**ST: BIG DATA ANALYTICS**

**(CS 696-16) (FA18)**

**Project 3**

**“Credit Card Fraud Detection”**

**Submitted By**

**Ashok Kumar Shrestha**

**#A25265227**

**Contents**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Topics** | **Page No.** |
| 1. | Introduction | 1 |
| 2. | Data Visualization | 1 |
| 3. | Methods | 2 |
| 4. | Results and Discussion | 3 |
| 5. | Conclusion | 5 |
| 6. | References | 6 |
| 7. | Appendix | 7 |

**Introduction:**

For final project, Credit Card Fraud Detection dataset[1] has been chosen. This dataset is taken from Kaggle challenge[1]. The dataset has been gathered from european customers for credit card transactions over the span of two days in September 2013. It is observed that, the dataset is highly skewed and unbalance with 0.17% of fraudulent transactions.

**Attribute Information:**

The names of features in the datasets are changed to protect confidentiality of user’s data as V1, V2,... V28 except for Time, Amount and Class features. The Class feature represents the label of the datasets: 1 - fraud transaction, 0 - otherwise. The values are obtained after applying PCA transformation on the data except the above mentioned features. There are total of 284315 samples with 492 fraudulent transactions. All the codings are done in Python on Jupyter Notebook[2] with Sklearn library support[3].

**Data Visualization:**

The dataset consists of two classes: fraudulent(1) and regular transactions(0). The sample distributions is shown in the figure-1.

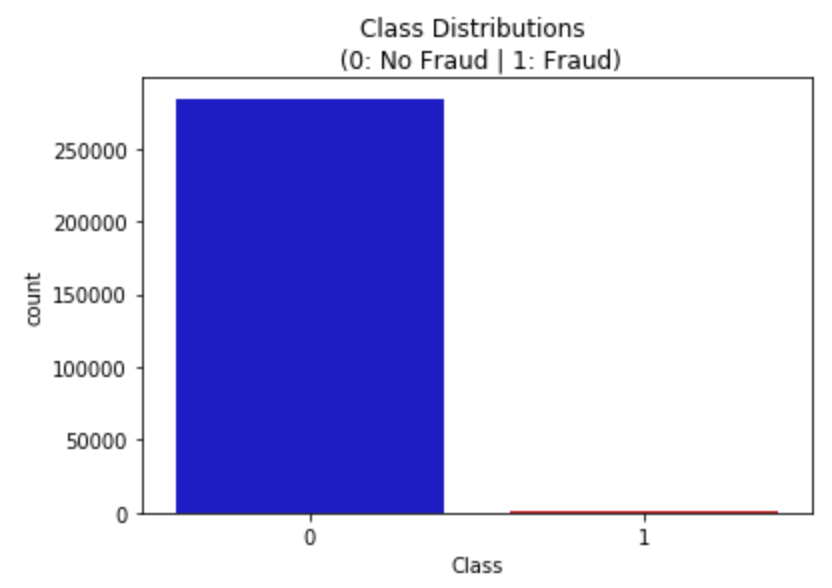


Fig-1: Class distribution for Credit card fraud detection dataset.

From the bar-graph in figure-1, it is evident that the given dataset is skewed and imbalance. If the dataset is directly used for training, it will overfit the model and predict most transactions to be non-fraudulent transactions and won’t be able to determine the pattern. There are several methods to address this issue. Some of them are listed below:

* **Collect more data:**

More data could be collected to represent proper sample distribution but this process could be costly and may not be possible or applicable in all cases such as this one.

* **Performance Metric:**

Use different performance measuring metrics to validate model designed such as:

* + Confusion matrix: measure recall and precision.
  + F1-score: weighted average of precision recall.
  + Kappa: classification accuracy obtained normalized by the skewed labels in datasets.
  + ROC curve: measure sensitivity/specificity ratio.
* **Resampling:**

Resample the dataset to have 50/50 ratio of both the classes. This can be done in two ways.

1. **Over-sampling:** Add copies of under-represented class. This will increase number of samples.
2. **Under-sampling:** Delete samples from over-represented class. The samples will be deleted in random fashion.

As shown in figure - 7, it is observed that no components have significant correlation with each other. The feature values of “Amount” and “Time” are not scaled and standardized using PCA. So, first we normalize the data using Standard-scaler. Next, the important features are identified and less important features are dropped from the dataset proceeding any further.

**Methods:**

The dataset provided is highly skewed and imbalance so it’s good idea to create subsample from provided dataset randomly to represent equal distribution of both class samples. In this case, it contains 50/50 fraud and non-fraud transactions samples which amounts to 984 samples (492 fraud and 492 non-fraud randomly chosen transactions). One concern of random under sampling is that large amount of information is lost and this might compromise the model designed. Before proceeding to random under sampling we need to separate datasets for testing model.



Fig-2: Equally distributed classes after random under sampling.

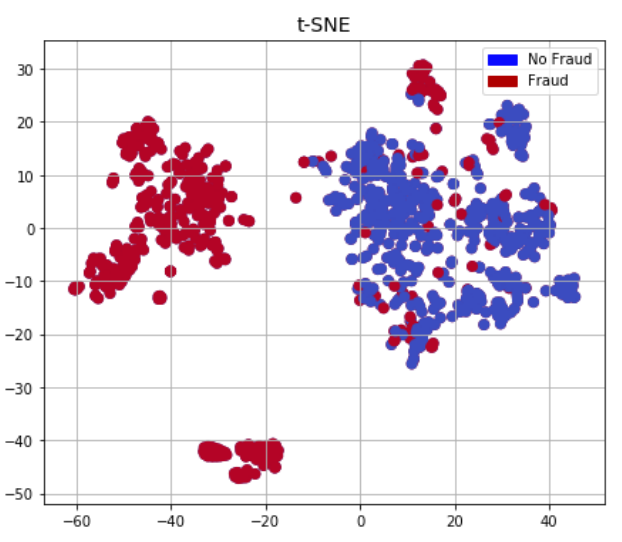
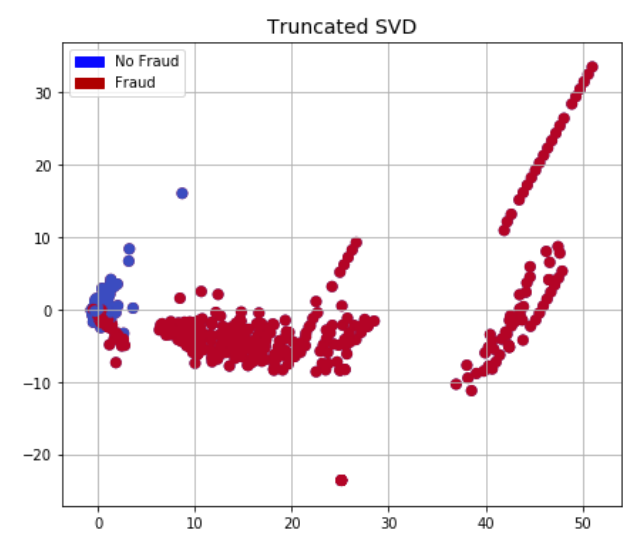
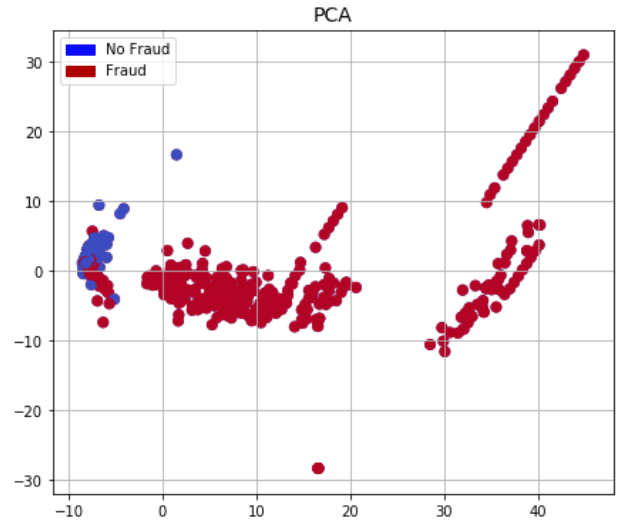
 

Fig-3: Data visualization for resampled dataset.

As shown in the figure-3, two classes have overlapping areas and cannot be easily separated. Different models were created and tested to achieve maximum class separability.

**Results and Discussion:**

The model was built with the resampled dataset and tested on the dataset. Upon testing following results were observed.

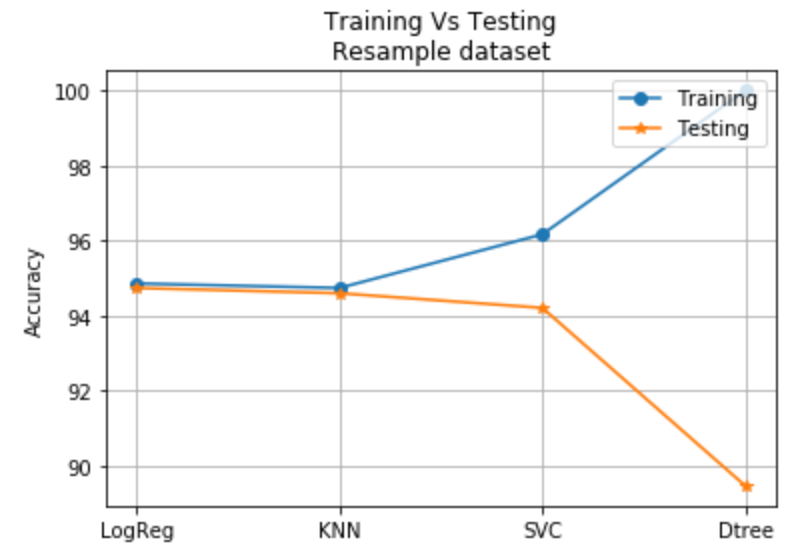


Fig-4: Training vs Testing for resampled dataset.

The model was designed with four different classifiers were built with the resampled dataset. The classifiers used are Logistic regression, KNN, SVC and Decision tree. The hyperparameters were tuned using GridSearchCV. As shown in the figure-4, the model seems to overfit the dataset. Among all the classifiers, KNN performs reasonably well which doesn’t overfit the model. The specifications of this model are given below:

Classifiers - Logistic regression, KNN, SVC and Decision tree.

Train:test - 80:20

Total resampled data - 984

Outliers - 35

final samples - 949

The same model was tested for different testing sample size. The results are shown below. As shown in the figure-5, all the models performs well for the original datasets. The experiment was carried out with various testing data sizes from 5% to 70% for all four classifiers.

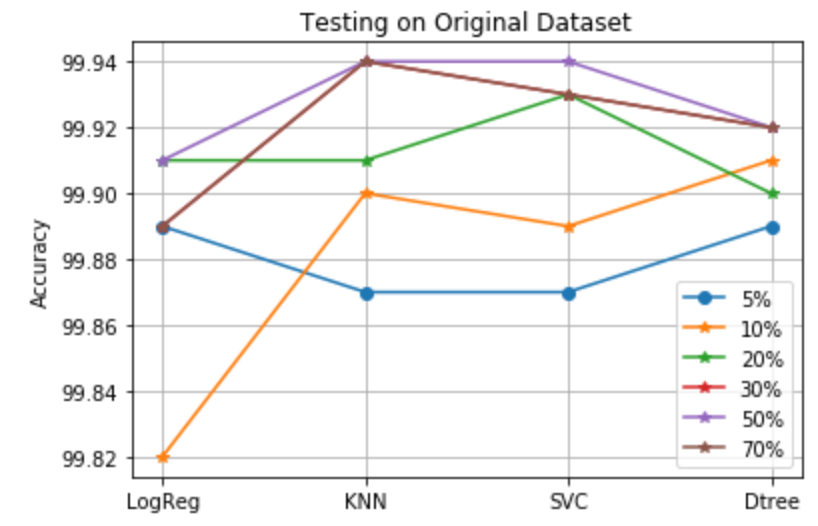


Fig-5: Testing on original dataset with various testing sample size.

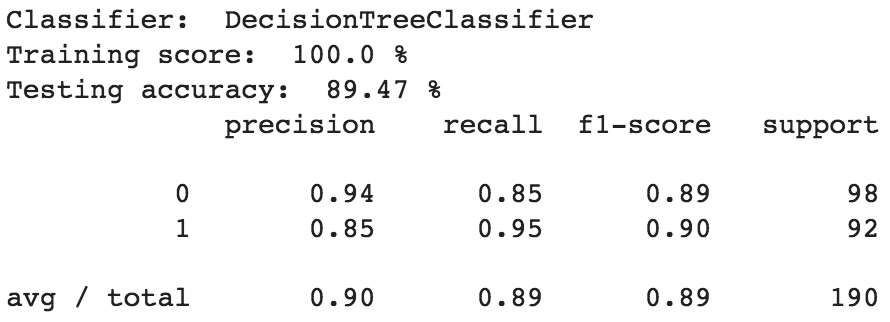
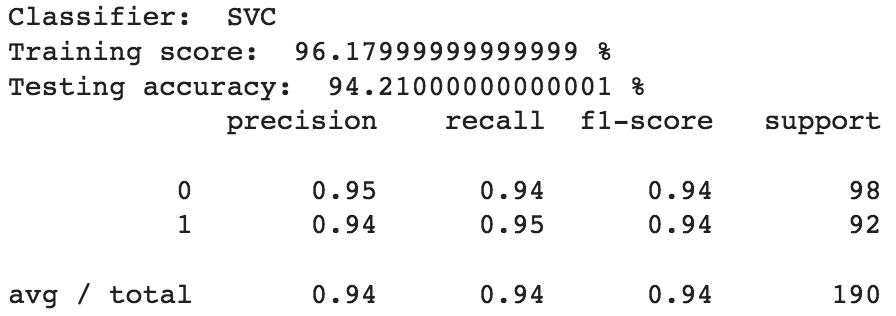
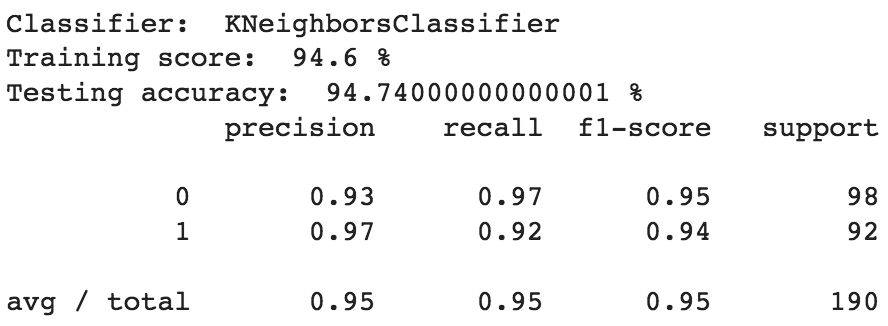
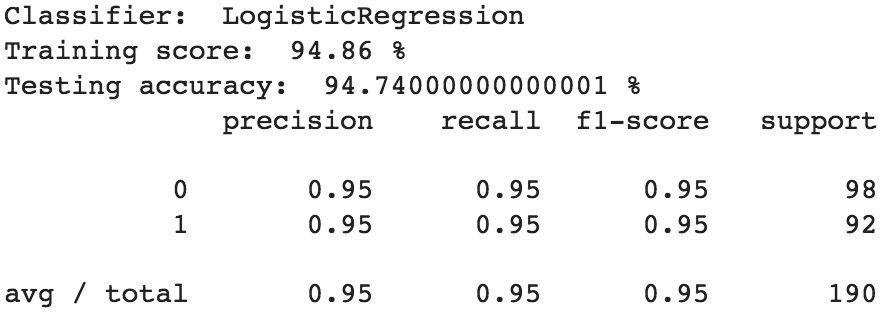




Fig-6: ROC curve for resampled dataset.

**Conclusion:**

Different classifiers were tested for highly skewed and imbalance datasets. Different solutions to skewed dataset were presented and models were designed and tested with four classifiers: Logistic regression, KNN, SVC and Decision tree.

**References:**

1. Credit Card Fraud Detection Kaggle challenge: <https://www.kaggle.com/mlg-ulb/creditcardfraud/home>
2. Sklearn Packages: [http://scikit-learn.org/](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html)
3. Jupyter Notebook: http://jupyter.org/

**Appendix:**

1. Correlation Heatmap for Credit Card Fraud Detection dataset:

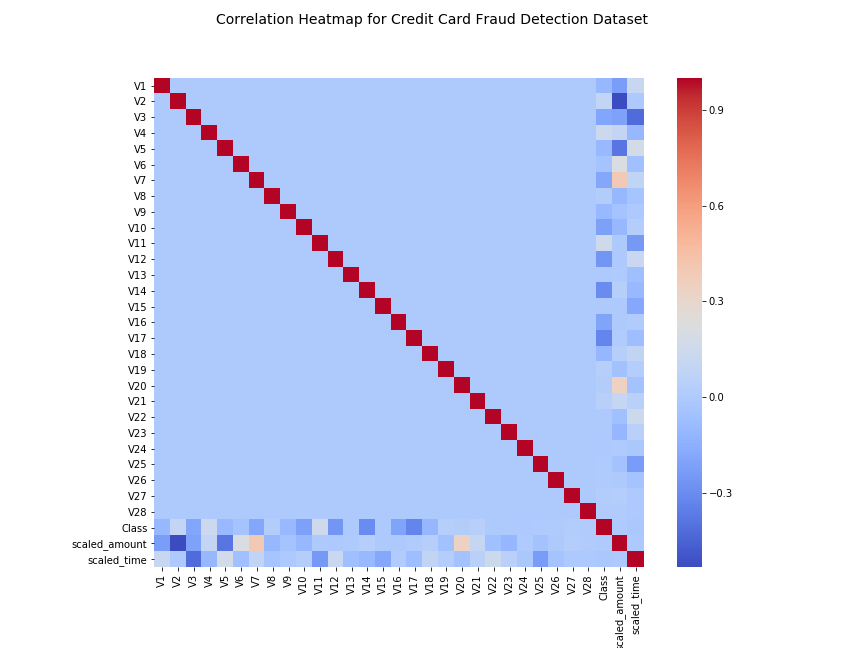


Fig-7: Correlation Heatmap for Credit Card fraud detection on original dataset.

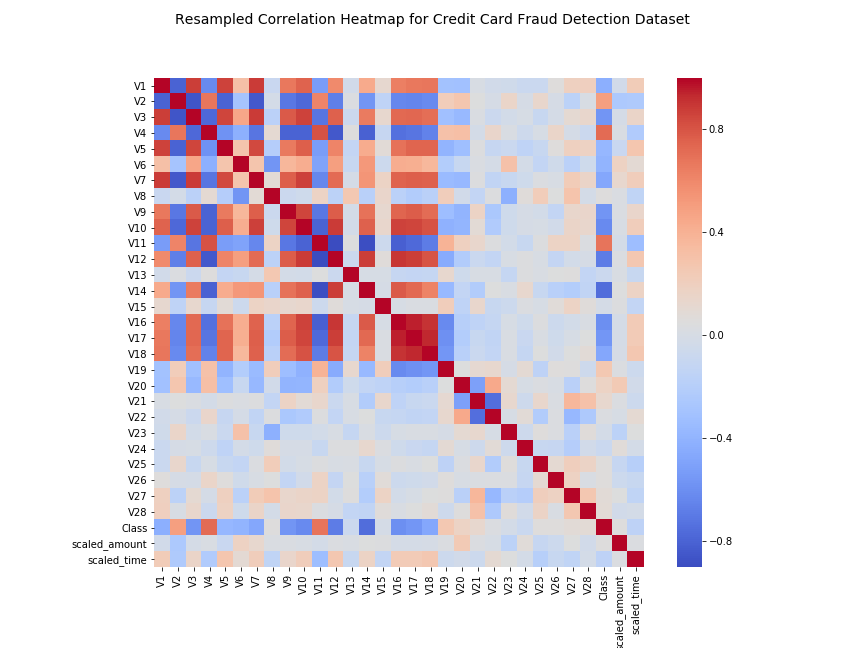


Fig-8: Correlation Heatmap for Credit card fraud detection on resampled dataset.

**Source Code:**

*#credit card fraud detection*

*#kaggle challenge*

*#*[*https://www.kaggle.com/mlg-ulb/creditcardfraud/home*](https://www.kaggle.com/mlg-ulb/creditcardfraud/home)

*import pandas as pd*

*data = pd.read\_csv("creditcard.csv")*

*data.head(5)*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*import numpy as np*

*%matplotlib inline*

*colors = ["#0101DF", "#DF0101"]*

*sns.countplot('Class', data=data, palette=colors)*

*plt.title('Class Distributions \n (0: No Fraud | 1: Fraud)', fontsize=12)*

*#split datasets to test-train*

*from sklearn.preprocessing import StandardScaler*

*data['Amount'] = StandardScaler().fit\_transform(data['Amount'].reshape(-1, 1))*

*data['Time'] = StandardScaler().fit\_transform(data['Time'].reshape(-1, 1))*

*# Correlation Heatmap for original datasets*

*correlation\_matrix = data.corr()*

*fig = plt.figure(figsize=(12,9))*

*fig.suptitle('Correlation Heatmap for Credit Card Fraud Detection Dataset',*

*fontsize=14);*

*sns.heatmap(correlation\_matrix,cmap='coolwarm', square = True)*

*plt.show()*

*#random under sampling*

*data = data.sample(frac=1)*

*# amount of fraud classes 492 rows.*

*fraud\_data = data.loc[data['Class'] == 1]*

*non\_fraud\_data = data.loc[data['Class'] == 0][:492]*

*normal\_distributed\_data = pd.concat([fraud\_data, non\_fraud\_data])*

*# Shuffle dataframe rows*

*new\_data = normal\_distributed\_data.sample(frac=1, random\_state=42)*

*new\_data.head()*

*sns.countplot('Class', data=new\_data, palette=colors)*

*plt.title('Equally Distributed Classes\n (0: No Fraud | 1: Fraud)', fontsize=12)*

*plt.show()*

*# Correlation Heatmap for resampled datasets*

*correlation\_matrix = new\_data.corr()*

*fig = plt.figure(figsize=(12,9))*

*fig.suptitle('Resampled Correlation Heatmap for Credit Card Fraud Detection Dataset',*

*fontsize=14);*

*sns.heatmap(correlation\_matrix,cmap='coolwarm',square = True)*

*plt.show()*

*from sklearn.manifold import TSNE*

*from sklearn.decomposition import PCA, TruncatedSVD*

*import matplotlib.patches as mpatches*

*import time*

*# new\_data is from the random undersample data (fewer instances)*

*X = new\_data.drop('Class', axis=1)*

*y = new\_data['Class']*

*# T-SNE Implementation*

*t0 = time.time()*

*X\_reduced\_tsne = TSNE(n\_components=2, random\_state=42).fit\_transform(X.values)*

*t1 = time.time()*

*print("T-SNE took {:.2} s".format(t1 - t0))*

*# PCA Implementation*

*t0 = time.time()*

*X\_reduced\_pca = PCA(n\_components=2, random\_state=42).fit\_transform(X.values)*

*t1 = time.time()*

*print("PCA took {:.2} s".format(t1 - t0))*

*# TruncatedSVD*

*t0 = time.time()*

*X\_reduced\_svd = TruncatedSVD(n\_components=2, algorithm='randomized', random\_state=42).fit\_transform(X.values)*

*t1 = time.time()*

*print("Truncated SVD took {:.2} s".format(t1 - t0))*

*f, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(24,6))*

*f.suptitle('Clusters using Dimensionality Reduction', fontsize=14)*

*blue\_patch = mpatches.Patch(color='#0A0AFF', label='No Fraud')*

*red\_patch = mpatches.Patch(color='#AF0000', label='Fraud')*

*# t-SNE scatter plot*

*ax1.scatter(X\_reduced\_tsne[:,0], X\_reduced\_tsne[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)*

*ax1.scatter(X\_reduced\_tsne[:,0], X\_reduced\_tsne[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)*

*ax1.set\_title('t-SNE', fontsize=14)*

*ax1.grid(True)*

*ax1.legend(handles=[blue\_patch, red\_patch])*

*# PCA scatter plot*

*ax2.scatter(X\_reduced\_pca[:,0], X\_reduced\_pca[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)*

*ax2.scatter(X\_reduced\_pca[:,0], X\_reduced\_pca[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)*

*ax2.set\_title('PCA', fontsize=14)*

*ax2.grid(True)*

*ax2.legend(handles=[blue\_patch, red\_patch])*

*# TruncatedSVD scatter plot*

*ax3.scatter(X\_reduced\_svd[:,0], X\_reduced\_svd[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)*

*ax3.scatter(X\_reduced\_svd[:,0], X\_reduced\_svd[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)*

*ax3.set\_title('Truncated SVD', fontsize=14)*

*ax3.grid(True)*

*ax3.legend(handles=[blue\_patch, red\_patch])*

*plt.show()*

*from sklearn.linear\_model import LogisticRegression*

*from sklearn.svm import SVC*

*from sklearn.neighbors import KNeighborsClassifier*

*from sklearn.tree import DecisionTreeClassifier*

*from sklearn.ensemble import RandomForestClassifier*

*import collections*

*# Undersampling before cross validating (prone to overfit)*

*X = new\_data.drop('Class', axis=1)*

*y = new\_data['Class']*

*print(X.shape, y.shape)*

*# Our data is already scaled we should split our training and test sets*

*from sklearn.model\_selection import train\_test\_split*

*# This is explicitly used for undersampling.*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)*

*# Turn the values into an array for feeding the classification algorithms.*

*X\_train = X\_train.values*

*X\_test = X\_test.values*

*y\_train = y\_train.values*

*y\_test = y\_test.values*

*# Let's implement simple classifiers*

*classifiers = {*

*"LogisiticRegression": LogisticRegression(),*

*"KNearest": KNeighborsClassifier(),*

*"Support Vector Classifier": SVC(),*

*"DecisionTreeClassifier": DecisionTreeClassifier()*

*}*

*# Wow our scores are getting even high scores even when applying cross validation.*

*from sklearn.model\_selection import cross\_val\_score*

*for key, classifier in classifiers.items():*

*classifier.fit(X\_train, y\_train)*

*training\_score = cross\_val\_score(classifier, X\_train, y\_train, cv=5)*

*print("Classifiers: ", classifier.\_\_class\_\_.\_\_name\_\_, "Has a training score of", round(training\_score.mean(), 2) \* 100, "% accuracy score")*

*# validate data*

*def test(X\_train, y\_train):*

*log\_reg\_score = cross\_val\_score(log\_reg, X\_train, y\_train, cv=5)*

*print('Logistic Regression Accuracy: ', round(log\_reg\_score.mean() \* 100, 2).astype(str) + '%')*

*knears\_score = cross\_val\_score(knears\_neighbors, X\_train, y\_train, cv=5)*

*print('Knears Neighbors Accuracy: ', round(knears\_score.mean() \* 100, 2).astype(str) + '%')*

*svc\_score = cross\_val\_score(svc, X\_train, y\_train, cv=5)*

*print('Support Vector Accuracy: ', round(svc\_score.mean() \* 100, 2).astype(str) + '%')*

*tree\_score = cross\_val\_score(tree\_clf, X\_train, y\_train, cv=5)*

*print('DecisionTree Classifier Accuracy: ', round(tree\_score.mean() \* 100, 2).astype(str) + '%')*

*test(X\_train, y\_train)*

*test(X\_test, y\_test)*