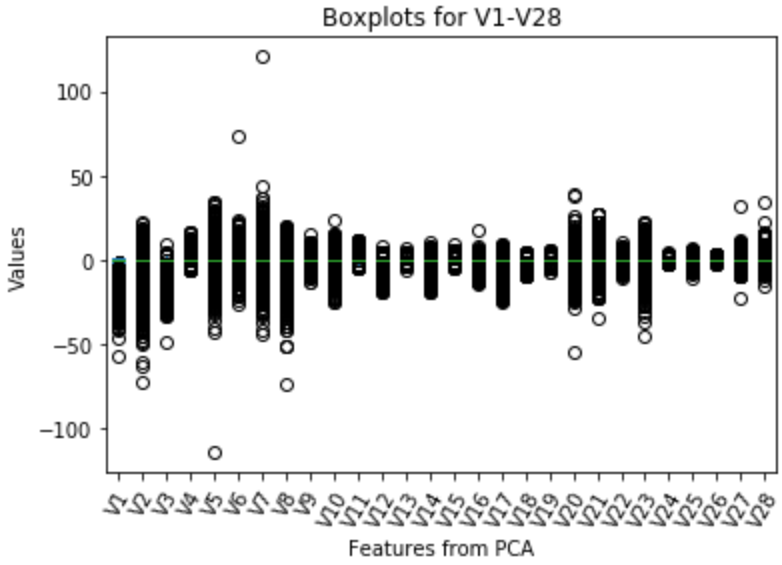
First, we do some general checks on the data:

- None of the columns have missing data, so we don't need to clean the data. For the next part, we only need to focus our attention on dealing with outliers, if any.
- It has 284807 records in total, which is consistent with the description on Kaggle. All of the features are numerical, with only 'Class' in the integer format, others being float.
- Further .shape attribute is used to confirm the dimensions of (284807, 31); and .columns attribute to confirm the column names.
- Finally, .describe(), .head() and .tail() methods are used to inspect summary stats, and general data structures of the beginning, and the end of the data frame respectively.
- Time is not unique for each entry, which means there are multiple transactions happening at the same time in the dataset.

Next we explore the overall distributions of the data as well as the distribution of data in fraud part vs legal part.

For the dataset as a whole:

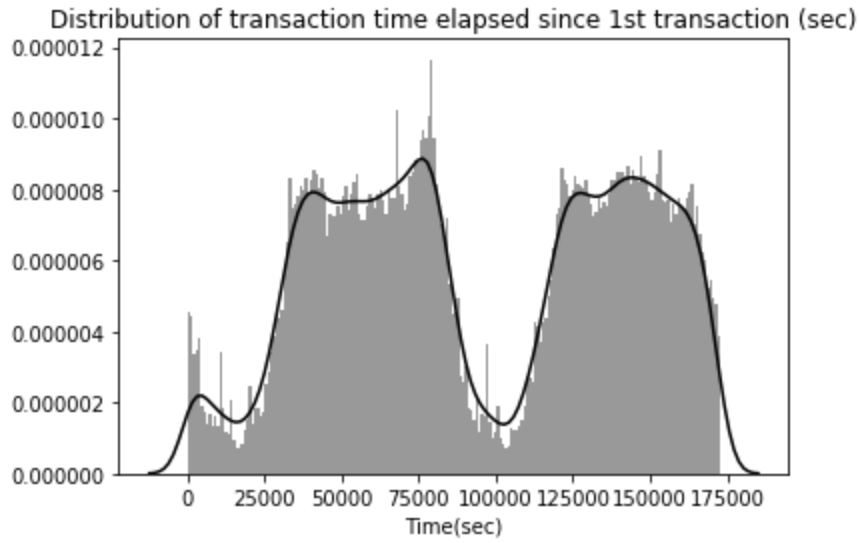
1) V1-V28




We can see from the plot that in general, these 28 features are on the same scale of within -50 to 50. Several particularly noticeable outliers are in V5, V6, V7. Considering this is the

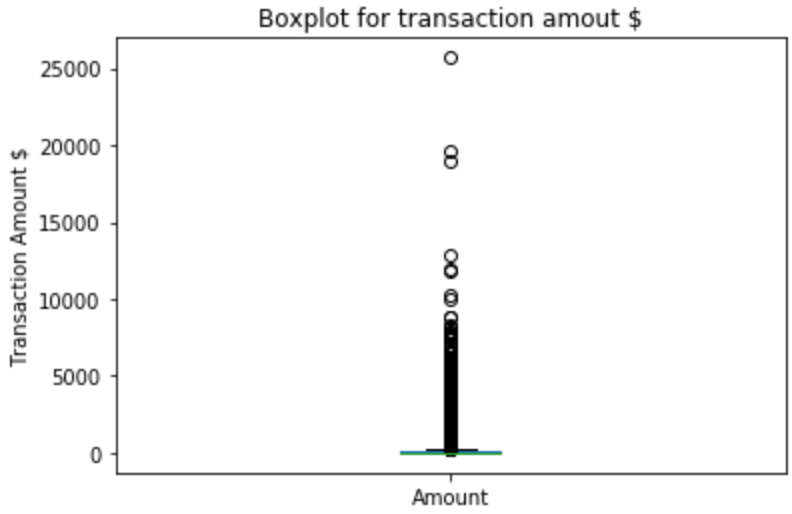
box plot for all data including legal and fraud transactions, we will examine further later in the analysis to see where these outliers are from.

2) Time




The plot shows us that there were certain time periods that transactions happened more frequently. considering this variable is expressed in seconds, we are creating a new column of datetime objects and another of hour for later analysis.

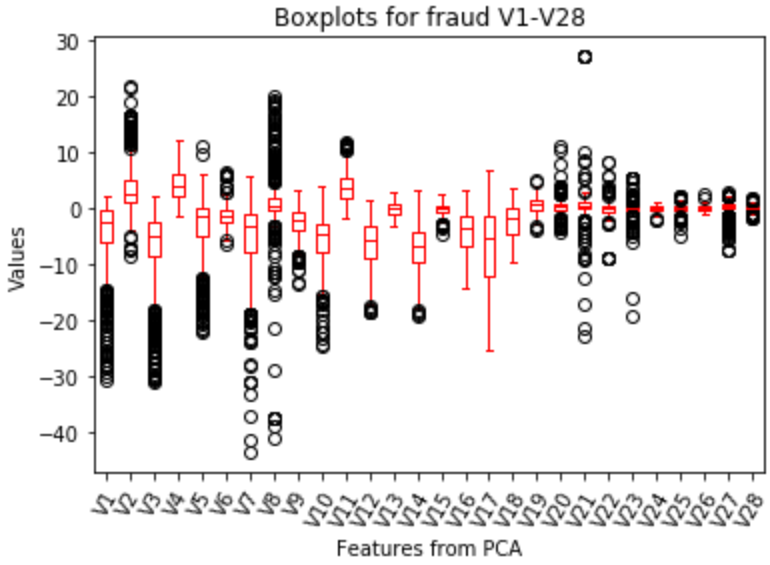
3) Amount

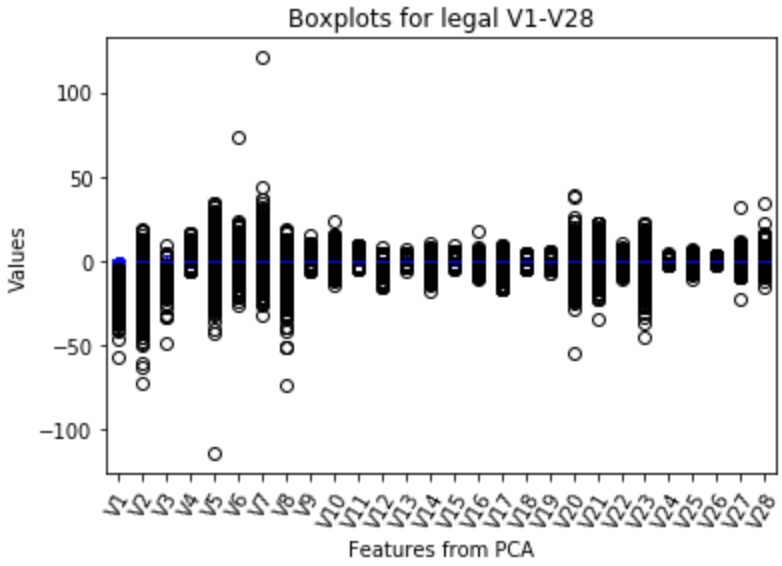



We can see there are some outliers with really big values in this feature, and we'll examine it further in the analysis.

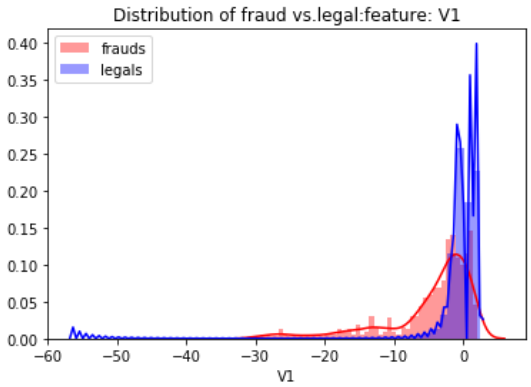
As we can calculate from the data, the fraud transactions only accounts for 0.17% of all transactions, and therefore it makes sense to investigate all the variables separately based on legal/fraud status.

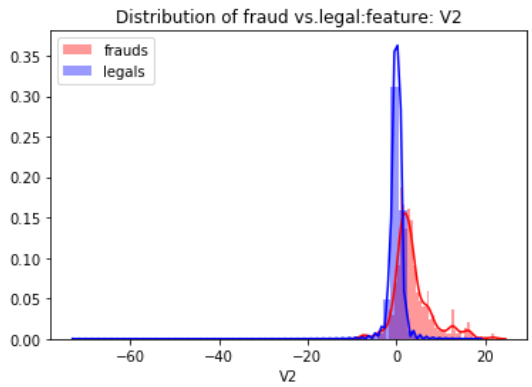
1) V1-V28

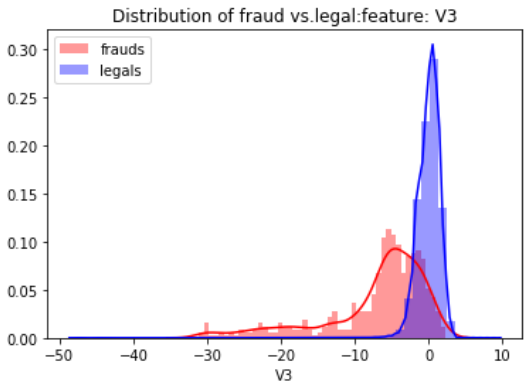


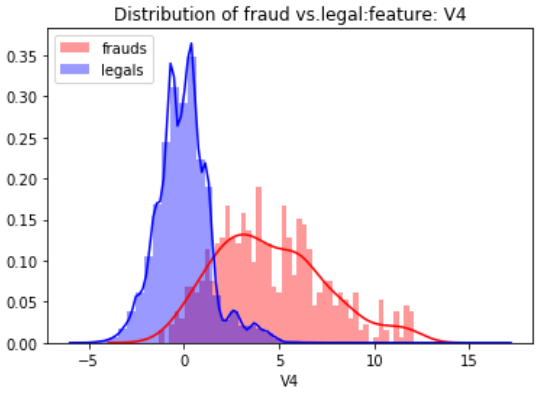



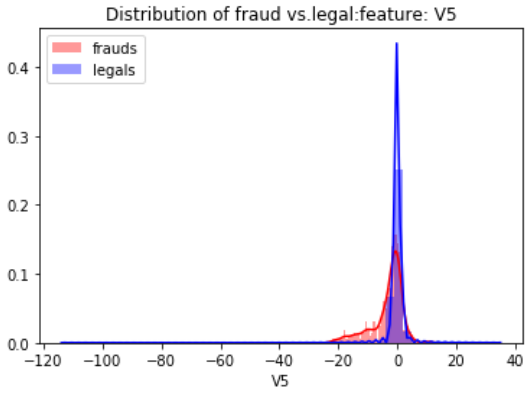

As V1-V28 of fraud transactions show relatively (but not absolutely) more variations and therefore more dispersed within itself than legal transactions. However, as we can see here, the extremely big outliers that we spotted earlier in the whole dataset(V5, V6, V7) are actually from legal transactions. Further we are plotting out the distribution of each PCA feature of legals vs. frauds.

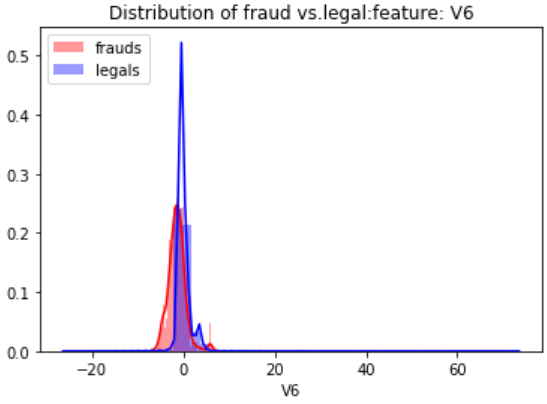

Distribution of fraud vs.legal:feature: V1
frauds
legals
V1

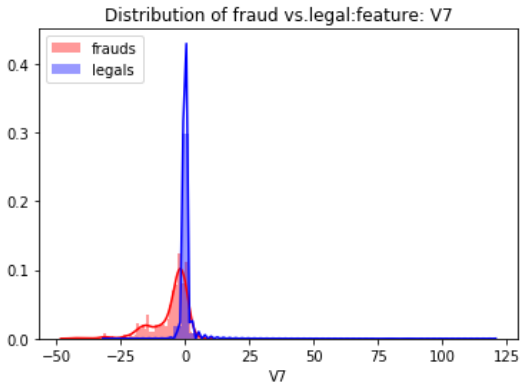

Distribution of fraud vs.legal:feature: V2
frauds
legals
V2

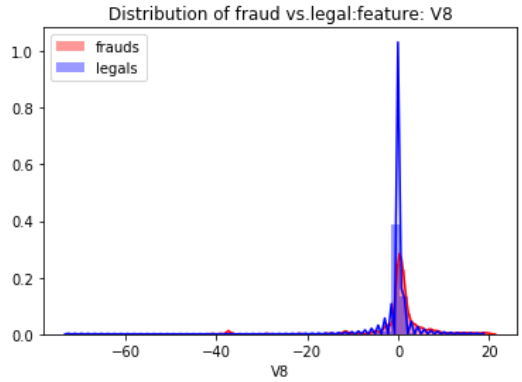

Distribution of fraud vs.legal:feature: V3
frauds
legals
V3

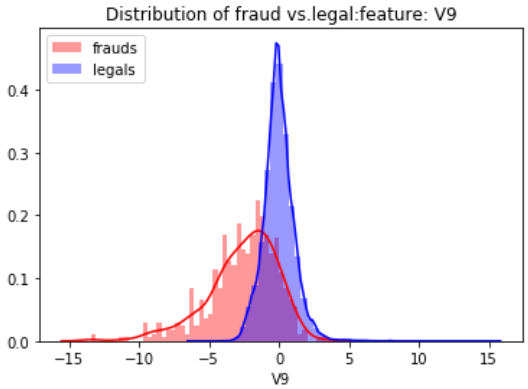

Distribution of fraud vs.legal:feature: V4
frauds
legals
V4

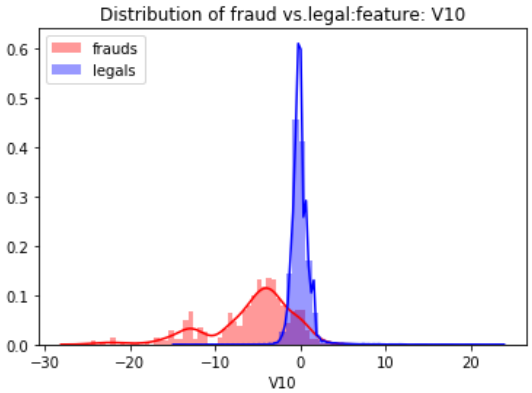

Distribution of fraud vs.legal:feature: V5
frauds
legals
V5

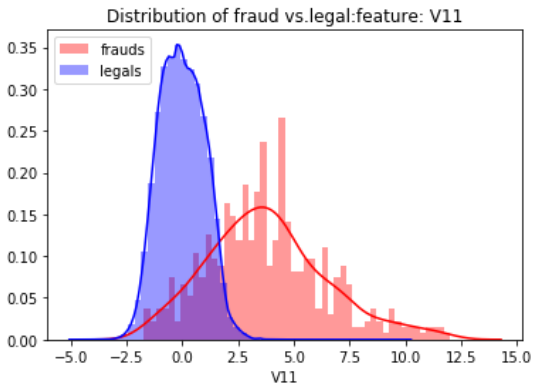

Distribution of fraud vs.legal:feature: V6
frauds
legals
V6

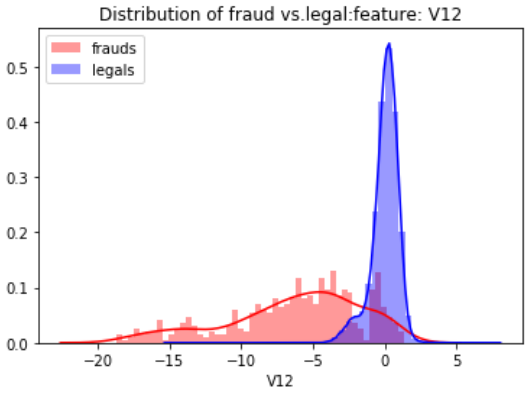

Distribution of fraud vs.legal:feature: V7
frauds
legals
V7

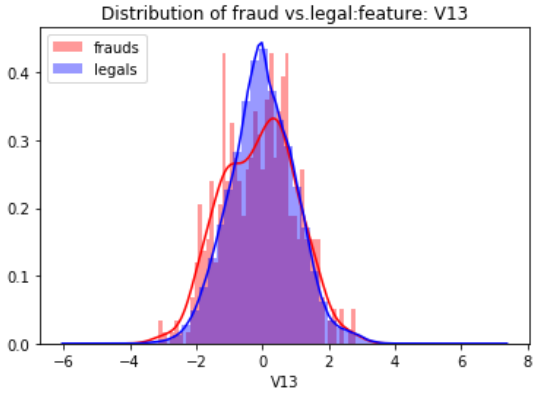

Distribution of fraud vs.legal:feature: V8
frauds
legals
V8

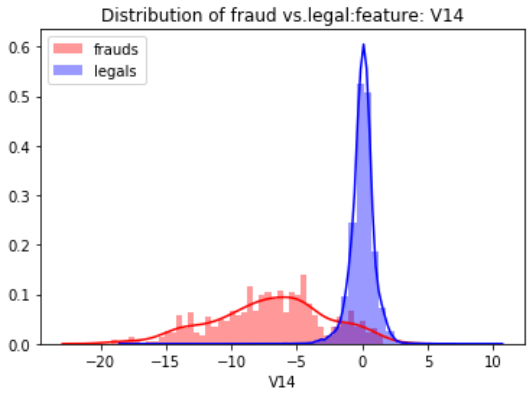

Distribution of fraud vs.legal:feature: V9
frauds
legals
V9

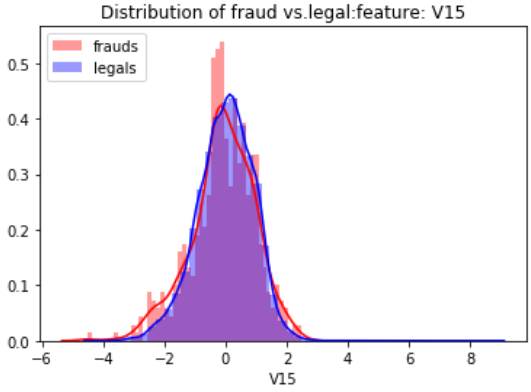

Distribution of fraud vs.legal:feature: V10
frauds
legals
V10

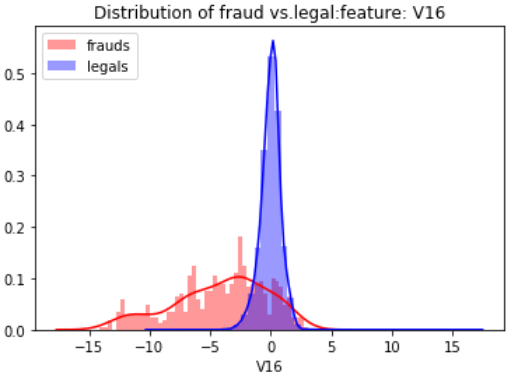

Distribution of fraud vs.legal:feature: V11
frauds
legals
V11

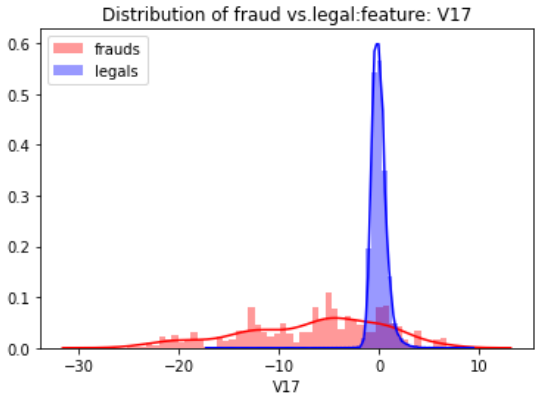

Distribution of fraud vs.legal:feature: V12
frauds
legals
V12

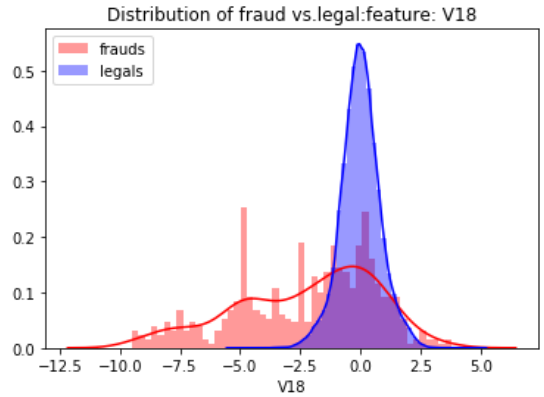

Distribution of fraud vs.legal:feature: V13
frauds
legals
V13

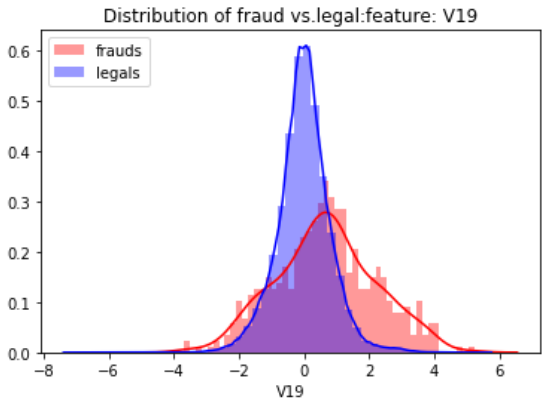

Distribution of fraud vs.legal:feature: V14
frauds
legals
V14

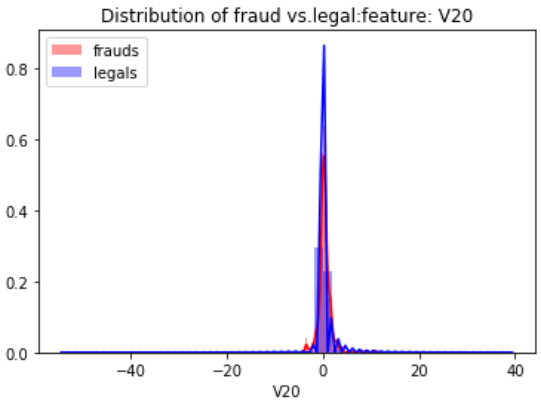

Distribution of fraud vs.legal:feature: V15
frauds
legals
V15

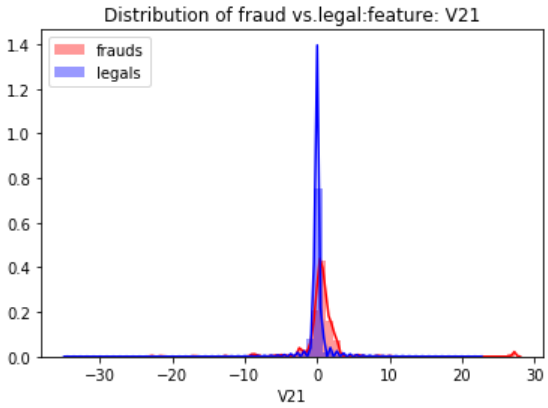

Distribution of fraud vs.legal:feature: V16
frauds
legals
V16

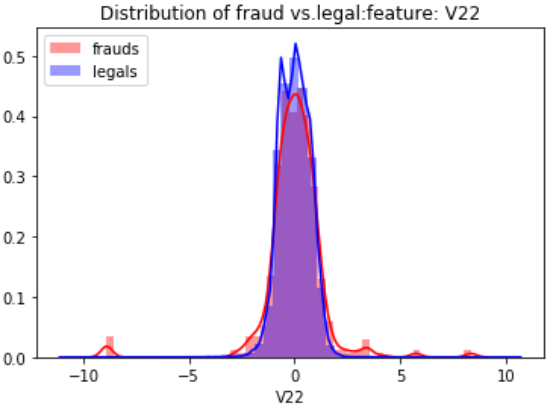

Distribution of fraud vs.legal:feature: V17
frauds
legals
V17

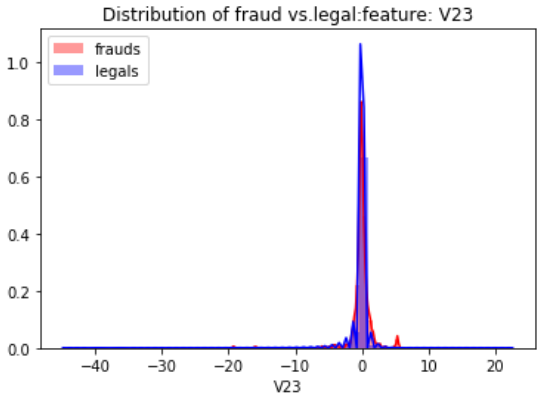

Distribution of fraud vs.legal:feature: V18
frauds
legals
V18

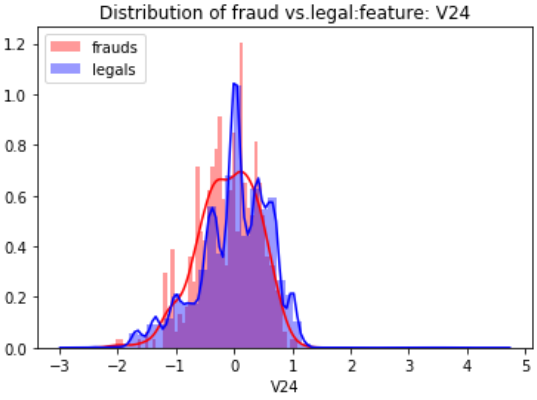

Distribution of fraud vs.legal:feature: V19
frauds
legals
V19

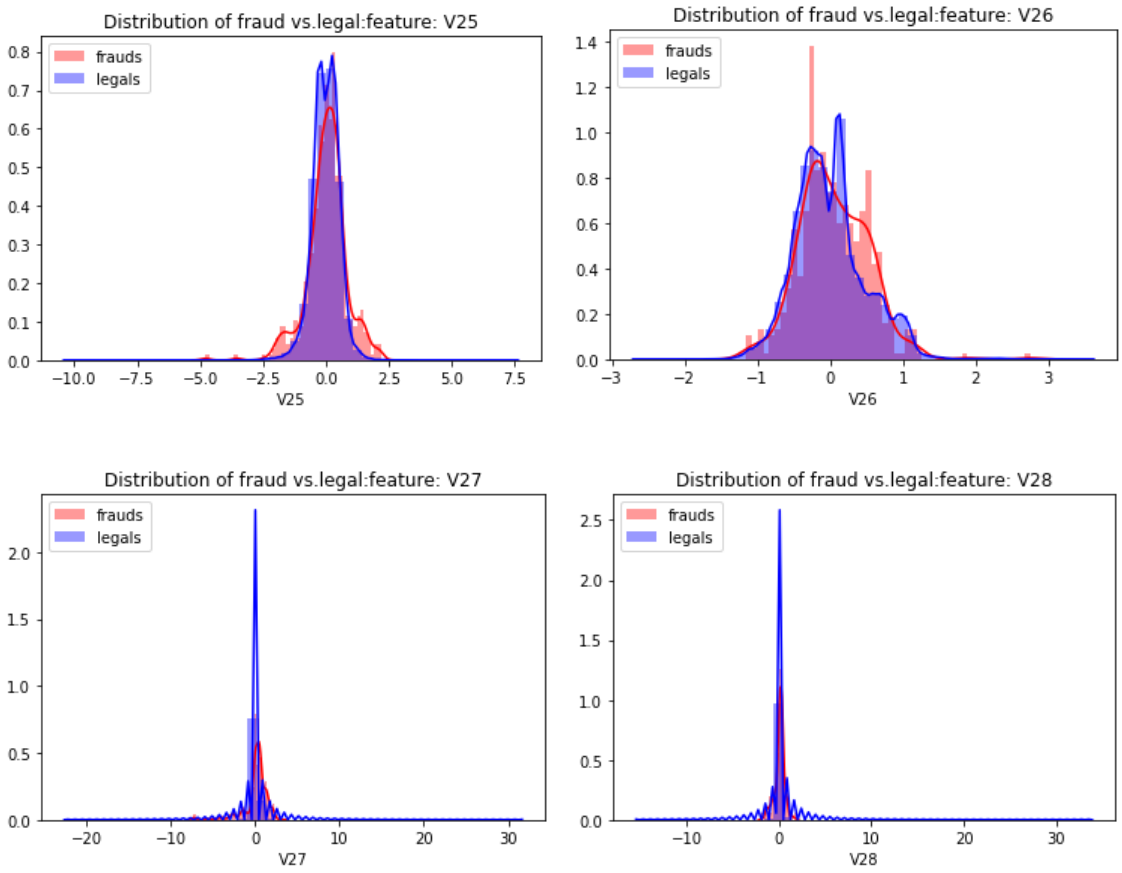

Distribution of fraud vs.legal:feature: V20
frauds
legals
V20


Distribution of fraud vs.legal:feature: V21
frauds
legals
V21


Distribution of fraud vs.legal:feature: V22
frauds
legals
V22

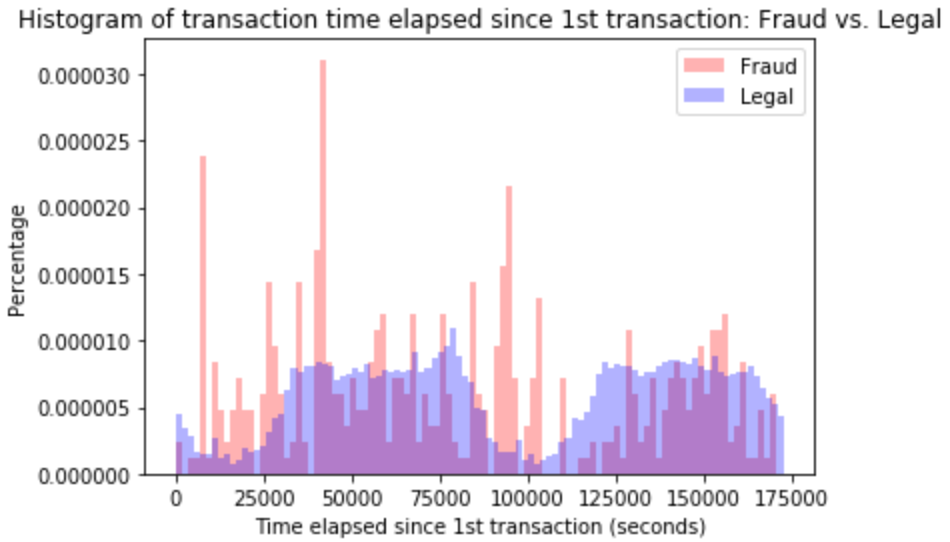

Distribution of fraud vs.legal:feature: V23
frauds
legals
V23

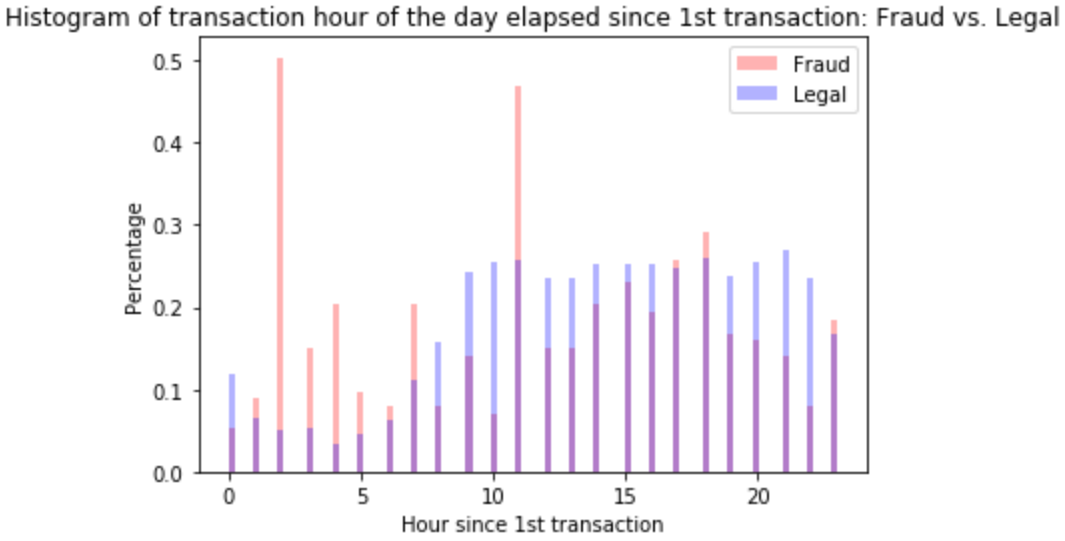

Distribution of fraud vs.legal:feature: V24
frauds
legals
V24




These plots confirm our previous conclusion that, while legal transactions spread over a larger range in general, it has relatively much narrower peak ranges.

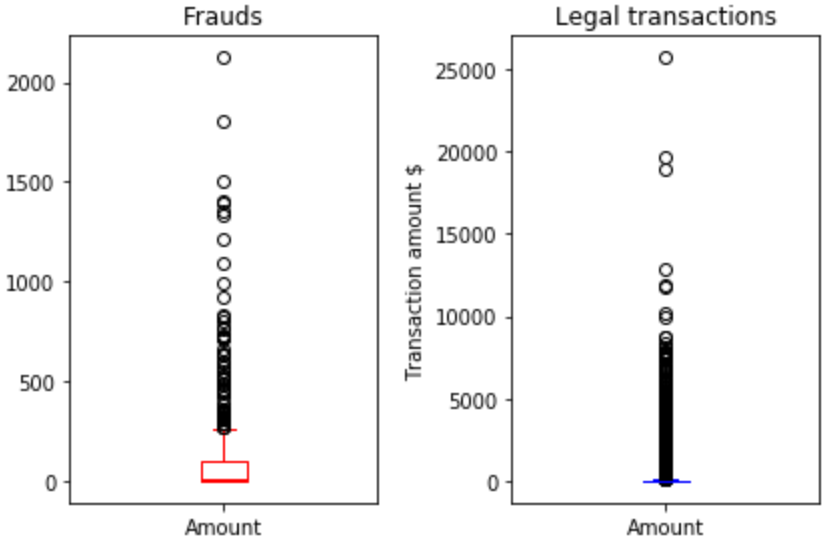
2) Time






Both of the plots tell us that fraud transactions peak when the legal ones are low. We don't have detailed time (24-hour clock) of the day of when the first transaction happened to calculate these transaction hours, but it would be logical to assume that these fraud transactions probably happen more frequently at late night and/or early morning when legal transactions are normally low.

3) Amount




Fraud transactions are all much higher than zero (which is logical), and disperse throughout around 250 to 2000. Further, even though legal transactions have some extremely large values, the fraud transactions has a higher average (122 vs. 88) as can be obtained from the summary stats.

## Inferential Statistics

Further, we carry out hypothesis testing comparing statistics between fraud and legal transactions. The specific tests will be performed are:

- z-tests for V1-V28. The null hypothesis would be h0: fraud and legal transactions have the same mean PCA.
- K-S test for Time. The null hypothesis would be h0: fraud and legal time are from the same distribution.
- Bootstrap test for Amount. The null hypothesis would be h0: fraud and legal transactions have the same mean amount.

Here are the results:

1) V1-V28

```
V1 statistically significant difference
V2 statistically significant difference
V3 statistically significant difference
V4 statistically significant difference
V5 statistically significant difference
V6 statistically significant difference
V7 statistically significant difference
V8 statistically significant difference
V9 statistically significant difference
V10 statistically significant difference
V11 statistically significant difference
V12 statistically significant difference
V13 statistically insignificant difference
V14 statistically significant difference
V15 statistically insignificant difference
V16 statistically significant difference
V17 statistically significant difference
V18 statistically significant difference
V19 statistically significant difference
V20 statistically significant difference
V21 statistically significant difference
V22 statistically insignificant difference
V23 statistically insignificant difference
V24 statistically significant difference
V25 statistically insignificant difference
V26 statistically insignificant difference
V27 statistically significant difference
V28 statistically significant difference
```

2) Time/hour

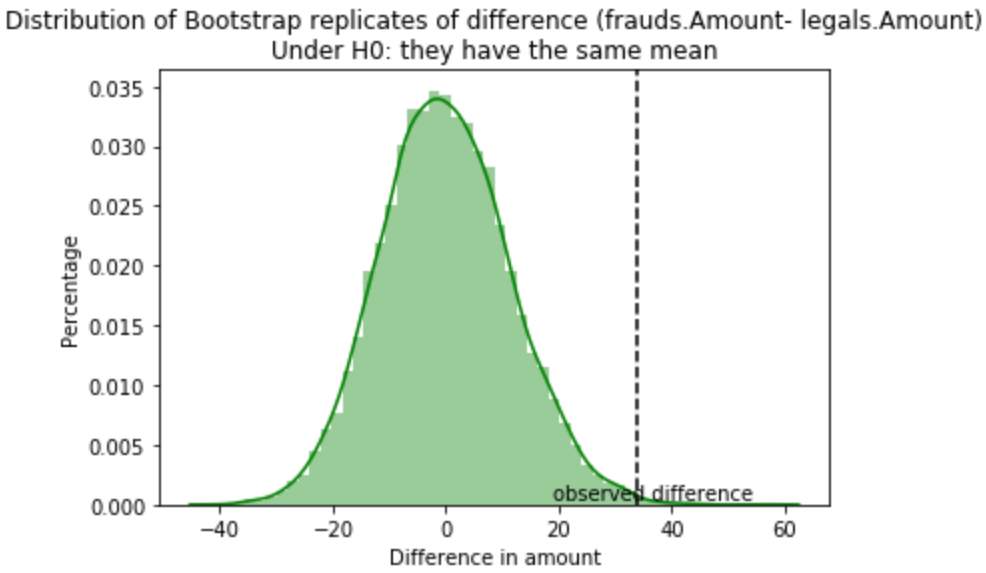
K-S statistics for Time: 0.16938909937434854 p-value: 8.357813751103828e-13
K-S statistics for hour: 0.1935411727716982 p-value: 1.3815502491489748e-16

For both (original) Time and hour, the p-values are small enough to reject the null hypothesis that the frauds and legals have the same distribution. The results are consistent with what we've obtained earlier using visual EDA.

3) Amount

We then carry out a bootstrap test for amount as well to see if the difference between frauds mean and legals mean is purely due to chance. Therefore our null hypothesis h0: the mean of fraud amount = legal amount.

p-value = 0.002100




As we can see here, the difference in means of fraud transactions and legal transactions is highly unlikely due to chance, as p-value for getting a observed difference of ~33.92 or more extreme is 0.000000.

Summary:

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. Highly unbalanced, frauds (class ==1) only account for 0.172% of all transactions.

- No missing value is present as concluded from EDA
- V1-V28 were preprocessed by PCA to maintain confidentiality of sensitive information and therfore PCA features outliers, if any, are not completely meaningless. For the outliers in the amount, we see that outliers almost exclusively come from legal transactions. Once in dispute, these large transactions could be of great concern to companies, and therefore it's important to keep this information in the model.
- V1-V28, time(hour) and amount in general, all demonstrate different stats properties (mean, std, distribution etc.) in fraud and legal transactions, so it would be reasonable to include them in the prediction model.

## Predictive Modeling

Now onto building the model.

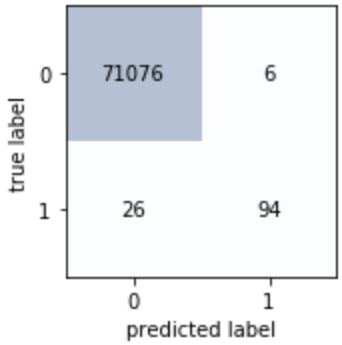
The models we're going build:

1) SVM with tuned hyperparameters
2) SVM with only statistically significant different features
3) SVM with SMOTE upsampling on frauds
4) Random Forest with tuned hyperparameters
5) A 3-layer ANN model

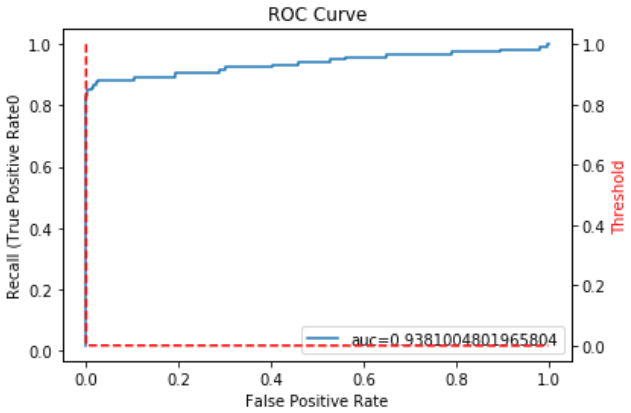
We use confusion matrix, precision, recall, AUC as metrics for model performance, and we also plot AUC curve and precision-recall curve.

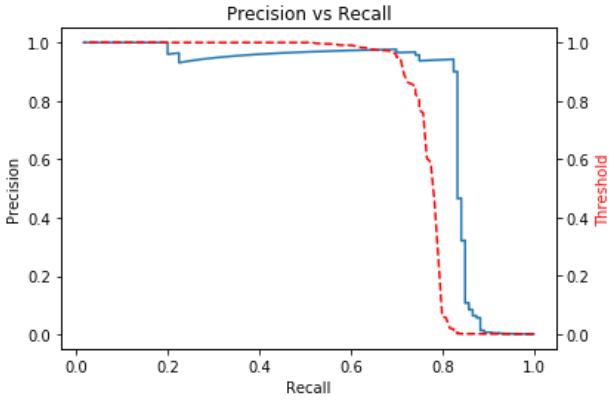
The confusion matrix gives us the predicted class results, and the classification report gives us the result metrics weighted by support (the number of true instances for each label). Recall measures: how many true positives (fraud) are actually predicted as positives. Precision measures: how many predicted positives are actually positives(fraud). So for a fraud detection model, our principle should be to detect all possible frauds (highest possible recall) and at the same time reasonably high proportion of frauds among those predicted as frauds.The ROC and Precision-Recall curve gives us a direct view of how those metrics change with different threshold settings.(3)(4)(5) So in general, we want to set a threshold that result in best possible recall and precision. For ROC curve, the optimal point should be upper left corner (high recall, and low False positive rate); for Precision-Recall curve, the optimal point should be upper right corner(high recall and high precision).

1) SVM with tuned parameters using grid search

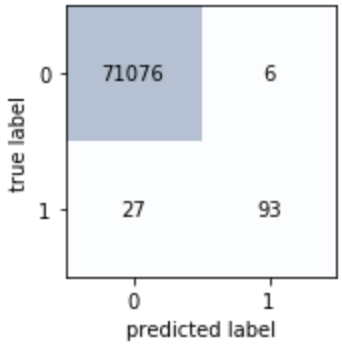



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 71082 |
| 1 | 0.94 | 0.78 | 0.85 | 120 |
| avg / total | 1.00 | 1.00 | 1.00 | 71202 |

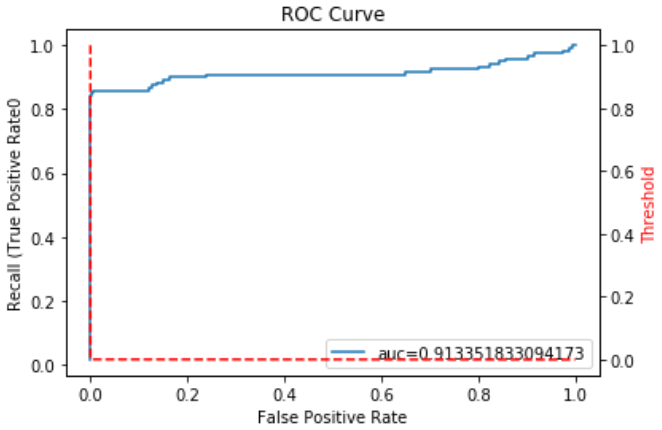


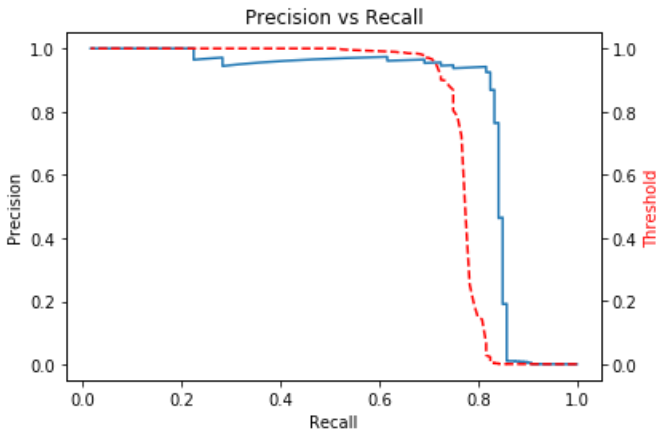



2) SVM with only statistically significantly different features

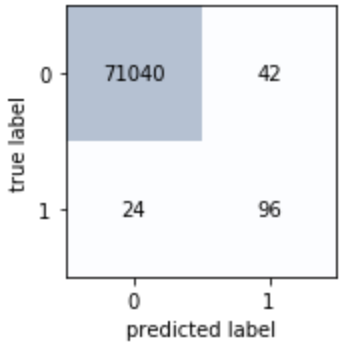



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 71082 |
| 1 | 0.94 | 0.78 | 0.85 | 120 |
| avg / total | 1.00 | 1.00 | 1.00 | 71202 |

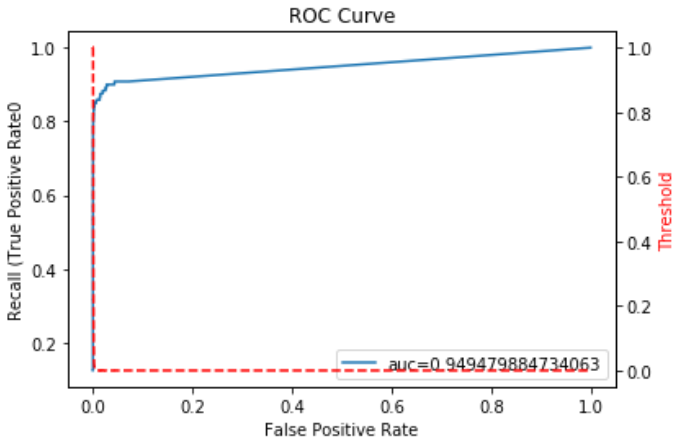


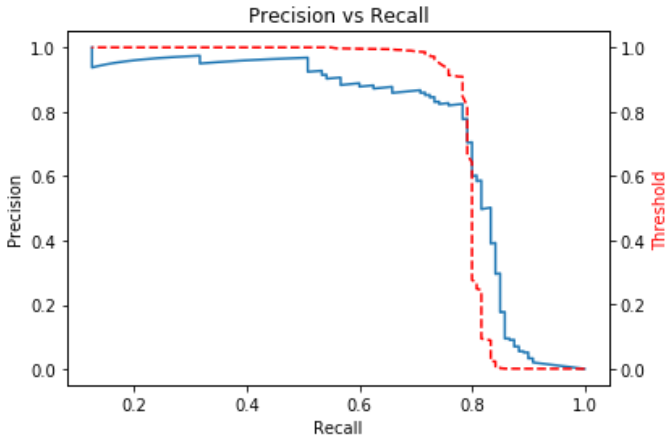



3) SVM with SMOTE upsampling on frauds

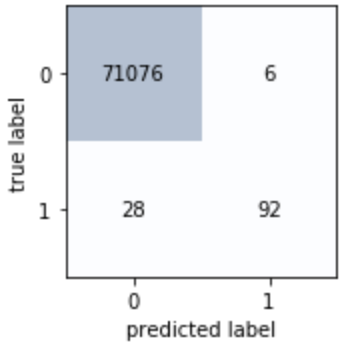



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 71082 |
| 1 | 0.70 | 0.80 | 0.74 | 120 |
| avg / total | 1.00 | 1.00 | 1.00 | 71202 |

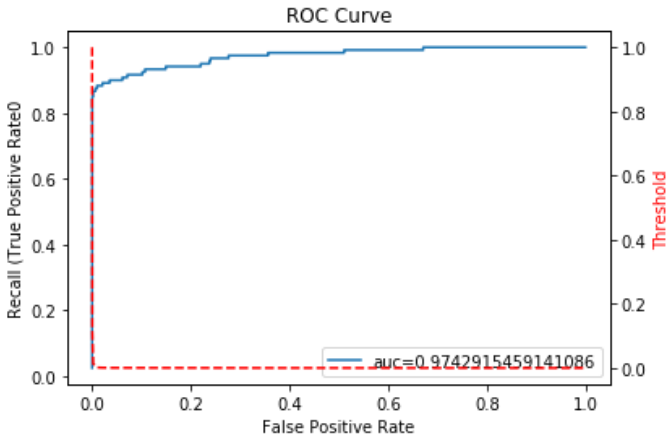


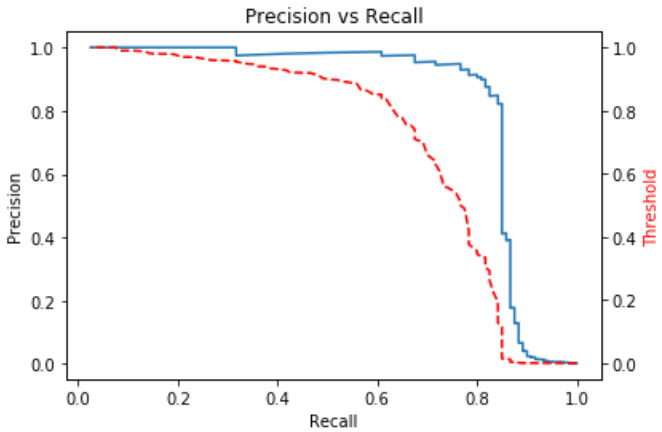



4) Random Forest with tuned parameters




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 71082 |
| 1 | 0.94 | 0.77 | 0.84 | 120 |
| avg / total | 1.00 | 1.00 | 1.00 | 71202 |

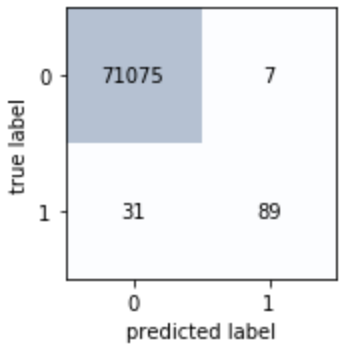


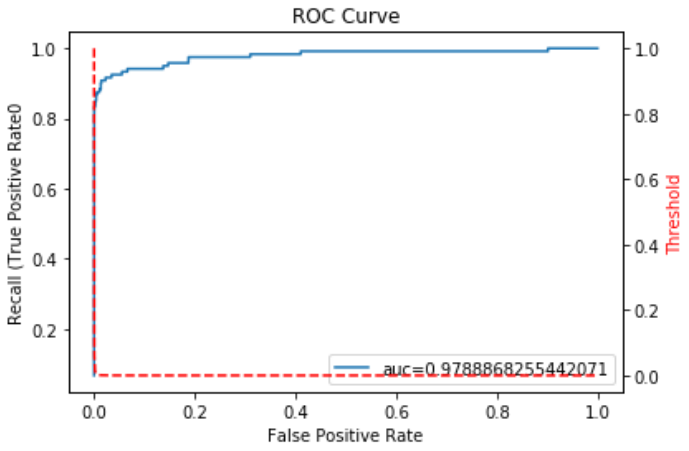
5) 3-layer ANN model

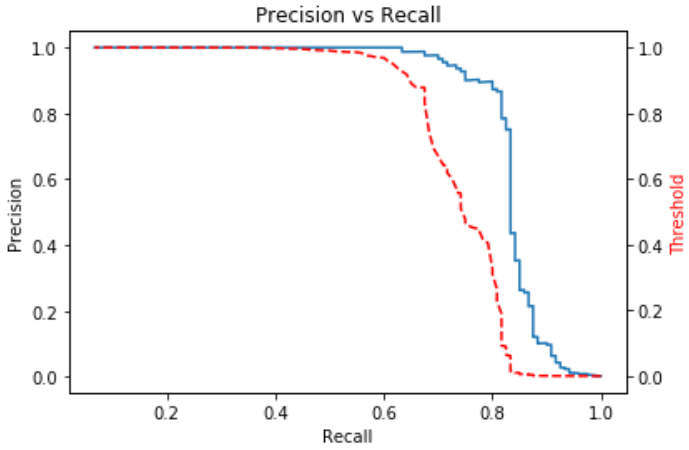
We build the model with Keras with the following architecture, using 'binary_crossentropy' as the loss function, and 'adam' optimizer, and early stopping if the model doesn't improve after 3 epoches.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               3100
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 202
=================================================================
Total params: 13,402
Trainable params: 13,402
Non-trainable params: 0
```




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 71082 |
| 1 | 0.93 | 0.74 | 0.82 | 120 |
| avg / total | 1.00 | 1.00 | 1.00 | 71202 |




All important results are summarized as the table below:

| **Model** | **Precision (for positive)** | **Recall (for positive)** | **Test AUC** |
|---|---|---|---|
| 1.SVM with tuned parameters | 0.94 | 0.78 | 0.94 |
| 2. SVM with only selected features | 0.94 | 0.78 | 0.91 |
| 3.SVM with SMOTE Upsampling | 0.70 | 0.80 | 0.95 |
| 4.Random Forest with tuned parameters | 0.94 | 0.77 | 0.97 |
| 5. ANN | 0.93 | 0.74 | 0.98 |

With a simple NN model, after only around 2 epochs, it achieves support weighted recall of 0.74 and precision of 0.93, with comparable ROC and Precision-Recall curve with tuned random forest and fast training time.

Overall, in terms of best results, tuned SVM outperforms others. But in terms of practical considerations of efficiency of training the model with more real-life data, random forests and nn model are similar in performances, so they are both apt for the detection task.

However if only one choice should be made, then I would suggest random forest as the top pick for its slightly better recall (for in real life situations we would want to capture as many frauds as possible to reduce the risk of potential financial loss, so the slight higher rate(0.3 higher) could mean a practical difference) and precision, and also ease of implementation as compared to the nn model.

Finally, we can incorporate the model into the banking system to help better detect and alert customers as well as companies of possible frauds.

## References


(1) https://www.forbes.com/sites/rogeraitken/2016/10/26/us-card-fraud-losses-could-exceed-12bn-by-2020/#53aa8baed243

(2) K.R., Seeja & Zareapoor, Masoumeh. (2014). FraudMiner: A Novel Credit Card Fraud Detection Model Based on Frequent Itemset Mining. TheScientificWorldJournal. 2014. 252797. 10.1155/2014/252797

(3) https://classeval.wordpress.com/introduction/introduction-to-the-precision-recall-plot/

(4) Receiver Operating Characteristic, https://en.wikipedia.org/wiki/Receiver_operating_characteristic

(5) Area under curve, https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve