MOBILE TRANSACTION FRAUD DETECTION WITH MACHINE LEARNING

3253 - Machine Learning
University of Toronto

Prepared By: EHSANUL HAQUE

4/19/2019

[This report is a requirement for the final project of 3253-Machine Learning course. This report analyzes the performance of various Machine Learning classifiers on a synthetic mobile transaction data set and selects the best model to detect fraud transactions]

Problem description:

With the advancement of technology, monetary transactions between customers and merchants have been revolutionized. Both customers and merchants tend to go for transactions that are convenient, quick and secure at the same time. For this need for reliable transactions, quite a few technologies have been utilized. The mobile transaction is one of the most popular methods these days and will continue to grow in future.

Mobile transactions are performed from a mobile device. It allows users to pay for a variety of service or goods. Since this process does not involve cash, cheque or credit cards, and almost everyone has access to a mobile device, investment on mobile payment services is expected to grow throughout the world. With the widespread use of mobile transactions, it has also become a target for the fraudsters to commit fraudulent activities. To get rid of mobile transaction frauds, machine learning can be an effective tool to combat the fraudsters.

Data description:

The dataset used for this project is a synthetic data of mobile money transactions. This data is produced by a simulator called Paysim. PaySim uses data from the private dataset to generate a synthetic dataset that resembles the normal operation of transactions and injects malicious behaviour to later evaluate the performance of fraud detection methods.

Following is screenshot and description of the features of the dataset:

```
In [3]: # data row and column present is data set

        data.shape

Out[3]: (6362620, 11)

In [94]: data.head(10)

Out[94]:
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
3526993	259	CASH_OUT	89497.31	C1850478992	68039.07	0.00	C903434885	775605.94	865103.25	0	0
5406486	377	CASH_OUT	53397.25	C2001426032	21417.00	0.00	C1110643880	0.00	53397.25	0	0
1539057	154	CASH_OUT	156453.28	C1740123408	0.00	0.00	C110783307	206713.53	363166.82	0	0
3161323	237	TRANSFER	124106.92	C1323564635	0.00	0.00	C826077736	229902.82	354009.74	0	0
1730862	161	CASH_OUT	324131.59	C1880594736	124177.00	0.00	C1704782973	70233.11	394364.70	0	0
3797545	281	CASH_OUT	199136.38	C887802259	0.00	0.00	C845622255	764122.11	963258.49	0	0
3477417	258	CASH_OUT	15642.23	C1598783730	20204.00	4561.77	C1614960524	1345032.88	1360675.11	0	0
6152896	547	TRANSFER	266174.03	C1355645544	0.00	0.00	C1829891106	7797770.28	8063944.31	0	0
2262581	187	TRANSFER	780958.98	C1767879737	0.00	0.00	C78867731	6899640.79	7680599.77	0	0
697840	36	CASH_OUT	48535.10	C997557013	0.00	0.00	C1056780168	1750499.40	1799034.51	0	0

This data set has 6362620 trasaction records, each with 11 features.

It has 11 columns and 6362620 rows.

This is a sample of one row with headers explanation:

Feature	Description
step	Maps a unit of time - 1 hour
type	CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER
amount	amount of the transaction in local currency
nameOrig	customer who started the transaction
oldbalanceOrg	initial balance before the transaction
newbalanceOrig	the new balance after the transaction
nameDest	customer who is the recipient of the transaction
oldbalanceDest	initial balance recipient before the transaction
newbalanceDest	new balance recipient after the transaction
isFraud	This is the transactions made by the fraudulent agents inside the simulation
isFlaggedFraud	Flags illegal attempts

Exploratory data analysis:

We ask following questions that will help us develop a strategy to clean the dataset and later apply machine learning model on the data. To verify these questions we ran a few tests on the dataset:

1. **Are there any missing or NULL data?**

If there is any missing or NULL data, we have to replace with the mean value. Our test shows that there is no such missing or NULL value.

2. **Which transactions are more vulnerable to fraud?**

There are five types of transactions in the dataset: CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER. We have to find if any of the above types are more vulnerable to the fraud. Our test shows that all fraud happens with CASH OUT and TRANSFER.

Among all the transactions, CASH OUT has 4116 and TRANSFER has 4097 fraud transactions.

3. **Does 'isFlaggedFraud' detect fraud transactions correctly?**

The feature ‘isFlaggedFraud’ is set to ‘1’ if the mobile transaction system thinks that there is a potential fraud transaction. If ‘isFlaggedFraud’ is set to ‘1’, that transaction will be cancelled right away. We have to check how correctly this feature is able to detect fraud transactions. We compared with ‘isFraud’ column to get the number of correct detections and incorrect detections.

We separated all the transactions with fraud cases and compare ‘isFlaggedFraud' and ‘isFraud' columns. The test shows that only 16 cases were detected correctly by ‘isFlaggedFraud' out of 8213 cases.

From the above test, we find that ‘isFlaggedFraud' feature does not produce a reliable result.

4. **How many fraud transactions are in the data set?**

This test shows the percentage of fraud cases among all the transactions.

In the dataset, fraud transactions are only .12908 % of all the transactions.

5. **Are the values of old balance and new balance after transactions correct?**

Dataset has information for both original balance and destination balance. When money is transferred from the original balance, the 'newbalanceOrig' will be less than 'oldbalanceOrg'. Similarly, when money is received in the destination balance, 'newbalanceDest' will be more than 'oldbalanceDest'**.** And, the difference between both the accounts is the 'amount**'** transferred.

Our test confirms that there is a mismatch in new and old balances in 5776 transactions. In this test, we have considered only CASH OUT and TRANSFER transactions as they have all the fraud cases.

Data cleaning and feature engineering:

Depending on the above exploratory tests, we perform the following tasks:

1. Keeping only TRANSFER and CASH OUT transaction in feature Metrics:

Since all the frauds are present is CASH OUT and TRANSFER transactions, we discard other transaction types from our dataset. This reduces our dataset to 2762196 rows. But still, the fraud percentage is only .2964544 % of the total data.

```
# Data cleaning: Keeping only TRANSFER and CASH_OUT transactions as they have all the frauds

randomState = 42
np.random.seed(randomState)
data = data.loc[(data.type == 'TRANSFER') | (data.type == 'CASH_OUT')]
X_fraud = data.loc[(data.isFraud == 1)]
X_not_fraud = data.loc[(data.isFraud == 0)]
```

```
len(X_fraud), len(X_not_fraud)
```

```
(8213, 2762196)
```

With TRANSFER and CASH OUT transactions, there are 8213 fraud transactions and 2762196 good transactions.

.2964544 % of total data

2. Binary encoding to TRANSFER and CASH OUT:

Since TRANSFER and CASH OUT are category variable, we convert them to ‘0' and ‘1'.

```
# Binary-encoding of labelled data in 'type'

X.loc[X.type == 'TRANSFER', 'type'] = 0
X.loc[X.type == 'CASH_OUT', 'type'] = 1
X.type = X.type.astype(int)          # convert dtype('O') to dtype(int)
```

Applied binary encoding to TRANSFER = 0 and CASH OUT = 1

3. Adding new features to address old balance and new balance errors:

Two new features 'errorBalanceOrig' and 'errorBalanceDest' are added.

For original balance, Error = New balance + Amount – Old balance

For destination balance, Error = Old balance + Amount – New balance

```
# Two new features added- original and destination account errors

X['errorBalanceOrig'] = X.newbalanceOrig + X.amount - X.oldbalanceOrg # e = N + A - O
X['errorBalanceDest'] = X.oldbalanceDest + X.amount - X.newbalanceDest # e = O + A -N
```

Two new feature added- 'errorBalancedOrig' and 'errorBalancedDest'

After data cleaning, our final feature metrics consists of 9 columns and 'isFraud' is used in label metrics.

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	errorBalanceOrig	errorBalanceDest
3289983	252	1	69263.27	21438.00	0.0	1158936.13	1228199.41	47825.27	-1.000000e-02
3947219	286	1	97091.24	97091.24	0.0	88247.71	185338.95	0.00	0.000000e+00
1904324	165	1	107406.97	0.00	0.0	351826.14	459233.11	107406.97	0.000000e+00
6281780	650	1	692654.27	692654.27	0.0	0.00	692654.27	0.00	0.000000e+00
243018	14	1	63328.35	0.00	0.0	117421.11	268072.69	63328.35	-8.732323e+04
1059684	117	0	251621.18	251621.18	0.0	0.00	0.00	0.00	2.516212e+05
1030499	68	0	281743.12	281743.12	0.0	0.00	0.00	0.00	2.817431e+05
402512	18	1	43442.74	289.00	0.0	0.00	43442.74	43153.74	0.000000e+00
1133180	131	1	58326.92	10253.00	0.0	459896.97	518223.89	48073.92	-5.820766e-11
6074514	517	0	699183.61	699183.61	0.0	0.00	0.00	0.00	6.991836e+05

Since the dataset is highly imbalanced [only .296455 % are a fraud and rest 99.7 % are not fraud], we have created 3 sets of data.

1. Data set 1 consists of 8213 fraud transaction + 18,000 non fraud transactions = Total 26213 [Fraud to non Fraud ratio = 1:2 approximately]

2. Data set 2 consists of 8213 fraud transaction + 26,000 non fraud transactions = Total 34213 [Fraud to non Fraud ratio = 1 : 3 approximately]

3. Data set 3 consists of 8213 fraud transaction + 34,000 non fraud transactions = Total 42213 [Fraud to non Fraud ratio = 1 : 4 approximately]

All of these data sets are split into 80% : 20% for train and test. All non fraud transactions are randomly selected. Training and test sets are run with our machine learning models and results are observed.

Model selection:

We have decided to run following classifiers on all 3 data sets

1. SGD classifier
2. KNN classifier
3. Linear SVM classifier
4. SVC classifier with tuned up hyper parameters
5. Random forest classifier
6. Feature reduction with PCA + Random forest classifier and
7. Votin classifier

To evaluate the classifier performance, we use

Accuracy = (TP + TN) / Total

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

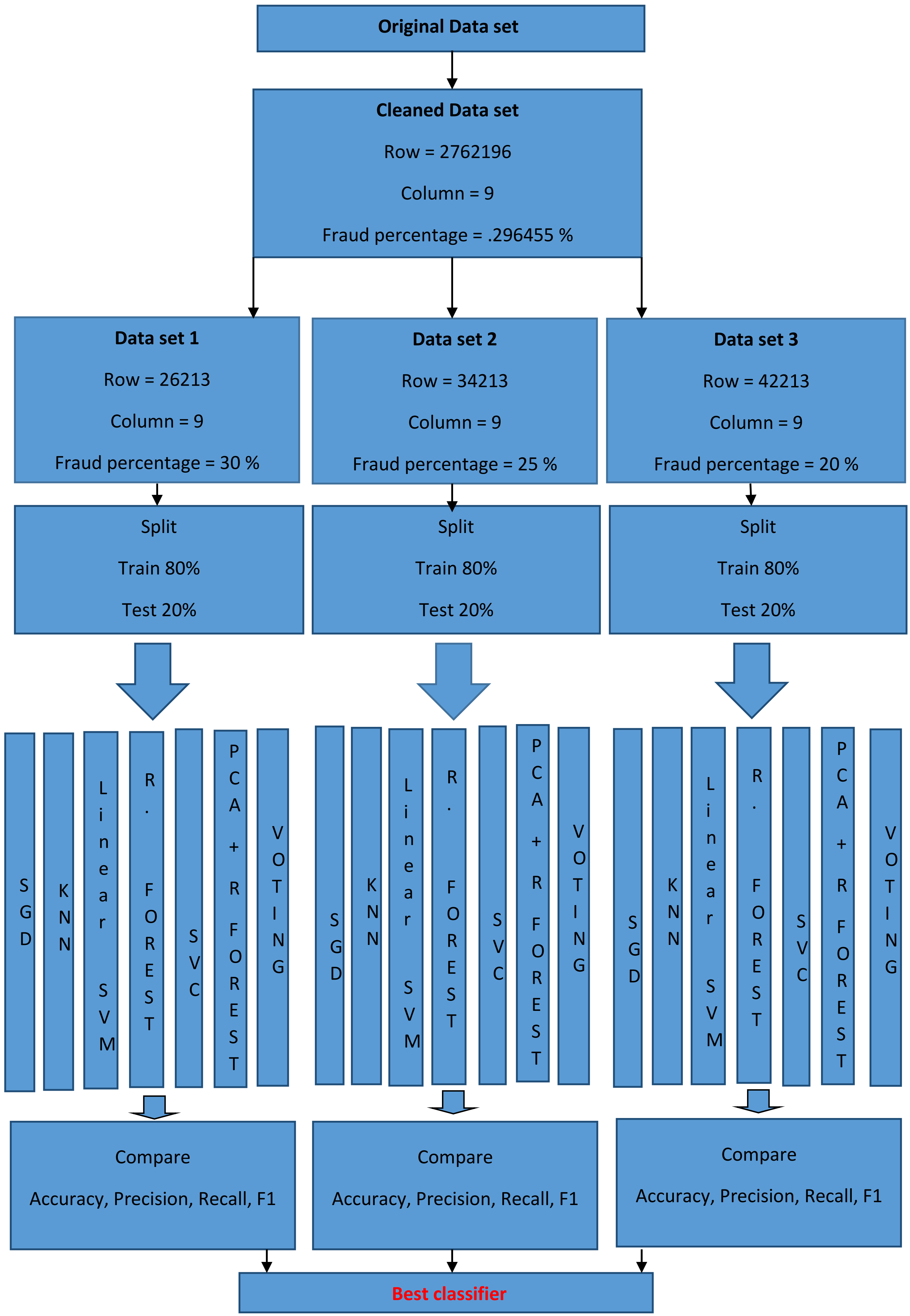
F1 score = 2*(Precision*Recall)/(Precision + Recall)

For this problem, we are interested in the Recall score to capture the most fraudulent transactions.

If Recall is increased, Precision is decreased. For this problem, if we predict that a transaction is a fraud and turns out not to be, is not a massive problem compared to the opposite.

After running 7 classifiers on 3 data sets, we will find the classifier with the best performance.

Following flow chart shows the steps to select the final model.


Original Data set
Cleaned Data set
Row = 2762196
Column = 9
Fraud percentage = .296455 %
Data set 1
Row = 26213
Column = 9
Fraud percentage = 30 %
Data set 2
Row = 34213
Column = 9
Fraud percentage = 25 %
Data set 3
Row = 42213
Column = 9
Fraud percentage = 20 %
Split
Train 80%
Test 20%
SGD
KNN
Linear SVM
R. FOREST
SVC
PCA + R FOREST
VOTING
Compare
Accuracy, Precision, Recall, F1
Best classifier

Results and comparison:

After running 7 classifiers on all 3 data sets, the best are results obtained from data set 1. Following section shows the Accuracy, Precision, Recall and F1 score of various classifiers on data set 1.

1. **SGD classifier:**

	Train	Test
Accuracy	. 87525	.87512

SGD classifier cross validation score [Accuracy, cv = 3]

	Train	Train	Train	Test
Accuracy	.93264	.92282	.86723	.93641

	Train	Test
Precision	.90372	.88517
Recall	.98210	.91578
F1	.94840	.90021

SGD classifier observations:

- Accuracy improves during cross-validation

- Accuracy and Recall of the test set are good but not very good.

2. **KNN classifier:**

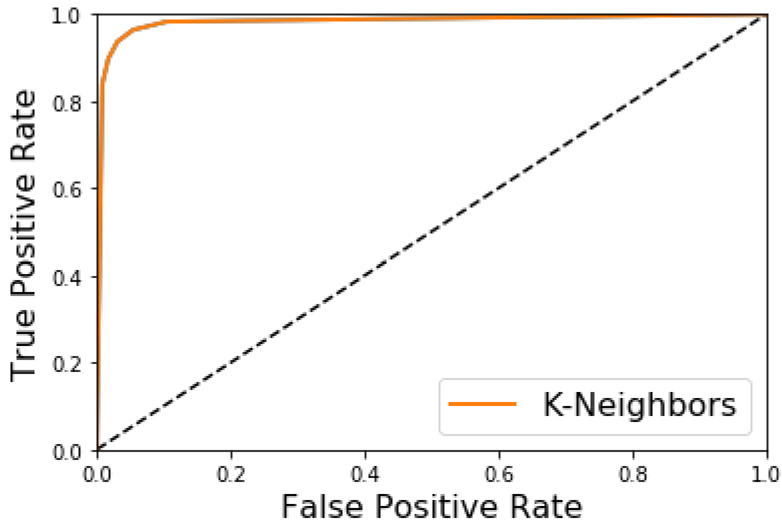
	Train	Test
Accuracy	. 97210	.96248

KNN classifier cross validation score [Accuracy, cv = 3]

	Train	Train	Train	Test
Accuracy	.95913	.95667	.95912	.95248

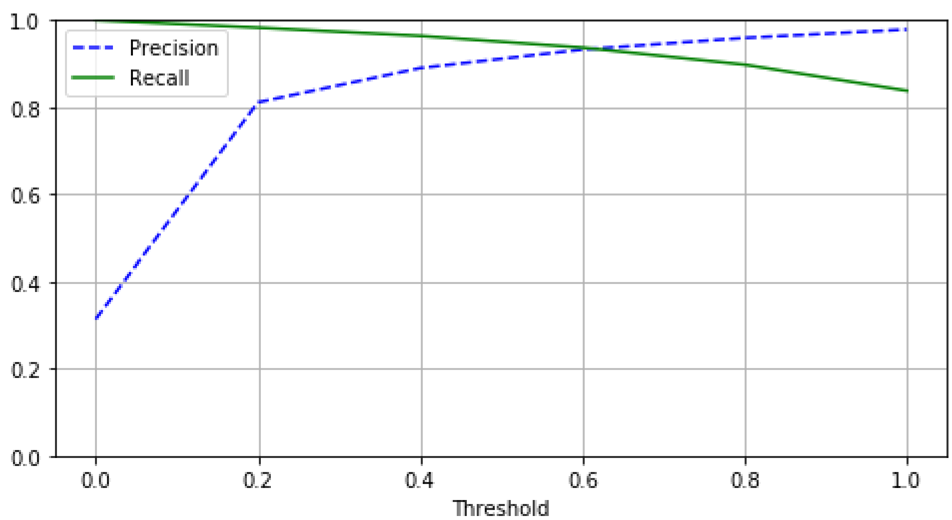
	Train
Precision	.93256
Recall	.93709
F1	.93482

ROC curve:



ROC score = .98272

Precision and Recall vs Threshold:



KNN classifier observations:

- Accuracy decreases in cross-validation

- Accuracy and Recall on the test set is good

3. **Linear SVM classifier:**

	Train	Test
Accuracy	.71660	.71553
Precision	.52558	.52376
Recall	.89044	.88064
F1	.68674	.65523

Linear SVM classifier observations:

- Both Accuracy and Recall are not good.

4. **SVC with hyperparameter tuning:**

Best Estimator: C = 489.595, gamma = .00107

	Test
Accuracy	.69010
Precision	.92102
Recall	.08213
F1	.15081

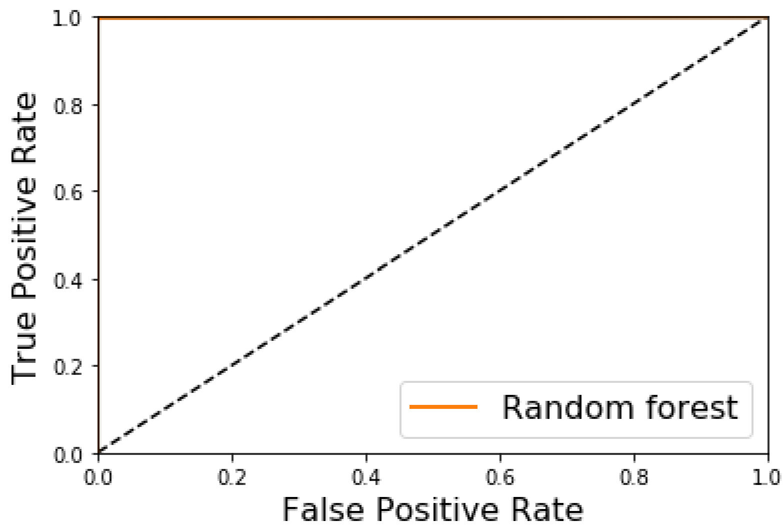
SVC with hyperparameter tuning observations:

- Both Accuracy and Recall are very poor.

5. **Random forest:**

	Train	Test
Accuracy	.99994	.99872
Precision	.98000	.99959
Recall	.99982	.99638
F1	.99913	.99796

ROC curve:



ROC score = .99312

Random forest observations:

- Both Accuracy and Recall are very good.

6. **Feature reduction with PCA + Random forest:**

	Test
Accuracy	.82744
Precision	.89237
Recall	.50935
F1	.64853

PCA _ Random forest observations:

- Accuracy is OK but Recall is very poor.

7. **Voting classifier:**

voting_clf = VotingClassifier(estimators = [('lr',log_clf),('kn',knn_clf),('rf',rnd_clf), ('svc',svm_clf)], voting = 'hard')

	Test
Accuracy	.98028
Precision	.99913
Recall	.93775
F1	.96747

Voting classifier observations:

- Both Accuracy and Recall are good.

After reviewing performances of all classifiers, it is evident that **Random forest has the best Accuracy and Recall** [on both train and test set]. So, we select the Random forest classifier as our final model.

Classifiers	Accuracy	Recall
SGD	.93641	.91578
KNN	.96248	.92611
R. Forest	**.99872**	**.99638**
SVC	.69010	.08213
Voting	.98028	.93775

Challenges and solutions:

1. Original dataset has 6.3 million rows. It takes a very long time to train all the data. We had to use 3 data sets that are much reduced in size. Using 3 different data sets also helped us to analyze classifier performance.

2. Selecting the correct feature was also a challenge. We ran quite a few tests to get rid of less important features. And, we also created 2 new features. While running PCA, it was observed that these new features have a significant impact on results.

3. The original data set is highly imbalanced. Fraud cases are only .12908% of all transactions. So, we had to create 3 data sets with a balanced amount of fraud and non-fraud transactions.

4. Using a laptop [i5 processor, 8GB RAM], running 7 classifiers on 3 different data sets was very time-consuming. To speed up the process, we also used google colab to execute some sections of the code.

Conclusion and future direction:

With the knowledge obtained from this course and assignments/project, we are confident that we can plan and execute a machine learning project with greater confidence. To make this project work more efficiently, we can implement a few things for the future:

1. We can use PCA more frequently with other classifiers to reduce features. Currently, we have used PCA only with Random forest and the result was not impressive. However, with more experiment with PCA may help us achieve better results.

2. We have to spend more time and execute more tests to find new features. Currently, we have added two new features that have a significant impact on the results. With carefully selected new features, it is possible to achieve better performance.

3. With most of the classifiers, we used default hyperparameters. In future, we have to spend more time on finding tuned parameters to obtain better results.

4. Finally, the decision metrics [to obtain the best classifier] we used is based primarily on Accuracy and Recall. For this problem, we have to find metrics with more considerations that will help us find the most suitable classifier.