**MINI PROJECT REPORT ON**

**Credit Card Fraud Detection Using Machine Learning Techniques**

Submitted in partial fulfillment of the requirements for

the award of the degree of

**BACHELOR OF TECHNOLOGY**

**Submitted by**

119003047- Enamandram Phani Kaushik -Computer Science and Engineering

119003233 - Sasanka Mouli VSS - Computer Science and Engineering



**Under the Guidance of**

**Dr. Santhi.B**

**Professor**

**School of Computing**

**SASTRA DEEMED TO BE UNIVERSITY**

(A University established under section 3 of the UGC Act, 1956)

Tirumalaisamudram

Thanjavur – 613401

**November (2018)**

**SHANMUGHA**

**ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY**

**(SASTRA DEEMED TO BE UNIVERSITY)**

**(A University Established under section 3 of the UGC Act, 1956)**

**TIRUMALAISAMUDRAM, THANJAVUR – 613401**



**BONAFIDE CERTIFICATE**

Certified that this project work entitled “**CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING TECHNIQUES**” submitted to the Shanmugha Arts, Science, Technology & Research Academy (SASTRA Deemed to be University), Tirumalaisamudram - 613401 by Enamandram Phani Kaushik(119003047), B.Tech CSE, Sasanka Mouli VSS (119003233), B.Tech CSE in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in their respective programme. This work is an original and independent work carried out under my guidance, during the period July 2018 - November 2018.

**Dr. B.Santhi ASSOCIATE DEAN**

**GUIDE SCHOOL OF COMPUTING**

Submitted for Project Viva Voce held on\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Examiner -I Examiner-II**

**TABLE OF CONTENTS**

Page

ACKNOWLEDGEMENTS ………….…………………………………...………………... (iii)

ABSTRACT……………………………………………………..……………..…………… (iv)

CHAPTER 1 BASE PAPER………..………………………………………………….….… (1)

CHAPTER 2 MERITS AND DEMERITS OF THE BASE PAPER ……………....……..... (4)

CHAPTER 3 SOURCE CODE……………………………………………………………... (5)

CHAPTER 4 SNAPSHOTS…..…….……………………………………….….…….…….. (21)

CHAPTER 5 CONCLUSION AND FUTURE PLANS ………..……………….…………. (28)

CHAPTER 6 REFERENCES…………………………….……………………………....… (29)

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to Prof. R. Sethuraman, Vice Chancellor for his encouragement during the span of my academic life at SASTRA University.

I would forever remain grateful and would like to thank **Dr. S. Vaidhyasubramaniam, Dean – Planning & Development, Dr. S. Swaminathan, Director CeNTAB** and **Prof. G.Bhalachandran, Registrar** for their overwhelming support provided during my course span in SASTRA University.

I am extremely grateful to **Dr. Umamakeswari. A, Associate Dean, School of Computing** for her constant support, motivation and academic help extended for the past three years of my life in School of Computing.

I would specially thank and express my gratitude to **Dr.Santhi.B, Professor,School of Computing** for providing me an opportunity to do this project and for her guidance and support to successfully complete this project.

I also thank all the Teaching and Non-teaching faculty, and all other people who have directly or indirectly helped me through their support, encouragement and all other assistance extended for completion of my project and for successful completion of all courses during my academic life at SASTRA University. Finally, I thank God Almighty, my parents and all others who helped me acquire this interesting project and aided me in completing it within the deadline without much struggle. iii

**ABSTRACT**

With Splurge of Internet the Online Transactions have taken a wave. Millions of people are transacting online through various means of which credit card is one mean. There are many number of fraud transactions being recorded. In Order to safeguard the interest of the customer the banks have brought in various security measures. Based on the features augments like location, transaction amount one detect whether the transaction is a fraudulent or genuine transaction. These systems are called Fraud Detection Systems.

We apply various models on the given dataset and conclude on which model gives us with a best output. As the given dataset is an imbalanced dataset we need to balance it for which we make use of SMOTE (Synthetic Minority Oversampling Technique), this technique would balance the dataset. The models we use are K-Means (Unsupervised Learning), Logistic Regression (Supervised Learning), and Random Forest Classifier (Supervised Learning).

**KEYWORDS:** Credit Card Fraud, SMOTE, Classifier

iv

**CHAPTER 1: BASE PAPER**

Detecting frauds in credit card transactions is perhaps one of the best test beds for computational intelligence algorithms. In fact, this problem involves a number of relevant challenges, namely: concept drift (customers’ habits evolve and fraudsters change their strategies over time), class imbalance (genuine transactions far outnumber frauds), and veriﬁcation latency (only a small set of transactions are timely checked by investigators). However, the vast majority of learning algorithms that have been proposed for fraud detection rely on assumptions that hardly hold in a real-world fraud-detection system (FDS). This lack of realism concerns two main aspects:

1) The way and timing with which supervised information is provided and

2) The measures used to assess fraud-detection performance.

This paper has three major contributions. First, we propose, with the help of our industrial partner, a formalization of the fraud-detection problem that realistically describes the operating conditions of FDSs that everyday analyzes massive streams of credit card transactions. We also illustrate the most appropriate performance measures to be used for fraud-detection purposes. Second, we design and assess a novel learning strategy that effectively addresses class imbalance, concept drift, and veriﬁcation latency. Third, in our experiments, we demonstrate the impact of class unbalance and concept drift in a real-world data stream containing more than 75 million transactions, authorized over a time window of three years. The keywords like Concept drift, Credit card fraud detection, learning in non-stationary environments, unbalanced classiﬁcation are explained in the following sections below.

**Introduction**

The main challenges from a learning perspective are class imbalance, namely, genuine transactions far outnumber frauds, and concept drift, namely, transactions might change their statistical properties over time. Thus, in practice, most of supervised samples are provided with a substantial delay, a problem known as veriﬁcation latency. The only recent supervised information made available to update the classiﬁer is provided through the alert– feedback interaction. The main contributions of this paper are as follows.

1) We describe the mechanisms regulating a real-world FDS, and provide a formal model of the articulated classiﬁcation problem to be addressed in fraud detection.

2) We introduce the performance measures that are considered in a real-world FDS.

3) Within this sound and realistic model, we propose an effective learning strategy for addressing the above challenges, including the veriﬁcation latency and the alert– feedback interaction. This learning strategy is tested on a large number of credit card transactions.

**Layers of Control in FDS**

1) Terminal: Performs conventional security check on all payment request (pin, card status)

2) Transaction Blocking Rules: These if else statements to block fraud transactions.

3) Scoring Rules: Assign score to the each transaction higher the score likely to be fraud.

4) Data driven Model: Estimates the probability for each feature vectors.

4) Investigators: Professionals who analyze frauds and false alarms.

**Feature Augmentation**:

During feature augmentation, a speciﬁc set of aggregated features associated with each authorized transactions is computed, to provide additional information about the purchase and better discriminate frauds from genuine transactions.

**Supervised Information:**

Overall, there are two types of supervised information: 1) feedbacks provided by investigators that are limited in number but refer to recent transactions and 2) delayed supervised transactions, which are the vast majority for which the labels become available after several days.

**Data-Driven Approaches in Credit Card Fraud Detection:**

Several classiﬁcation algorithms have been tested on credit card transactions to detect frauds, including neural networks, logistic regression, association rules, support vector machines, modiﬁed Fisher discriminant analysis and decision trees. Several studies have reported random forest (RF) to achieve the best performance.

**Performance Measure for Fraud Detection:**

The typical performance measure for fraud-detection problems is the AUC. AUC can be estimated by means of the Mann–Whitney statistic and its value can be interpreted as the probability that a classiﬁer ranks frauds higher than genuine transactions. Another ranking measure frequently used in fraud detection is average precision, which corresponds to the area under the precision– recall curve.

Major Challenges to be Addressed in a Real-World FDS:

1) Class Imbalance

2) Concept Drift

3) Alert–Feedback Interaction and Sample Selection Bias

**Conclusion:**

Feedbacks play a central role in the proposed learning strategy, which consists in separately training a classiﬁer on feedbacks.

**CHAPTER 2**

**MERITS AND DEMERITS OF THE BASE PAPER**

**Merits**

* Addresses the concept drift (customers’ habits evolve and fraudsters change their strategies over time),
* Addresses class imbalance (genuine transactions far outnumber frauds)
* Addresses veriﬁcation latency (only a small set of transactions are timely checked by investigators
* Implements Real time Fraud Detection System.

**Demerits**

* SMOTE algorithm brings in the tradeoff between precision and recall values
* The algorithm gets random values which may affect the f1-score

**CHAPTER 3**

**Source Code**

*import warnings*

*warnings.filterwarnings("ignore")*

*from tkinter import \**

*import tkinter.messagebox*

*from tkinter import filedialog*

*import numpy as np*

*import sklearn as sk*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*from pandas\_ml import ConfusionMatrix*

*import pandas\_ml as pdml*

*from sklearn.preprocessing import scale*

*from sklearn import linear\_model*

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

*from sklearn.cluster import KMeans*

*from sklearn import metrics*

*from sklearn.decomposition import PCA*

*from sklearn.ensemble import RandomForestClassifier*

*from sklearn.metrics import auc,roc\_curve*

*root=Tk()*

*root.geometry("800x600")*

*root.title("Credit Card Fraud Detection")*

*var=StringVar()*

*choice=StringVar()*

*choice.set("-select-")*

*d={'ACC':[],'TPR':[]}*

*aucarr={'auc':[]}*

*def logistic\_regression():*

*print("------------------------LOGISTIC REGRESSION-----------------------")*

*df = pd.read\_csv(var.get(), low\_memory=False)*

*df = df.sample(frac=1).reset\_index(drop=True)*

*frauds = df.loc[df['Class'] == 1]*

*non\_frauds = df.loc[df['Class'] == 0]*

*print("\n")*

*print("We have", len(frauds), "fraud data points and", len(non\_frauds), "nonfraudulent data points.\n")*

*X = df.iloc[:,:-1]*

*y = df['Class']*

*print("X and y sizes, respectively:", len(X), len(y))*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.35)*

*'''print("\nTrain and test sizes, respectively:", len(X\_train), len(y\_train), "|", len(X\_test), len(y\_test))*

*print("Total number of frauds:", len(y.loc[df['Class'] == 1]))*

*print("Number of frauds on y\_test:", len(y\_test.loc[df['Class'] == 1]))*

*print("Number of frauds on y\_train:", len(y\_train.loc[df['Class'] == 1]))'''*

*logistic = linear\_model.LogisticRegression(C=1e5)*

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

*print("\nScore: ", logistic.score(X\_test, y\_test))*

*y\_predicted = np.array(logistic.predict(X\_test))*

*y\_right = np.array(y\_test)*

*confusion\_matrix = ConfusionMatrix(y\_right, y\_predicted)*

*print("\n\nConfusion matrix:\n%s" % confusion\_matrix)*

*#confusion\_matrix.plot(normalized=True)*

*T = Text(root, height=60, width=60)*

*T.pack(pady=20,side=BOTTOM, fill=Y)*

*for l in confusion\_matrix.stats():*

*T.insert(END,[l,confusion\_matrix.stats()[l]])*

*T.insert(END,"\n")*

*d['ACC'].append(confusion\_matrix.stats()['ACC']\*100)*

*d['TPR'].append(confusion\_matrix.stats()['TPR']\*100)*

*fpr,tpr,thresholds=roc\_curve(y\_right, y\_predicted)*

*aucarr['auc'].append(auc(fpr,tpr))*

*#plt.show()*

*def logistic\_reg\_smote():*

*l=1*

*if(l==1):*

*print("------------------------LOGISTIC REGRESSION WITH SMOTE-----------------------")*

*df = pd.read\_csv(var.get(), low\_memory=False)*

*df = df.sample(frac=1).reset\_index(drop=True)*

*frauds = df.loc[df['Class'] == 1]*

*non\_frauds = df.loc[df['Class'] == 0]*

*print("\nWe have", len(frauds), "fraud data points and", len(non\_frauds), "nonfraudulent data points.")*

*X = df.iloc[:,:-1]*

*y = df['Class']*

*print("X and y sizes, respectively:", len(X), len(y))*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.35)*

*print("Train and test sizes, respectively:", len(X\_train), len(y\_train), "|", len(X\_test), len(y\_test))*

*print("Total number of frauds:", len(y.loc[df['Class'] == 1]))*

*print("Number of frauds on y\_test:", len(y\_test.loc[df['Class'] == 1]))*

*print("Number of frauds on y\_train:", len(y\_train.loc[df['Class'] == 1]))*

*df2 = pdml.ModelFrame(X\_train, target=y\_train)*

*sampler = df2.imbalance.over\_sampling.SMOTE()*

*sampled = df2.fit\_sample(sampler)*

*print("\nSize of training set after over sampling:", len(sampled))*

*X\_train\_sampled = sampled.iloc[:,1:]*

*y\_train\_sampled = sampled['Class']*

*logistic = linear\_model.LogisticRegression(C=1e5)*

*logistic.fit(X\_train\_sampled, y\_train\_sampled)*

*print("Score: ", logistic.score(X\_test, y\_test))*

*y\_predicted1 = np.array(logistic.predict(X\_test))*

*y\_right1 = np.array(y\_test)*

*confusion\_matrix1 = ConfusionMatrix(y\_right1, y\_predicted1)*

*print("\n\nConfusion matrix:\n%s" % confusion\_matrix1)*

*#confusion\_matrix1.plot(normalized=True)*

*T = Text(root, height=60, width=60)*

*T.pack(pady=20,side=BOTTOM, fill=Y)*

*for l in confusion\_matrix1.stats():*

*T.insert(END,[l,confusion\_matrix1.stats()[l]])*

*T.insert(END,"\n")*

*d['ACC'].append(confusion\_matrix1.stats()['ACC']\*100)*

*d['TPR'].append(confusion\_matrix1.stats()['TPR']\*100)*

*fpr,tpr,thresholds=roc\_curve(y\_right1, y\_predicted1)*

*aucarr['auc'].append(auc(fpr,tpr))*

*#plt.show()*

*def random\_forest():*

*l=1*

*if(l==1):*

*print("------------------------RANDOM FOREST-----------------------")*

*df = pd.read\_csv(var.get(), low\_memory=False)*

*df = df.sample(frac=1).reset\_index(drop=True)*

*frauds = df.loc[df['Class'] == 1]*

*non\_frauds = df.loc[df['Class'] == 0]*

*print("\nWe have", len(frauds), "fraud data points and", len(non\_frauds), "nonfraudulent data points.")*

*X = df.iloc[:,:-1]*

*y = df['Class']*

*print("X and y sizes, respectively:", len(X), len(y))*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.35)*

*print("Train and test sizes, respectively:", len(X\_train), len(y\_train), "|", len(X\_test), len(y\_test))*

*print("Total number of frauds:", len(y.loc[df['Class'] == 1]))*

*print("Number of frauds on y\_test:", len(y\_test.loc[df['Class'] == 1]))*

*print("Number of frauds on y\_train:", len(y\_train.loc[df['Class'] == 1]))*

*clf= RandomForestClassifier()*

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

*y\_predicted1 =np.array(clf.predict(X\_test))*

*y\_right1=np.array(y\_test)*

*confusion\_matrix1=ConfusionMatrix(y\_right1,y\_predicted1)*

*print("\n\nConfusion matrix:\n%s" % confusion\_matrix1)*

*#confusion\_matrix1.plot(normalized=True)*

*T = Text(root, height=60, width=60)*

*T.pack(pady=20,side=BOTTOM, fill=Y)*

*for l in confusion\_matrix1.stats():*

*T.insert(END,[l,confusion\_matrix1.stats()[l]])*

*T.insert(END,"\n")*

*d['ACC'].append(confusion\_matrix1.stats()['ACC']\*100)*

*d['TPR'].append(confusion\_matrix1.stats()['TPR']\*100)*

*fpr,tpr,thresholds=roc\_curve(y\_right1, y\_predicted1)*

*aucarr['auc'].append(auc(fpr,tpr))*

*#plt.show()*

*def random\_for\_smote():*

*l=1*

*if(l==1):*

*print("------------------------RANDOM FOREST WITH SMOTE-----------------------")*

*df = pd.read\_csv(var.get(), low\_memory=False)*

*df = df.sample(frac=1).reset\_index(drop=True)*

*frauds = df.loc[df['Class'] == 1]*

*non\_frauds = df.loc[df['Class'] == 0]*

*print("We have", len(frauds), "fraud data points and", len(non\_frauds), "nonfraudulent data points.\n")*

*X = df.iloc[:,:-1]*

*y = df['Class']*

*print("X and y sizes, respectively:", len(X), len(y))*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.35)*

*print("\nTrain and test sizes, respectively:", len(X\_train), len(y\_train), "|", len(X\_test), len(y\_test))*

*print("Total number of frauds:", len(y.loc[df['Class'] == 1]))*

*print("Number of frauds on y\_test:", len(y\_test.loc[df['Class'] == 1]))*

*print("Number of frauds on y\_train:", len(y\_train.loc[df['Class'] == 1]))*

*df2 = pdml.ModelFrame(X\_train, target=y\_train)*

*sampler = df2.imbalance.over\_sampling.SMOTE()*

*sampled = df2.fit\_sample(sampler)*

*print("\nSize of training set after over sampling:", len(sampled))*

*X\_train\_sampled = sampled.iloc[:,1:]*

*y\_train\_sampled = sampled['Class']*

*clf= RandomForestClassifier()*

*clf.fit(X\_train\_sampled, y\_train\_sampled)*

*y\_predicted1 =np.array(clf.predict(X\_test))*

*y\_right1=np.array(y\_test)*

*confusion\_matrix1=ConfusionMatrix(y\_right1,y\_predicted1)*

*print("\n\nConfusion matrix:\n%s" % confusion\_matrix1)*

*#confusion\_matrix1.plot(normalized=True)*

*T = Text(root, height=60, width=60)*

*T.pack(pady=20,side=BOTTOM, fill=Y)*

*for l in confusion\_matrix1.stats():*

*T.insert(END,[l,confusion\_matrix1.stats()[l]])*

*T.insert(END,"\n")*

*d['ACC'].append(confusion\_matrix1.stats()['ACC']\*100)*

*d['TPR'].append(confusion\_matrix1.stats()['TPR']\*100)*

*fpr,tpr,thresholds=roc\_curve(y\_right1, y\_predicted1)*

*aucarr['auc'].append(auc(fpr,tpr))*

*#plt.show()*

*def choose():*

*tempdir = filedialog.askopenfilename(parent=root, initialdir= "C:/Users/Kaushik/Desktop/", title='Please select a directory')*

*var.set(tempdir)*

*if(len(var.get())>0):*

*mEntry.insert(0,var)*

*def run():*

*if(len(var.get())==0):*

*tkinter.messagebox.showinfo(title="Dialog Box", message="Cannot upload empty file!")*

*elif(not(var.get()).endswith('csv')):*

*tkinter.messagebox.showinfo(title="Dialog Box", message="Unsupported format of the file\n The file should be csv")*

*if((choice.get())=='-select-'):*

*tkinter.messagebox.showinfo(title="Dialog Box", message="Please select the Algorithm")*

*elif((var.get()).endswith('csv') and (choice.get())=='Kmeans'):*

*print("------------------------"+str(choice.get())+"-----------------------")*

*print("\n")*

*df = pd.read\_csv(var.get(), low\_memory=False)*

*#print(df.head())*

*X = df.iloc[:,:-1]*

*y = df['Class']*

*X\_scaled = scale(X)*

*pca = PCA(n\_components=2)*

*X\_reduced = pca.fit\_transform(X\_scaled)*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_reduced, y, test\_size = 0.33, random\_state=500)*

*kmeans = KMeans(init='k-means++', n\_clusters=2, n\_init=10)*

*kmeans.fit(X\_train)*

*h = .01*

*x\_min, x\_max = X\_reduced[:, 0].min() - 1, X\_reduced[:, 0].max() + 1*

*y\_min, y\_max = X\_reduced[:, 1].min() - 1, X\_reduced[:, 1].max() + 1*

*xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))*

*Z = kmeans.predict(np.c\_[xx.ravel(), yy.ravel()])*

*Z = Z.reshape(xx.shape)*

*plt.figure(1)*

*plt.clf()*

*plt.imshow(Z, interpolation='nearest',*

*extent=(xx.min(), xx.max(), yy.min(), yy.max()),*

*cmap=plt.cm.Paired,*

*aspect='auto', origin='lower')*

*plt.plot(X\_reduced[:, 0], X\_reduced[:, 1], 'k.', markersize=2)*

*# Plot the centroids as a white X*

*centroids = kmeans.cluster\_centers\_*

*plt.scatter(centroids[:, 0], centroids[:, 1],*

*marker='x', s=169, linewidths=3,*

*color='w', zorder=10)*

*plt.title('K-means clustering on the credit card fraud dataset\n'*

*'Centroids are marked with white cross')*

*plt.xlim(x\_min, x\_max)*

*plt.ylim(y\_min, y\_max)*

*plt.xticks(())*

*plt.yticks(())*

*predictions = kmeans.predict(X\_test)*

*pred\_fraud = np.where(predictions == 1)[0]*

*real\_fraud = np.where(y\_test == 1)[0]*

*false\_pos = len(np.setdiff1d(pred\_fraud, real\_fraud))*

*pred\_good = np.where(predictions == 0)[0]*

*real\_good = np.where(y\_test == 0)[0]*

*false\_neg = len(np.setdiff1d(pred\_good, real\_good))*

*false\_neg\_rate = false\_neg/(false\_pos+false\_neg)*

*accuracy = (len(X\_test) - (false\_neg + false\_pos)) / len(X\_test)*

*print("Accuracy:\n", accuracy)*

*print("False negative rate (with respect to misclassifications): ", false\_neg\_rate)*

*print("False negative rate (with respect to all the data): ", false\_neg / len(predictions))*

*print("False negatives, false positives, mispredictions:", false\_neg, false\_pos, false\_neg + false\_pos)*

*print("Total test data points:", len(X\_test))*

*plt.show()*

*elif((var.get()).endswith('csv') and (choice.get())=='Logistic Regression'):*

*logistic\_regression()*

*elif((var.get()).endswith('csv') and (choice.get())=='Logistic Regression with SMOTE'):*

*logistic\_reg\_smote()*

*elif((var.get()).endswith('csv') and (choice.get())=='Random Forest'):*

*random\_forest()*

*elif((var.get()).endswith('csv') and (choice.get())=='Random Forest with SMOTE'):*

*random\_for\_smote()*

*def compare(): print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COMPARISON\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")*

*if(len(var.get())==0):*

*tkinter.messagebox.showinfo(title="Dialog Box", message="Cannot upload empty file!")*

*elif(not(var.get()).endswith('csv')):*

*tkinter.messagebox.showinfo(title="Dialog Box", message="Unsupported format of the file\n The file should be csv")*

*d['ACC']=[]*

*d['TPR']=[]*

*aucarr['auc']=[]*

*logistic\_regression()*

*logistic\_reg\_smote()*

*random\_forest()*

*random\_for\_smote()*

*objects = ('Logistic Regression','Logistic with SMOTE','Random forest','Random forest with SMOTE')*

*y\_pos = np.arange(len(objects))*

*plt.bar(y\_pos, aucarr['auc'], align='center', alpha=0.5)*

*plt.xticks(y\_pos, objects)*

*plt.ylabel('Area Under Curve')*

*plt.xlabel('Models')*

*plt.title('Algorithms Comparisons')*

*plt.show()*

***#MAIN FUNCTION:***

*frame=Frame(root)*

*frame.pack()*

*button=Button(frame,text="Choose File",command=choose)*

*button.pack(padx=50,pady=50,side=LEFT)*

*choices = ['Kmeans','Logistic Regression','Random Forest','Random Forest with SMOTE','Logistic Regression with SMOTE']*

*popupMenu = OptionMenu(frame,choice, \*choices)*

*popupMenu.pack(side=LEFT)*

*slogan=Button(frame,text="Performance Measures",command=run)*

*slogan.pack(padx=50,pady=50,side=LEFT)*

*button1=Button(frame,text="Compare",command=compare)*

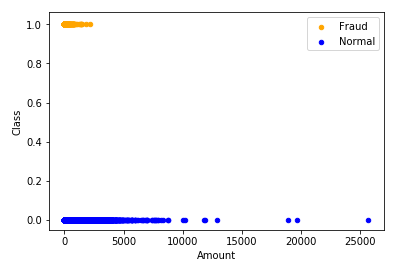
*button1.pack(padx=50,pady=50,side=LEFT)*

*mEntry= Entry(root,width=60,textvariable=var).pack()*

*root.mainloop()*

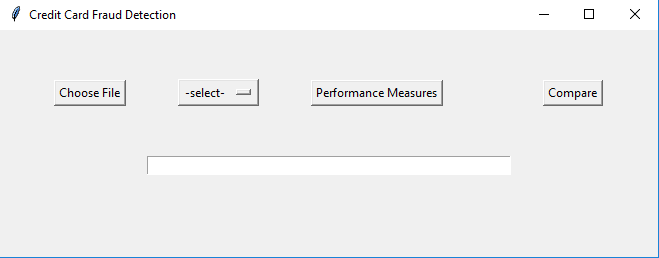
**CHAPTER 4**

**SNAPSHOTS**

****

The dataset visualization.

GUI for the Analysis



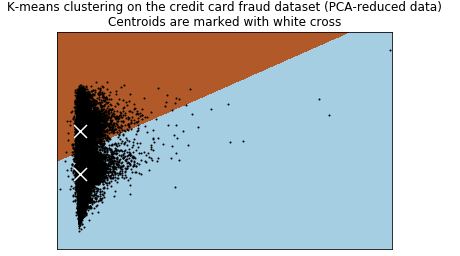
Choose File allows the user to choose the dataset

Select allows the user to choose the Algorithm to implement

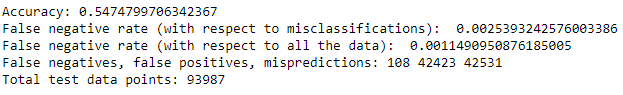
Performance Measure allows the user to see the performance metrics of the selected Algorithm.

Compare allows the user to compare the results of all the Algorithms.

**K-Means**



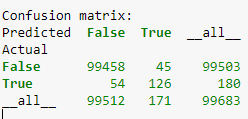
This plot shows us the result of applying k-means algorithm on the given dataset

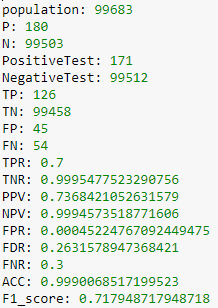


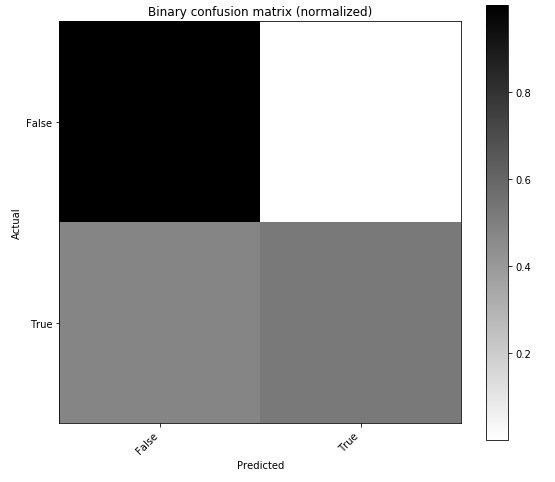
The Accuracy value is calculated on a test data of 0.33 of total with 0.67 as train data. We can

Infer that the accuracy value is too low so move onto the next algorithm

**Logistic Regression**

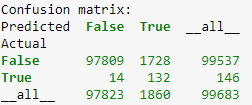


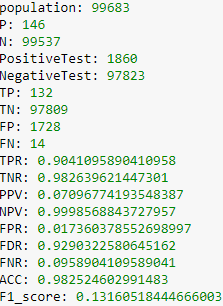


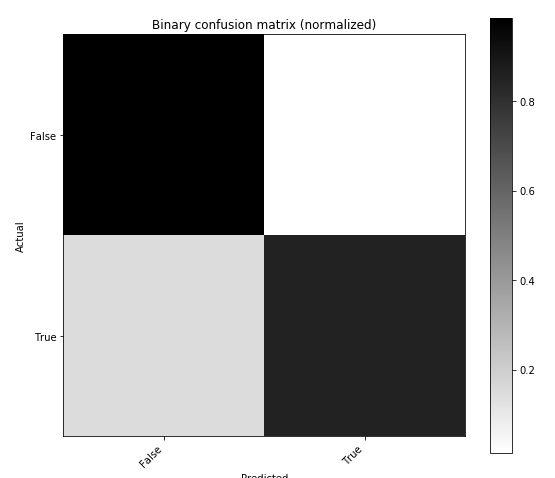


The Results of Logistic Regression are show above we can infer that the FNR rate is high so now we will balance the dataset and implement Logistic Regression .

**Logistic Regression with SMOTE**

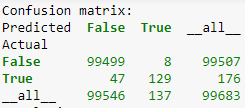


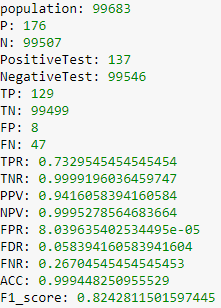


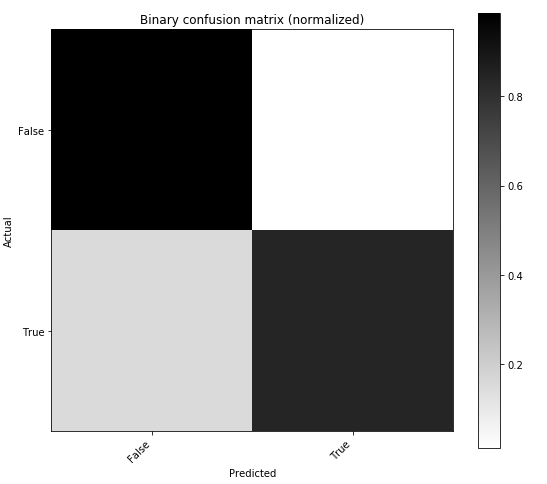


The Results of Logistic Regression with SMOTE are show above we can infer that the accuracy is decreased.

**Random Forest Algorithm**

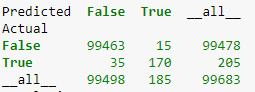


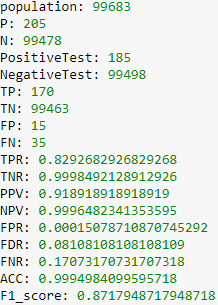


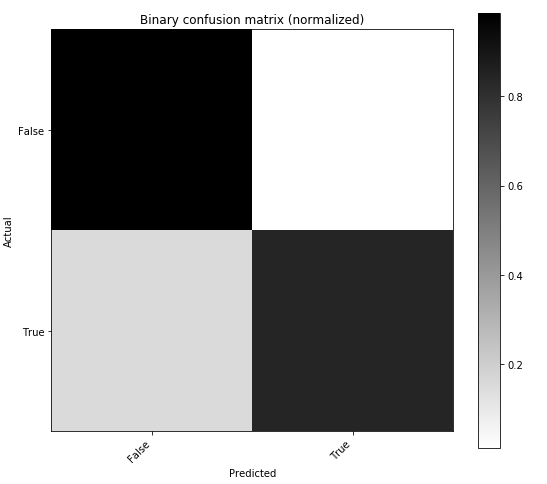


The Results of Random Forest are show above we can infer that the accuracy is fine but there is scope to improve the result.

**Random Forest with SMOTE**



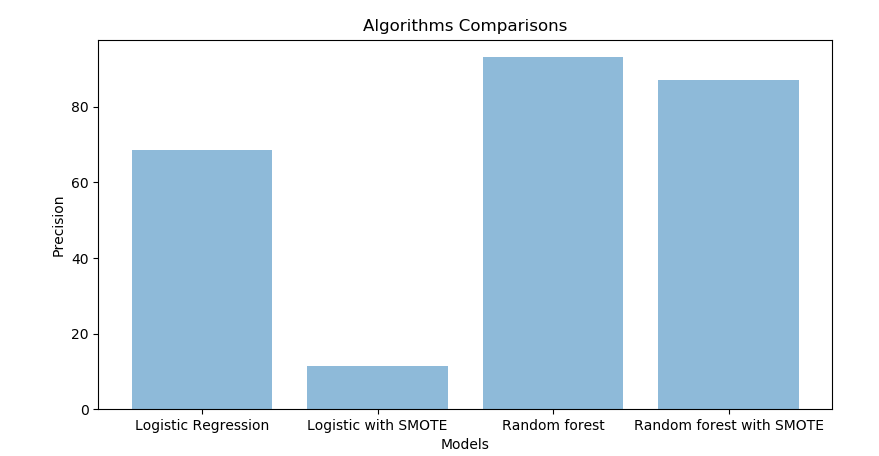




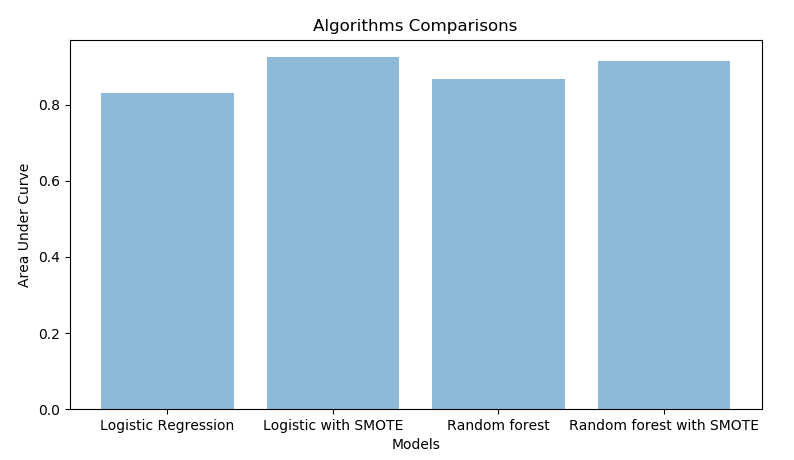
The Results of Random Forest with SMOTE are show above we can infer that the TPR rate has increased compared to without balancing.

**Comparison of the Results**

As the accuracy value is very near we use precision as metric



As we can infer from the bar graph plotter the algorithm which gives which gives the best precision value is Random Forest.



From the values of AUC we can infer that Logistic Regression with SMOTE has higher AUC value.

**CHAPTER 5**

**CONCLUSION AND FUTURE PLANS**

**Conclusion**

* The K-means clustering model produced a low accuracy of 54.27%. Of the wrongly predicted transactions, 99.75% were false positives, giving only 0.24% false negatives, or 0.11% of the validation set. However, the false negative rate was only so low due to the extremely low proportion of frauds in the dataset. In reality, 112 of the 176 frauds were misclassiﬁed as non-frauds, giving this a true accuracy rate of 36.36%.
* Therefore, K-means would not be the preferred model for this dataset, as it did not correctly predict frauds and it also produced a lot of false positives.
* The logistic regression gave us the best results. The logistic regression gave us a great accuracy rate of 99.88%, with 0.079% of the validation set being false negatives (or 0.49% of the number of misclassiﬁcations).
* The logistic regression with oversampling gave us an interesting result, The accuracy was 98.01%, with 1.56% of the validation set being false negatives (or 3.12% of the misclassiﬁcations) but has the highest AUC value and low precision value.
* The Random forest gave us best results with 99.9% accuracy and higher precision value with less number of false negatives compared to smote.

**Future Plans**

* Extend the algorithms to a real time e-commerce site.
* To implement using Autoencoders which might be a more efficient algorithm.
* The study of adaptive and possibly nonlinear aggregation methods for the classiﬁers trained on feedbacks and delayed supervised samples

**CHAPTER 6:**

**REFERENCES**

1. **Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, Fellow, IEEE, and Gianluca Bontempi, Senior Member, IEEE**(2018),”*Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy*”, VOL. 29, NO. 8
2. **L. Breiman**, “*Random forests*,” Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.
3. **N. Chawla, K. Bowyer, L. O. Hall, and W. P. Kegelmeyer**, “*SMOTE: Synthetic minority over-sampling technique*,” J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.
4. **J. Demšar**, “*Statistical comparisons of classiﬁers over multiple data sets*,” J. Mach. Learn. Res., vol. 7, pp. 1–30, Jan. 2006.
5. **D. Hand**, “*Measuring classiﬁer performance: A coherent alternative to the area under the ROC curve*,” Mach. Learn., vol. 77, no. 1, pp. 103–123, 2009.
6. **C. Alippi, G. Boracchi, and M. Roveri**, “*A just-in-time adaptive classiﬁcation system based on the intersection of conﬁdence intervals rule*,” Neural Netw., vol. 24, no. 8, pp. 791–800, 2011.
7. **R. Bolton and D. Hand**, “*Statistical fraud detection: A review*,” Stat. Sci., vol. 17, no. 3, pp. 235–249, 2002.
8. **H. He and E. A. Garcia**, “*Learning from imbalanced data*,” IEEE Trans. Knowl. Data Eng., vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
9. **G. Widmer and M. Kubat**, “*Learning in the presence of concept drift and hidden contexts*,” Mach. Learn., vol. 23, no. 1, pp. 69–101, 1996
10. **C. Whitrow, D. J. Hand**, P. Juszczak, D. J. Weston, and N. M. Adams, “*Transaction aggregation as a strategy for credit card fraud detection,*” Data Mining Knowl. Discovery, vol. 18, no. 1, pp. 30–55, 2009.
11. **V. Van Vlasselaer et al.**,“*APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions*,” Decision Support Syst., vol. 75, pp. 38–48, Jul. 2015
12. **W. N. Street and Y. Kim**, “*A streaming ensemble algorithm (SEA) for large-scale classiﬁcation*,” in Proc. 7th Int. Conf. Knowl. Discovery Data Mining, 2001, pp. 377–382.
13. **J. C. Schlimmer and R. H. Granger**, Jr., “*Incremental learning from noisy data*,” Mach. Learn., vol. 1, no. 3, pp. 317–354, 1986.
14. **C. Phua, V. Lee, K. Smith, and R. Gayler. (Sep. 2010)**. “*A comprehensive survey of data mining-based fraud detection research.*” [Online]. Available: https://arxiv.org/abs/1009.6119