ASSIGNMENT NO. 5

TITLE: Support vector machine classification for credit card fraud detection dataset

AIM: Apply the Support vector machine for classification on a dataset obtained from UCI ML repository.

SAVITRIBAI PHULE PUNE UNIVERSITY

A PRELIMINARY PROJECT REPORT ON

"SUPPORT VECTOR MACHINE CLASSIFICATION FOR CREDIT CARD FRAUD DETECTION DATASET"

SUBMITTED TOWARDS THE PARTIAL FULFILMENT OF THE REQUIREMENTS OF

LABORATORY PRACTICE - III (MACHINE LEARNING)

Academic Year: 2019-20

By:

Harish Choudhary	**BECOA120**
Nitiraj Desai	**BECOA122**
Shubham Desai	**BECOA123**
Onkar Deshmukh	**BECOA124**

Under The Guidance of

Prof. Alka Londhe

DEPARTMENT OF COMPUTER ENGINEERING,

PCET'S PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

Sector No. 26, Pradhikaran, Nigdi,
Pune - 411044

PCET'S PIMPRI CHINCHWAD COLLEGE OF ENGINEERING
Sector No. 26, Pradhikaran, Nigdi,
Pune - 411044

DEPARTMENT OF COMPUTER ENGINEERING

Certificate

This is to certify that the Mini Project report entitled

"SUPPORT VECTOR MACHINE CLASSIFICATION FOR CREDIT CARD FRAUD DETECTION DATASET"

Submitted By

Harish Choudhary	**BECOA120**
Nitiraj Desai	**BECOA122**
Shubham Desai	**BECOA123**
Onkar Deshmukh	**BECOA124**

is approved by Prof. **Alka Londhe** for submission. It is certified further that, to the best of my knowledge, the report represents work carried out by my students as the partial fulfillment for BE. Computer Engineering (Semester II) Laboratory-III Work (Machine Learning) as prescribed by the Savitribai Phule Pune University for the academic year 2019-20.

Prof. Alka Londhe
(Mini Project Guide)

Place: Pune
Date:

ABSTRACT

In day to day life credit cards are used for purchasing goods and services with the help of virtual card for online transaction or physical card for offline transaction. In a physical card-based purchase, the cardholder presents his card physically to a merchant for making a payment. To carry out fraudulent transactions in this kind of purchase; an attacker has to steal the credit card. If the cardholder does not realize the loss of card, it can lead to a substantial financial loss to the credit card company. In online payment mode, attackers need only little information for doing fraudulent transaction (secure code, card number, expiration date etc.). In this purchase method, mainly transactions will be done through Internet or telephone. To commit fraud in these types of purchases, a fraudster simply needs to know the card details. Most of the time, the genuine cardholder is not aware that someone else has seen or stolen his card information. The only way to detect this kind of fraud is to analyze the spending patterns on every card and to figure out any inconsistency with respect to the "usual" spending patterns.

Index

List of Figures & Tables

Chapter 1: Introduction

a) Problem Statement: -

To implement a system for Credit Card Fraud Detection based on Data Mining using Support Vector Machines

b) Project Idea: -

There are enormous transactions are processed every day. Most of the people do their payments using credit or debit cards instead of cash. Imbalanced Data i.e. most of the transactions *(99.8%)* are not fraudulent which makes it really hard for detecting the fraudulent ones. Misclassified Data can be another major issue, as not every fraudulent transaction is caught and reported. Adaptive techniques used against the model by the scammers. The model used should be simple and fast enough to detect the anomaly and classify it as a fraudulent transaction as quickly as possible.

c) Motivation: -

A major challenge involved in credit card fraud detection is to recognize fraudulent credit card transactions so that the customers of credit card companies are not charged for items that they did not purchase. The available credit card fraud detection database consists of both numerical and categorical data. Before further processing, cleaning and filtering are applied on these records in order to filter the irrelevant data from the database.

d)Scope: -

The proposed system classifies and predicts the credit card frauds using SVM technique.

Chapter 2: Project Design

a) H/W, S/W, Resources, Requirements & their details explanation: -

1). Hardware requirements: - Intel core i3 processor. Memory at least 1 GB ram is used. Hard disk space minimum 1 GB for database usage.

2). Software requirements: - Operating System Ubuntu /Windows.

3). Resources: -

4). Technical Details: (Platform, Language, API's Libraries, Packages):-

Platform: - PyCharm using tkinter libraries is one of the most popular Python bindings for the gui and framework design.

Language: - Python programming language is used to front end for gui design and SM classifier is used for back end. **V**

Package and API Libraries: - Tkinter is the standard GUI library for python. Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. Python when combined with Tkinter provides a fast and easy way to create GUI application. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. **Matplotlib** is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a matlab-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users. **NumPy** is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. The **Python Imaging Library (PIL)** adds image processing capabilities to your Python interpreter. This library supports many file formats and provides powerful image processing and graphics capabilities. The Tkinter **tkMessageBox** has various methods to display a **message box**.

a) **Dataset Design**

The **Credit Card Fraud Detection** dataset is used in order to learn the SVM algorithm. This dataset is freely available on **Kaggle.** The dataset contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features **V1, V2, ... V28 are the principal components obtained with PCA**, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature '**Time**' contains the float elapsed between each transaction and the first transaction in the dataset. The feature '**Amount**' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature '**Class**' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

- Time (in seconds) [Number of seconds elapsed between this transaction and the first transaction in the dataset]
- V1 (in float) [may be result of a PCA Dimensionality reduction to protect user identities and sensitive features(v1-v28)]
- V2 (in float)
- V3 (in float)
- V4 (in float)
- V5 (in float)
- V6 (in float)
- V7 (in float)
- V8 (in float)
- V9 (in float)
- V10 (in float)
- V11 (in float)
- V12 (in float)
- V13 (in float)
- V14 (in float)
- V15 (in float)
- V16 (in float)
- V17 (in float)
- V18 (in float)
- V19 (in float)
- V20 (in float)
- V21 (in float)
- V22 (in float)
- V23 (in float)
- V24 (in float)
- V25 (in float)
- V26 (in float)
- V27 (in float)
- V28 (in float)
- Amount (Transaction amount)
- **Target** (1 for fraudulent transactions, 0 otherwise)

Chapter 3: Module Description

a) Block diagram

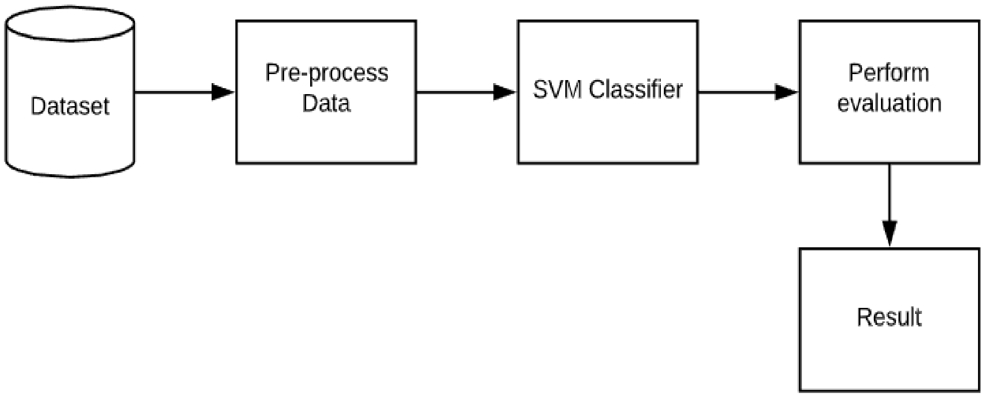


Fig.1. System Block Diagram

SUPPORT VECTOR MACHINE (SVM) CLASSIFIER: - The Support Vector Machine (SVM) is a powerful machine learning tool based on firm statistical and mathematical foundations concerning generalization and optimization theory. It offers a robust technique for many aspects of data mining including classification, regression, and outlier detection. SVM is based on Vapnik's statistical learning theory and falls at the intersection of kernel methods and maximum margin classifiers. Support vector machines have been successfully applied to many real-world problems such as face detection, intrusion detection, handwriting recognition, information extraction, and others. Support Vector Machine is an attractive method due to its high generalization capability and its ability to handle high-dimensional input data

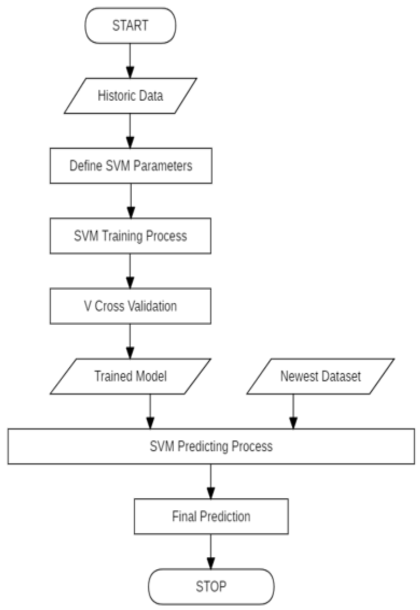


Fig.2. Flowchart for SVM Classifier

1.1 Advantages and Disadvantages

Advantages:

1. Regularization capabilities: SVM has L2 Regularization feature. So, it has good generalization capabilities which prevent it from over-fitting.

2. Handles non-linear data efficiently: SVM can efficiently handle non-linear data using Kernel trick

3. Solves both Classification and Regression problems: SVM can be used to solve both classification and regression problems. SVM is used for classification problems while SVR (Support Vector Regression) is used for regression problems.

4. Stability: A small change to the data does not greatly affect the hyperplane and hence the SVM. So the SVM model is stable.

Disadvantages:

1. Choosing an appropriate Kernel function is difficult: Choosing an appropriate Kernel function (to handle the non-linear data) is not an easy task. It could be tricky and complex. In case of using a high dimension Kernel, you might generate too many support vectors which reduce the training speed drastically.

2. Extensive memory requirement: Algorithmic complexity and memory requirements of SVM are very high. You need a lot of memory since you have to store all the support vectors in the memory and this number grows abruptly with the training dataset size.

3. Requires Feature Scaling: One must do feature scaling of variables before applying SVM.

4. Long training time: SVM takes a long training time on large datasets.

5. Difficult to interpret: SVM model is difficult to understand and interpret by human beings unlike Decision Trees.

1.2 Working

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).

The followings are important concepts in SVM −

- **Support Vectors** − Data points that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane** − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps −

- First, SVM will generate hyperplanes iteratively that segregates the classes in best way.
- Then, it will choose the hyperplane that separates the classes correctly.

Chapter 4: Results & Discussion

a) source code:-

```
#===========================Packages=======================================
from tkinter import *;
from tkinter.constants import *
import numpy as np
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report,
mean_absolute_error, mean_squared_error
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
import matplotlib.pyplot as plt



#=========================MainClass=======================================

class MachineLearning:
  def __init__(self):
    self.data = None
    self.table = None
    self.selection_x = None
    self.selection_y = None
    self.X = None
    self.y = None
    self.X_test = None
    self.X_train = None
    self.y_test = None
    self.y_train = None
    self.DF = None
```

```
    self.svm_model = None
    self.svm_predictions = None
    self.Accuracy = None

    self.le = LabelEncoder()

    self.window = Tk()
    self.color = 'grey95'
    self.window.geometry('900x620')
    self.window.resizable(False, False)
    self.window.configure(background=self.color)
    self.window.title('Machine Learning SVM')

   self.heading = Label(self.window, text="Machine Learning SVM", bg=self.color,
   pady=20, font=("Helvetica", 35, "bold"))
   self.heading.place(width=620, height=100, bordermode=OUTSIDE, x=0, y=0)

# =======================File Selection and Viewing=========================
   self.frame = LabelFrame(self.window, text='File Selection', bg=self.color)
   self.frame.place(width=580, height=80, bordermode=OUTSIDE, x=20, y=100)

   self.name_label = Label(self.frame, text="File Name : ", bg=self.color, padx=10,
   pady=10,font=("Helvetica", 15))
   self.name_label.place(width=120, height=30, bordermode=INSIDE, x=10, y=13)

   self.name = StringVar()
   self.name_entry = Entry(self.frame, exportselection=False,textvariable=self.name,
   font=("Helvetica", 12))
   self.name_entry.place(width=250, height=30, bordermode=INSIDE, x=130, y=13)

   self.name_select = Button(self.frame, text='Select', command=lambda:
   self.select())
   self.name_select.place(width=50, height=30, bordermode=INSIDE, x=395, y=13)

   self.df_show = Button(self.frame, text='Show', command=lambda:
   self.create_table(), state=DISABLED)
   self.df_show.place(width=50, height=30, bordermode=INSIDE, x=455, y=13)

# ===================GRAPH OF IRIS DATASET=================================
   self.graph = LabelFrame(self.window, text='Graph Plotting', bg=self.color)
   self.graph.place(width=700, height=80, bordermode=OUTSIDE, x=20, y=200)

   self.Amountpl = Button(self.graph, text=' Time vs Amount Fraud', command=lambda:
   self.AmountFraud_plot(),state=DISABLED)
   self.Amountpl.place(width=250, height=30, bordermode=INSIDE, x=5, y=13)



# ====================Train Test Split======================================
   self.ttsplit = LabelFrame(self.window, text='Train Test Split', bg=self.color)
   self.ttsplit.place(width=700, height=80, bordermode=OUTSIDE, x=20, y=300)

   self.trainsplit = Button(self.ttsplit, text='split', command=lambda:
   self.train_test_split(), state=DISABLED)
   self.trainsplit.place(width=125, height=30, bordermode=INSIDE, x=5, y=13)

# ====================SVM GUI===============================================
   self.svm = LabelFrame(self.window, text='Support Vector Machine Linear',
```

```python
    bg=self.color)
    self.svm.place(width=700, height=80, bordermode=OUTSIDE, x=20, y=400)

    self.svm_pred = Button(self.svm, text='Predict', command=lambda: self.pred_svm(),
    state=DISABLED)
    self.svm_pred.place(width=125, height=30, bordermode=INSIDE, x=5, y=13)

    self.report = Button(self.svm, text='Report', command=lambda: self.svm_report(),
    state=DISABLED)
    self.report.place(width=125, height=30, bordermode=INSIDE, x=140, y=13)

    self.confusion_matrix = Button(self.svm, text=' Confusion_Matrix',
    command=lambda: self.cm_svm(),state=DISABLED)
    self.confusion_matrix.place(width=125, height=30, bordermode=INSIDE, x=280, y=13)

    self.plot_data = Button(self.svm, text='Plot', command=lambda: self.plot(),
    state=DISABLED)
    self.plot_data.place(width=125, height=30, bordermode=INSIDE, x=420, y=13)

    self.svm_error = Button(self.svm, text='Error', command=lambda:
    self.errors_svm(), state=DISABLED)
    self.svm_error.place(width=125, height=30, bordermode=INSIDE, x=560, y=13)

    self.window.mainloop()


#============================Selecting an File===========================

  def select(self):
    try:
      self.data = pd.read_csv(self.name.get())
      self.Amountpl['state'] = NORMAL
      self.df_show['state'] = NORMAL
      self.trainsplit['state'] = NORMAL

    except FileNotFoundError:
      self.name.set("Invalid")


  def create_table(self):
    try:
      self.table.window.deiconify()
    except AttributeError:
      if self.data.shape[0] > 152:
        self.table = Table(self.data.head(152), self.window, self.name.get())
      else:
        self.table = Table(self.data, self.window, self.name.get())
    except TclError:
      if self.data.shape[0] > 152:
        self.table = Table(self.data.head(152), self.window, self.name.get())
      else:
        self.table = Table(self.data, self.window, self.name.get())

    self.trainsplit['state'] = NORMAL
    self.svm_pred['state'] = DISABLED
    self.report['state'] = DISABLED
    self.confusion_matrix['state'] = DISABLED
    self.plot_data['state'] = DISABLED
```

```
    self.svm_error['state'] = DISABLED

#==============================Splitting Data  In Train And Test=====================
   def train_test_split(self):
     self.DF = pd.read_csv(self.name.get())
    print(len(self.DF))
     x = self.DF.iloc[:, :-1]
     y = self.DF.iloc[:, -1]
     print("----------------------Dataset Values--------------------------")
     print(self.DF.head(5))
     print("Information about X dataframe:  ")
     print(x.info())
     print("Describing the data")
     self.DF[self.DF['Class'] == 1].describe()

    self.X = self.DF.drop(['Time', 'Class'], axis=1)
    self.y = self.le.fit_transform(self.DF['Class'])
    print(self.y)
    self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(self.X, self.y, test_size=0.20)
    print(self.X_train)
    print("--------------")
    print(self.X_test)
    print("--------------")

    print(self.y_train)
    print("--------------")

    print(self.y_test)



    self.svm_pred['state'] = NORMAL

    print("-------------------------------------------")

#==============================Predict Function For SVM========================
   def pred_svm(self):
    self.svm_model = SVC(C=1.0, kernel='linear', class_weight='balanced')

    self.svm_model.fit(self.X_train, self.y_train)
    self.Accuracy = self.svm_model.score(self.X_train, self.y_train)
    print("Accuracy is=",self.Accuracy*100)
    print("============================================================")

    self.svm_predictions= self.svm_model.predict(self.X_test)
    print("predictions is=",self.svm_predictions)

    self.svm_error['state'] = NORMAL
    self.confusion_matrix['state'] = NORMAL
    self.report['state'] = NORMAL
    self.plot_data['state'] = NORMAL


#=============================== AmountFraud ====================================
def AmountFraud_plot(self):
    AmountFraud(self.window)  # Calling AmountFraud Class

#================================SVM Plotting==============================
   def plot(self): # plot of svm
```

```
        Scatter(self.window, self.y_test, self.svm_predictions)  # Calling Scatter Class

#================================SVM Report================================
    def svm_report(self):  # report svm
        ClassificationReport(self.window, classification_report(self.le.inverse_transform(self.y_test),
        self.le.inverse_transform(self.svm_predictions)), 'Support Vector Machine')
        # Calling ClassificationReport Class


#===============================SVM Error==================================
    def errors_svm(self): # error svm
        temp = [mean_absolute_error(self.y_test, self.svm_predictions), mean_squared_error(self.y_test,
        self.svm_predictions), np.sqrt(mean_squared_error(self.y_test, self.svm_predictions))]
        Errors(self.window, temp, 'SVM')  # calling Error Class


#==========================SVM Confusion Matrix=============================
    def cm_svm(self):
        ConfusionMatrix(self.window, confusion_matrix(self.le.inverse_transform(self.y_test),
                                                      self.le.inverse_transform(self.svm_predictions)),
                        'Support Vector Matrix', self.le.classes_)  # ConfusionMatrix Class



#=====================================Table Class===========================
class Table:
    def __init__(self, data, master, name):
        self.master = master
        self.window = Toplevel(self.master)
        self.data = data
        self.name = name
        self.window.title(self.name)
        self.window.geometry('600x600')
        self.window.minsize(250, 250)

        self.frame = Frame(self.window)
        self.frame.pack(expand=True, fill=BOTH)

        self.canvas = Canvas(self.frame, background='white')

        self.h_scroll = Scrollbar(self.frame, orient=HORIZONTAL, command=self.canvas.xview)
        self.h_scroll.pack(side=BOTTOM, fill=X)
        self.v_scroll = Scrollbar(self.frame, orient=VERTICAL, command=self.canvas.yview)
        self.v_scroll.pack(side=RIGHT, fill=Y)

        self.canvas['xscrollcommand'] = self.h_scroll.set
        self.canvas['yscrollcommand'] = self.v_scroll.set
        self.canvas.pack(expand=True, fill=BOTH)

        self.label_frame = LabelFrame(self.canvas)
        self.canvas.create_window((0, 0), window=self.label_frame, anchor=N + W)

        self.shape = (data.shape[0], data.shape[1])

        Table.add_label(self, 0, 0, '#', font=('Helvetica', 15, 'bold'))
        for j in range(self.shape[1]):
            Table.add_label(self, 0, j + 1, self.data.columns[j], font=('Helvetica', 12, 'bold'))
        self.height = 20
        for i in range(self.shape[0]):
            Table.add_label(self, i + 1, 0, str(i + 1))
```

```
            ar = data.iloc[i].values
            for j in range(len(ar)):
                Table.add_label(self, i + 1, j + 1, ar[j])
        self.window.update()
        self.canvas.configure(scrollregion=self.label_frame.bbox(ALL))

    def add_label(self, i, j, text, font=('Helvetica', 10)):
        if j % 2 == 0:
            color = 'white'
        else:
            color = 'antique white'
        label = Label(self.label_frame, text=text, font=font, bg=color)
        label.grid(row=i, column=j, sticky=E+N+W+S)


#==========================Confusion Matrix Class==============================
class ConfusionMatrix:
    def __init__(self, master, data, name, labels):
        self.data = data
        self.master = master
        self.name = name
        self.labels = sorted(labels)

        self.total = np.sum(self.data)

        self.window = Toplevel(self.master)
        self.window.title(self.name + " Confusion Matrix")
        self.window.resizable(False, False)

        self.total_label = Label(self.window, text=f'Total = {self.total}', font=('Helvetica', 15, 'bold'),
                                                                  bg='antique white')

        self.total_label.grid(row=0, column=0, sticky=(N, S, E, W))

        for i in range(len(self.labels)):
            if i % 2 == 0:
                color = 'white'
            else:
                color = 'antique white'
            Label(self.window, text=f'Predicted\n{self.labels[i]}', font=('Helvetica', 15, 'bold'),
                                  bg=color).grid(row=0, column=i+1, sticky=(N, S, E, W))

        for i in range(len(self.labels)):
            if i % 2 == 0:
                color = 'white'
            else:
                color = 'antique white'
            Label(self.window, text=f'Actual\n{self.labels[i]}', font=('Helvetica', 15, 'bold'),
                                bg=color).grid(row=i+1, column=0, sticky=(N, S, E, W))

            for j in range(len(self.labels)):
                color = ['grey90', 'grey80', 'grey70']
                Label(self.window, text=str(self.data[i][j]), font=('Helvetica', 15, 'bold'),
                        bg=color[(i + j) % 3]).grid(row=i+1, column=j+1, sticky=(N, S, E, W))



#========================Error Class==========================================
class Errors:
    def __init__(self, master, data, name):
```

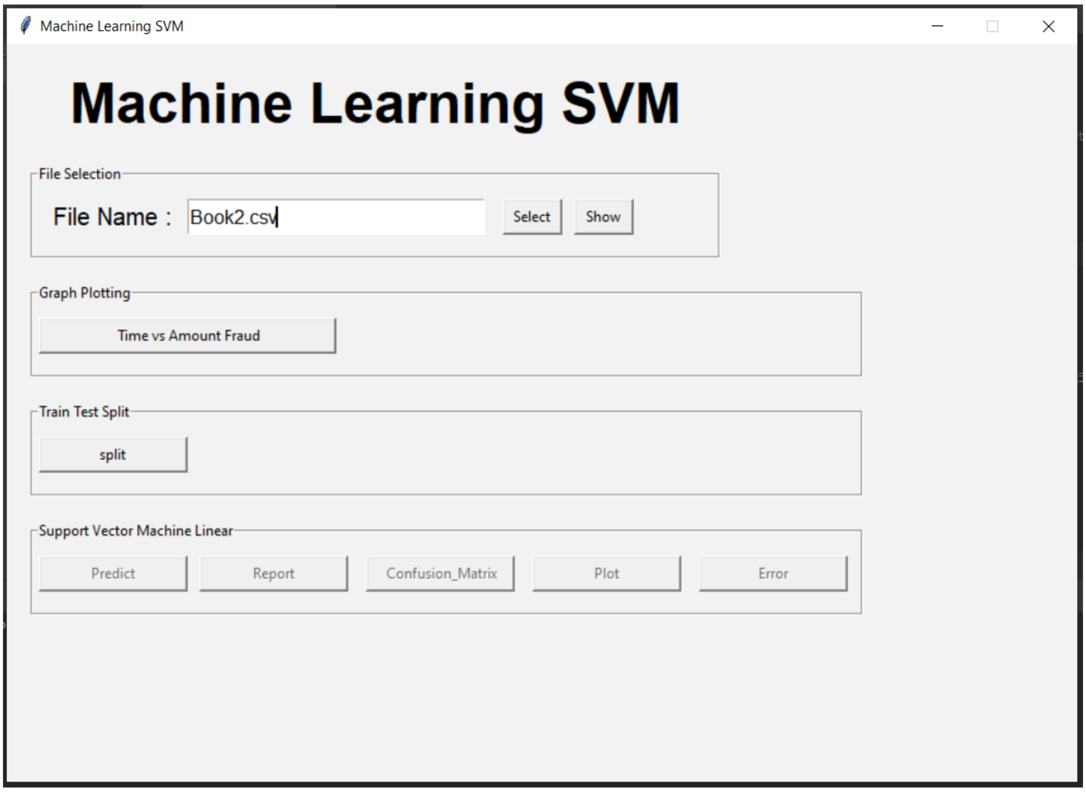
```
        self.master = master
        self.data = data
        self.name = name

        self.window = Toplevel(self.master)
        self.window.title(self.name + " Errors")
        self.window.geometry('500x180')
        self.window.resizable(False, False)

        self.frame = Frame(self.window)
        self.frame.place(width=504, height=184, bordermode=OUTSIDE, x=0, y=0)

        self.text1 = Label(self.frame, text='Mean Absolute Error :', font=('Helvetica', 15, 'bold'), bg='antique white')
        self.text1.place(width=260, height=60, bordermode=INSIDE, x=0, y=0)
        self.text2 = Label(self.frame, text='Mean Squared Error :', font=('Helvetica', 15, 'bold'), bg='white')
        self.text2.place(width=260, height=60, bordermode=INSIDE, x=0, y=60)
        self.text3 = Label(self.frame, text='Root Mean Squared Error: ', font=('Helvetica', 15, 'bold'), bg='antique
       white')
        self.text3.place(width=260, height=60, bordermode=INSIDE, x=0, y=120)

        self.value1 = Label(self.frame, text=str(data[0]), font=('Helvetica', 15, 'bold'), bg='antique white')
        self.value1.place(width=240, height=60, bordermode=INSIDE, x=260, y=0)
        self.value2 = Label(self.frame, text=str(data[1]), font=('Helvetica', 15, 'bold'), bg='white')
        self.value2.place(width=240, height=60, bordermode=INSIDE, x=260, y=60)
        self.value3 = Label(self.frame, text=str(data[2]), font=('Helvetica', 15, 'bold'), bg='antique white')
        self.value3.place(width=240, height=60, bordermode=INSIDE, x=260, y=120)



#==========================Classification Report Class============================
class ClassificationReport:
    def __init__(self, master, data, name):
        self.master = master
        self.data = data
        self.name = name

        self.window = Toplevel(self.master)
        self.window.title(self.name + " Classification Report")
        self.window.configure(background='white')
        self.window.resizable(False, False)
        y = 0

        Label(self.window, text='precision', font=('Helvetica', 15, 'bold'), anchor=E, bg='antique
        white').place(width=100, height=50, bordermode=INSIDE, x=150, y=y)
        Label(self.window, text='recall', font=('Helvetica', 15, 'bold'), anchor=E,
        bg='white').place(width=100, height=50, bordermode=INSIDE, x=250, y=0)
        Label(self.window, text='f1-score', font=('Helvetica', 15, 'bold'), anchor=E, bg='antique
        white').place(width=100, height=50, bordermode=INSIDE, x=350, y=y)
        Label(self.window, text='support', font=('Helvetica', 15, 'bold'), anchor=E,
       bg='white').place(width=100, height=50, bordermode=INSIDE, x=450, y=0)
        y = y + 50

        Label(self.window, bg='antique white').place(width=100, height=10, bordermode=INSIDE, x=150,
        y=y)
        Label(self.window, bg='antique white').place(width=100, height=10, bordermode=INSIDE, x=350,
        y=y)
        y = y + 10
```

```
        self.ar = self.data.split('\n\n')[1:]
        print(self.ar)
        self.part1 = self.ar[0].split('\n')

        for i in self.part1:
            temp = i.split()
            Label(self.window, text=temp[0], font=('Helvetica', 12, 'bold'), anchor=E,
            bg='white').place(width=150, height=30, bordermode=INSIDE, x=0, y=y)
            Label(self.window, text=temp[1], font=('Helvetica', 12), anchor=E, bg='antique
            white').place(width=100, height=30, bordermode=INSIDE, x=150, y=y)
            Label(self.window, text=temp[2], font=('Helvetica', 12), anchor=E, bg='white').place(width=100,
            height=30, bordermode=INSIDE, x=250, y=y)
            Label(self.window, text=temp[3], font=('Helvetica', 12), anchor=E, bg='antique
            white').place(width=100, height=30, bordermode=INSIDE, x=350, y=y)
            Label(self.window, text=temp[4], font=('Helvetica', 12), anchor=E, bg='white').place(width=100,
            height=30, bordermode=INSIDE, x=450, y=y)
            y = y + 30

        Label(self.window, bg='antique white').place(width=100, height=20, bordermode=INSIDE, x=150,
        y=y)
        Label(self.window, bg='antique white').place(width=100, height=20, bordermode=INSIDE, x=350,
        y=y)
        y = y + 20

        self.part2 = self.ar[1].split('\n')

        for i in self.part2:
            if i == '':
                continue
            temp = i.split()
            Label(self.window, text=temp.pop(), font=('Helvetica', 12), anchor=E,
            bg='white').place(width=100, height=30, bordermode=INSIDE, x=450, y=y)
            Label(self.window, text=temp.pop(), font=('Helvetica', 12), anchor=E, bg='antique
            white').place(width=100, height=30, bordermode=INSIDE, x=350, y=y)
            if len(temp) != 1:
                Label(self.window, text=temp.pop(), font=('Helvetica', 12), anchor=E,
                bg='white').place(width=100, height=30, bordermode=INSIDE, x=250, y=y)
            if len(temp) != 1:
                Label(self.window, text=temp.pop(), font=('Helvetica', 12), anchor=E, bg='antique
                white').place(width=100, height=30, bordermode=INSIDE, x=150, y=y)
            else:
                Label(self.window, bg='antique white').place(width=100, height=30, bordermode=INSIDE,
                x=150, y=y)
            Label(self.window, text=" ".join(temp), font=('Helvetica', 12, 'bold'), anchor=E,
            bg='white').place(width=150, height=30, bordermode=INSIDE, x=0, y=y)
            y = y + 30

        self.window.geometry('550x'+str(y))


#=======================Scatter Class=======================================
class Scatter:
    def __init__(self, master, y_test, pred):
        self.master = master
```

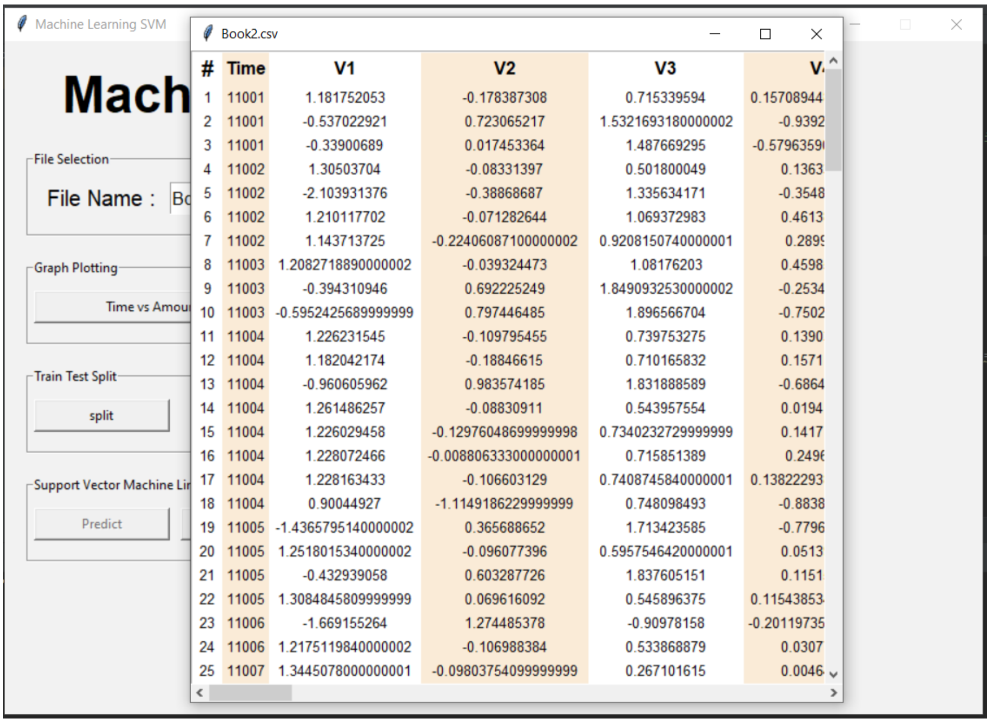
```
        self.y_test = y_test
        print(self.y_test)
        self.pred = pred
        print(self.pred)
        self.window = Toplevel(self.master)
        self.window.title("Scatter Plot (y_test vs predictions)")
        self.window.configure(background='white')
        self.window.resizable(False, False)

        self.figure = Figure(figsize=(5, 5), dpi=100)
        self.sub = self.figure.add_subplot(111,xlabel="Y_Predict", ylabel="Y_Test", title="Y_Predict &
Y_Test")
        self.sub.scatter(self.y_test, self.pred, edgecolor='black')
        self.sub.plot()
        self.sub.legend()
        self.sub.grid(True)
        self.canvas = FigureCanvasTkAgg(self.figure, master=self.window)
        self.canvas.get_tk_widget().pack()
        self.canvas.draw()


#================================ AmountFraud Class==================================
class AmountFraud:
    def __init__(self, master):
        self.master = master
        self.plt=plt
        self.le1 = LabelEncoder()
        self.DF = pd.read_csv('Book2.csv')
        self.df_fraud = self.DF[self.DF['Class'] == 'Fraud']  # Recovery of fraud data
        print(self.df_fraud)
        self.x = self.df_fraud.get("Time")
        print(self.x)
        self.y = self.df_fraud.get("Amount")
        print(self.y)
        self.window = Toplevel(self.master)
        self.window.title("Scatter Plot Amount Fraud")
        self.window.configure(background='white')
        self.window.resizable(False, False)

        self.figure = Figure(figsize=(5, 5), dpi=100)
        self.sub = self.figure.add_subplot(111, xlabel="Time", ylabel="Amount",
title="Time vs Amount Class=0(Fraud)")
        self.sub.scatter(self.x,self.x1, c=self.y)

        self.sub.plot()
        self.sub.legend()
        self.sub.grid(True)

        self.canvas = FigureCanvasTkAgg(self.figure, master=self.window)
        self.canvas.get_tk_widget().pack()
        self.canvas.draw()

#=====================Main Function=========================================
if __name__ == '__main__':
    MachineLearning()

#==================================================================
```

b) Screenshot including GUI:-

Main Page-



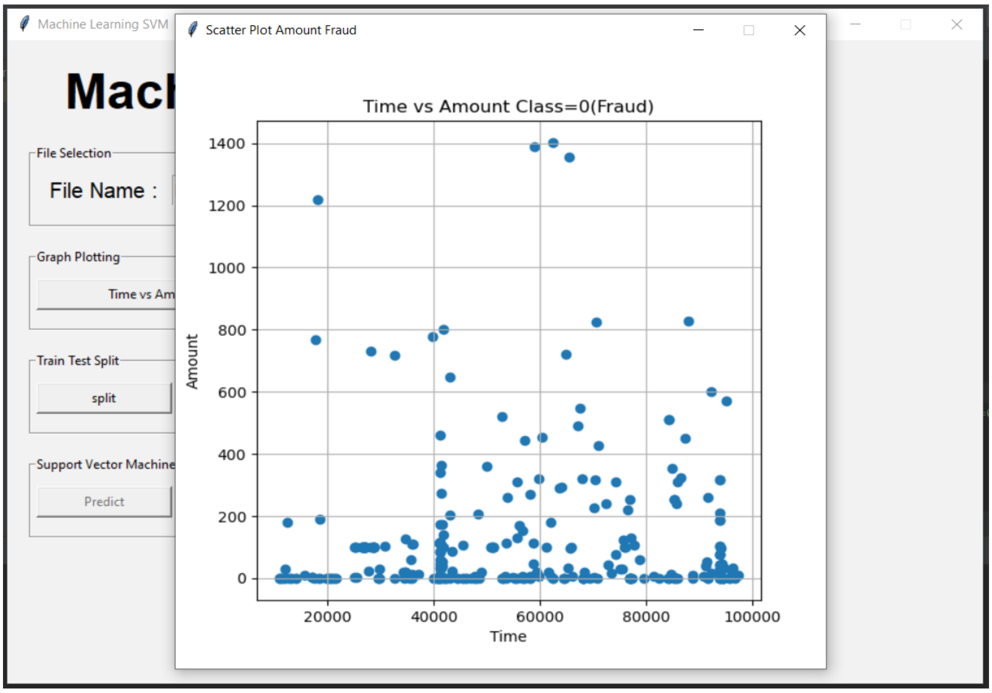
Credit Card Fraud Detection Dataset:-



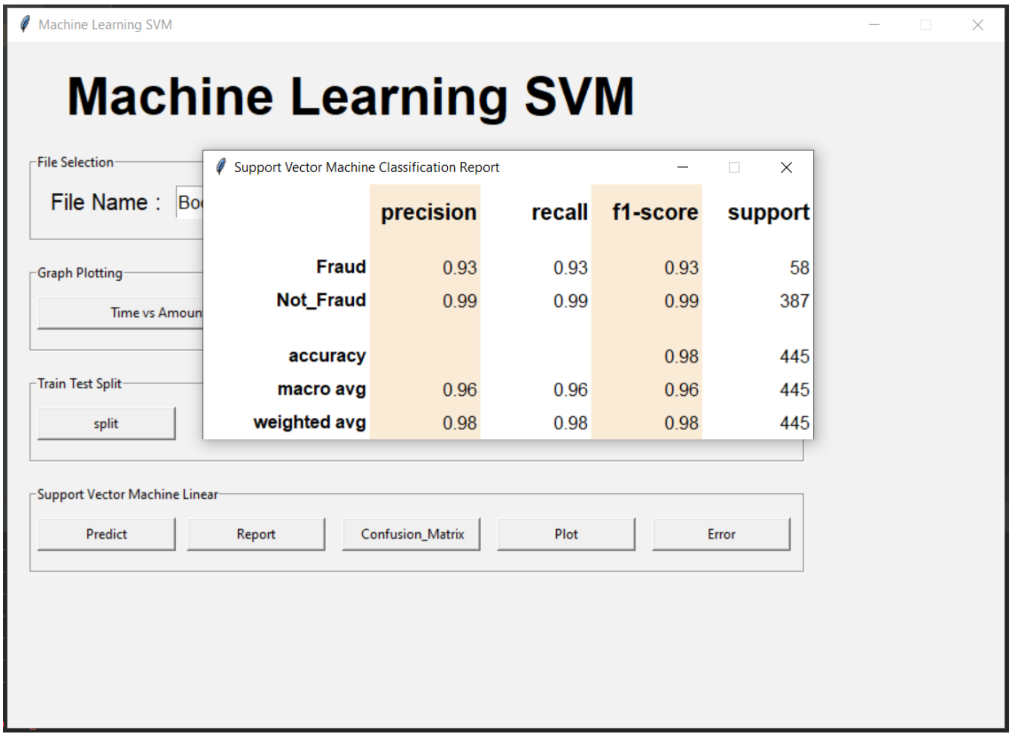
#	Time	V1	V2	V3	V
1	11001	1.181752053	-0.178387308	0.715339594	0.15708944
2	11001	-0.537022921	0.723065217	1.5321693180000002	-0.9392
3	11001	-0.33900689	0.017453364	1.487669295	-0.5796359
4	11002	1.30503704	-0.08331397	0.501800049	0.1363
5	11002	-2.103931376	-0.38868687	1.335634171	-0.3548
6	11002	1.210117702	-0.071282644	1.069372983	0.4613
7	11002	1.143713725	-0.22406087100000002	0.9208150740000001	0.289
8	11003	1.2082718890000002	-0.039324473	1.08176203	0.4598
9	11003	-0.394310946	0.692225249	1.8490932530000002	-0.2534
10	11003	-0.5952425689999999	0.797446485	1.896566704	-0.7502
11	11004	1.226231545	-0.109795455	0.739753275	0.1390
12	11004	1.182042174	-0.18846615	0.710165832	0.1571
13	11004	-0.960605962	0.983574185	1.831888589	-0.6864
14	11004	1.261486257	-0.08830911	0.543957554	0.0194
15	11004	1.226029458	-0.12976048699999998	0.7340232729999999	0.1417
16	11004	1.228072466	-0.008806333000000001	0.715851389	0.249
17	11004	1.228163433	-0.106603129	0.7408745840000001	0.13822293
18	11004	0.90044927	-1.1149186229999999	0.748098493	-0.8838
19	11005	-1.4365795140000002	0.365688652	1.713423585	-0.7796
20	11005	1.2518015340000002	-0.096077396	0.5957546420000001	0.0513
21	11005	-0.432939058	0.603287726	1.837605151	0.1151
22	11005	1.3084845809999999	0.069616092	0.545896375	0.11543853
23	11006	-1.669155264	1.274485378	-0.90978158	-0.20119735
24	11006	1.2175119840000002	-0.106988384	0.533868879	0.0307
25	11007	1.3445078000000001	-0.09803754099999999	0.267101615	0.0046

Accuracy is= 99.54980303882948

Scatter Plot Amount Fraud-

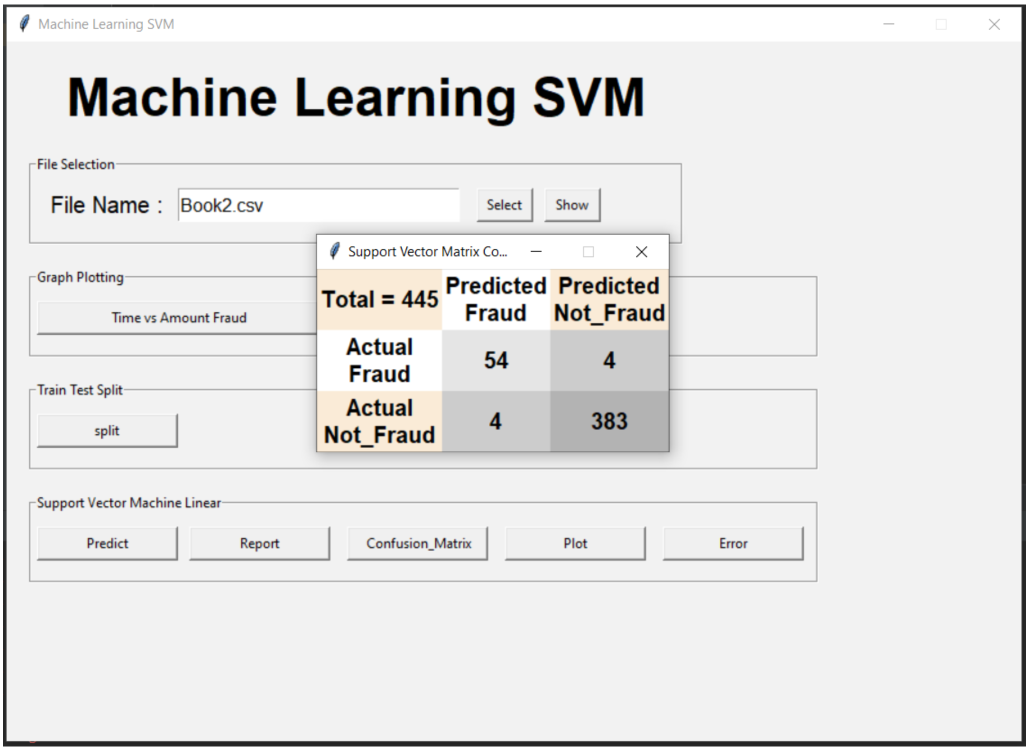


Classification Report-



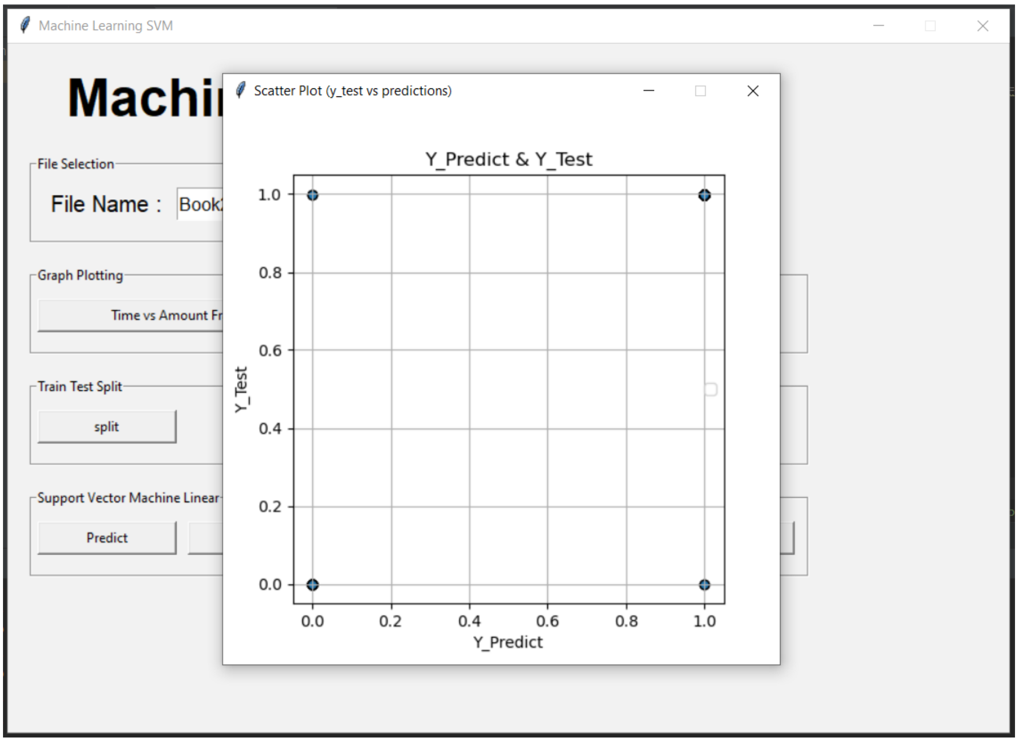
	precision	recall	f1-score	support
Fraud	0.93	0.93	0.93	58
Not_Fraud	0.99	0.99	0.99	387
accuracy			0.98	445
macro avg	0.96	0.96	0.96	445
weighted avg	0.98	0.98	0.98	445

Confusion Matrix:-

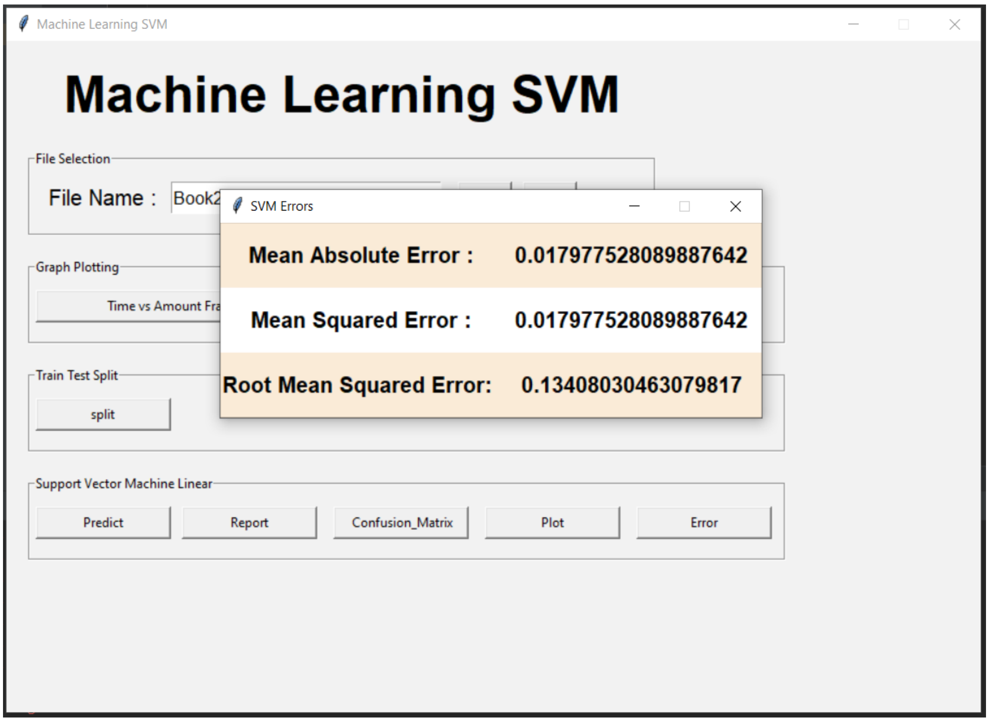


Scatter Plot:-

Graph Y_pred VS Y_test:-



SVM Errors-



Terminal Output-

```
C:\Users\SHUBHAM\PycharmProject\newProj\venv\Scripts\python.exe
"C:/Users/SHUBHAM/PycharmProject/newProj/spam_classifier (1).py"
       Time        V1         V2         V3  ...       V27       V28  Amount  Class
257   11080  -2.125490   5.973556 -11.034727  ...  2.119749  1.108933    1.00  Fraud
275   11092   0.378275   3.914797  -5.726872  ...  0.527938  0.411910    1.00  Fraud
297   11131  -1.426623   4.141986  -9.804103  ...  1.977258  0.711607    1.00  Fraud
578   11629  -3.891192   7.098916 -11.426467  ...  1.881529  0.875260    1.00  Fraud
580   11635   0.919137   4.199633  -7.535607  ...  0.627393  0.157851    1.00  Fraud
...     ...        ...        ...        ...  ...       ...       ...     ...    ...
2217  96135  -1.952933   3.541385  -1.310561  ...  0.532897  0.357892   18.96  Fraud
2218  96291  -3.552173   5.426461  -3.731810  ...  0.023025  0.164741   33.59  Fraud
2219  96717  -3.705856   4.107873  -3.803656  ... -0.315484 -0.097223    1.00  Fraud
2220  97121 -17.976266  12.864989 -19.575066  ... -3.255981 -0.538963    8.64  Fraud
```

```
2221  97235 -17.537592  12.352519 -20.134613  ... -3.838198 -0.802564    9.82  Fraud

[305 rows x 31 columns]=Fraud Data
2222
---------------------------------------------Dataset Values------------------------------------------------
   Time        V1        V2        V3  ...       V27       V28  Amount      Class
0  11001  1.181752 -0.178387  0.715340  ... -0.102278 -0.004942   39.00  Not_Fraud
1  11001 -0.537023  0.723065  1.532169  ...  0.257524  0.179779   15.95  Not_Fraud
2  11001 -0.339007  0.017453  1.487669  ... -0.064727 -0.011791   48.06  Not_Fraud
3  11002  1.305037 -0.083314  0.501800  ... -0.096176 -0.004423   15.95  Not_Fraud
4  11002 -2.103931 -0.388687  1.335634  ...  0.148958  0.001092   39.00  Not_Fraud

[5 rows x 31 columns]
Information about X dataframe:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2222 entries, 0 to 2221
Data columns (total 30 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Time    2222 non-null   int64
 1   V1      2222 non-null   float64
 2   V2      2222 non-null   float64
 3   V3      2222 non-null   float64
 4   V4      2222 non-null   float64
 5   V5      2222 non-null   float64
```

6 V6 2222 non-null float64

7 V7 2222 non-null float64

8 V8 2222 non-null float64

9 V9 2222 non-null float64

10 V10 2222 non-null float64

11 V11 2222 non-null float64

12 V12 2222 non-null float64

13 V13 2222 non-null float64

14 V14 2222 non-null float64

15 V15 2222 non-null float64

16 V16 2222 non-null float64

17 V17 2222 non-null float64

18 V18 2222 non-null float64

19 V19 2222 non-null float64

20 V20 2222 non-null float64

21 V21 2222 non-null float64

22 V22 2222 non-null float64

23 V23 2222 non-null float64

24 V24 2222 non-null float64

25 V25 2222 non-null float64

26 V26 2222 non-null float64

27 V27 2222 non-null float64

28 V28 2222 non-null float64

29 Amount 2222 non-null float64

dtypes: float64(29), int64(1)

memory usage: 520.8 KB

None

Describing the data

[1 1 1 ... 0 0 0]

```
            V1        V2         V3  ...       V27       V28  Amount
1661  1.280171  0.291302   0.275947  ... -0.040525  0.000535    1.29
259   1.372983 -0.382354   0.064602  ...  0.003622  0.020262   39.00
1990 -3.896583  4.518355  -4.454027  ...  0.635789  0.501050    4.56
797   1.184367 -0.033174   0.747380  ... -0.070607 -0.008811   15.95
632   1.311948 -0.162217  -0.677790  ... -0.062374 -0.029138   14.95
...        ...       ...        ...  ...       ...       ...     ...
2190 -6.185857  7.102985 -13.030455  ... -0.203917  0.398927   44.90
1940 -3.146600 -4.162695   2.002792  ...  0.141391  0.301709   11.85
825   1.162377 -0.119161   0.856811  ... -0.078824 -0.008585   14.95
1632  1.171382 -0.347695   1.137520  ...  0.058442  0.021713   11.85
1742 -0.044275  0.862166   2.947061  ...  0.164526  0.177120   18.24
```

[1777 rows x 29 columns]=Training Dataset

```
            V1        V2         V3  ...       V27       V28  Amount
711  -2.097094 -0.503764   0.481407  ... -0.325427 -0.144792   48.00
1797  1.019145  0.685685   0.282878  ...  0.007344 -0.006755    4.95
1884  1.090666  0.336109  -0.072621  ... -0.050001  0.022698   99.99
1071  1.239091 -0.040335   0.861176  ... -0.099424  0.008193   18.22
538   1.201082  0.361576   0.303518  ... -0.021048  0.002418   18.00
```

```
...         ...       ...        ...  ...       ...       ...     ...
2015 -10.940739  6.261586 -14.182339  ...  0.062293 -0.439770   45.49
1363   1.077549  0.389295   1.806577  ...  0.018941  0.036979   15.21
1838   0.639868 -0.252632   0.992716  ... -0.050154  0.058896  231.30
1045  -3.131633  2.985144   1.180058  ...  0.729349 -0.353849    2.31
1598   1.205742  0.519267   0.899012  ...  0.000024  0.028761    1.00

[445 rows x 29 columns]=Testing Dataset
Accuracy is= 99.71862689926843
==============================================================================
predictions is= [1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1
 1 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1
 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1
 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1
 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1
 1]
```

['   Fraud   0.93   0.93   0.93   58\n   Not_Fraud   0.99   0.99   0.99   387', '   accuracy   0.98   445\n   macro avg   0.96   0.96   0.96   445\nweighted avg   0.98   0.98   0.98   445\n']

['Fraud', 'Not_Fraud']

Chapter 5: Conclusion

After running the Support Vector Machine Classifier on the Credit Card Fraud detection data set, we can conclude that it gives highest accuracy from size of the data set and the attributes taken into consideration for prediction calculation.

a) References: -

https://www.kaggle.com/pierra/credit-card-dataset-svm-classification

https://docs.python.org/3/library/tk.html