PROJECT REPORT

ON

CREDIT CARD FRAUD DETECTION

(UNDER THE PARTIAL FULFILLMENT OF THE UNIVERSITY

FOR THE COURSE OF T.Y.BSC

COMPUTER SCIENCE)

DESIGNED AND DEVELOPED

BY

SHILPA SUBHASH TIWARI

GUIDED BY

Ms. BHOOMIKA PANSARE

DEPARTMENT OF COMPUTER SCIENCE

PARLE TILAK VIDYALAYA ASSOCIATION’S

MULUND COLLEGE OF COMMERCE S.N.ROAD,

MULUND (WEST) MUMBAI-80

UNIVERSITY OF MUMBAI

2022-23

**ACKNOWLEDGEMENT**

I have a great pleasure in representing this project report entitled “Credit Card Fraud Detection” and I grab this opportunity to convey my immense regards towards all the distinguished people who have their valuable contribution in the hour of need.

I like to extend my gratitude to our beloved Principal Dr. Sonali Pednekar for her timely and prestigious guidance.

I take this opportunity to thank Dr. Reena Nagda, Coordinator of the Department and all the faculty members of the Department of Computer Science of Mulund College of Commerce, for giving me an opportunity to complete this project and the most needed guidance throughout the duration of the programme.

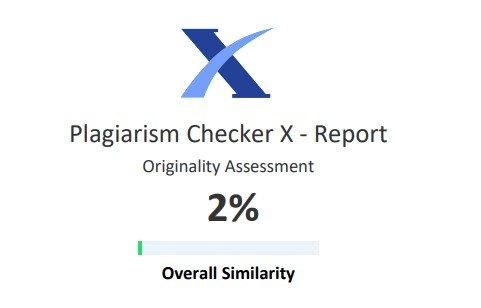
I am extremely grateful to my project guide Ms. Bhoomika Pansare for her valuable guidance and necessary support during each phase of the project. She was the source of continuous encouragement as each milestone was crossed.

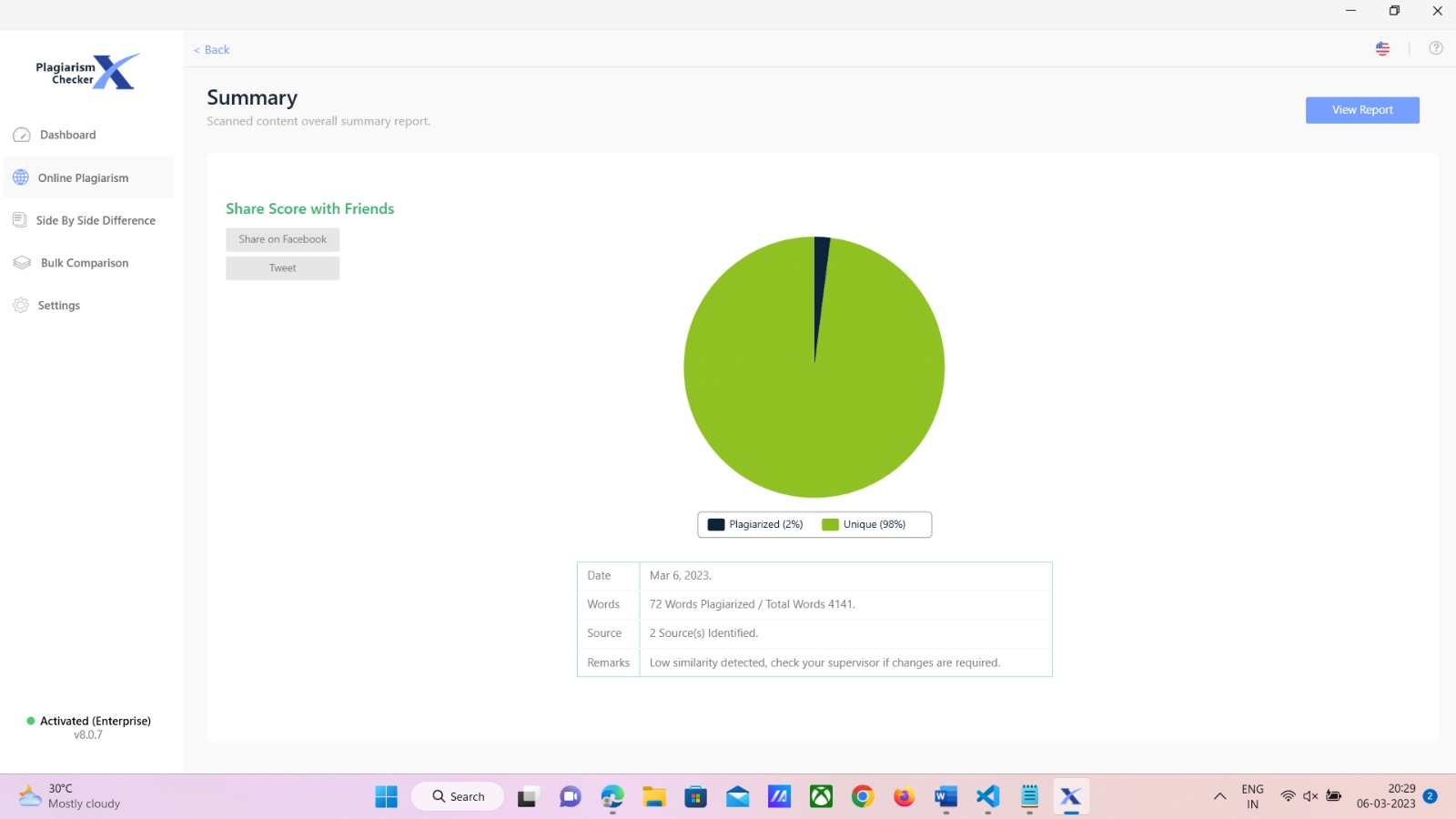
A special thanks to the University of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Finally, I also owe to my fellow friends who have been a constant source of help to solve the problems that cropped up during the project development process.

Shilpa Tiwari

PLAGIARISM

****



INDEX

|  |  |  |
| --- | --- | --- |
| Sr. No. | TOPIC | Page No. |
| 1. | Title | 1 |
| 2. | Introduction | 2 |
| 3. | Requirement Specification | 3 |
| 4. | System Design Details | 4 |
| 5. | System Implementation | 11 |
| 6. | Results | 18 |
| 7. | Conclusion and Future Enhancements | 20 |
| 8. | References | 21 |
| 9. | Annexure | 22 |

1.Title

Credit Card Fraud

# 2. Introduction

The “CREDIT CARD FRAUD DETECTION SYSTEM'' is a fraud detection system. Fraud detection is the set of processes analyses which allows businesses to identify and prevent unauthorized financial activity. This can include fraudulent credit card transactions. These frauds usually occur in credit card-based purchase transactions or any other credit card transaction.

**2.1 Objective of the project:**

The aim of the project is to predict whether a credit card transaction is fraudulent or not, based on the transaction amount, location and other transaction related data. It aims to track down credit card transaction data, which is done by detecting anomalies in the transaction data.

**2.2 Description of the current system:-**

* Storing information of credit card users.
* Check validity of information provided by the user.
* Storing information of users according to their id.
* Generating reports for different user ids.

**2.3 Limitations of the current system:-**

* The credit card fraud detection system is done manually.
* The current system is time consuming and also very costly, because it involves a lot of paperwork.
* Manually handling the system was a very difficult task but now-a-days computerization makes it easy to work.
* As the system is manual there are a lot of chances of human errors. These can
* cause errors in calculating mechanisms or maintaining customer details.

**2.4 Advantage of proposed system:**

* It provides wide range of certain criteria in each window the client is working for
* better and quicker solution.
* It maintains a report for all criteria and transactions.
* This system can run on any windows operating system.

# 3. Requirement Specification

**3.1 Software Requirement:**

* Python 3.0 or higher
* Windows operating system
* Microsoft Visual Studio

**3.2 Hardware Requirement:**

* **Processor** – Intel Pentium G4560 3.50GHz. Any 7th generation processor will be sufficient
* **RAM** – Minimum 4GB
* **Disk Space** – Minimum 2GB

**3.3 Fact-finding questions:-**

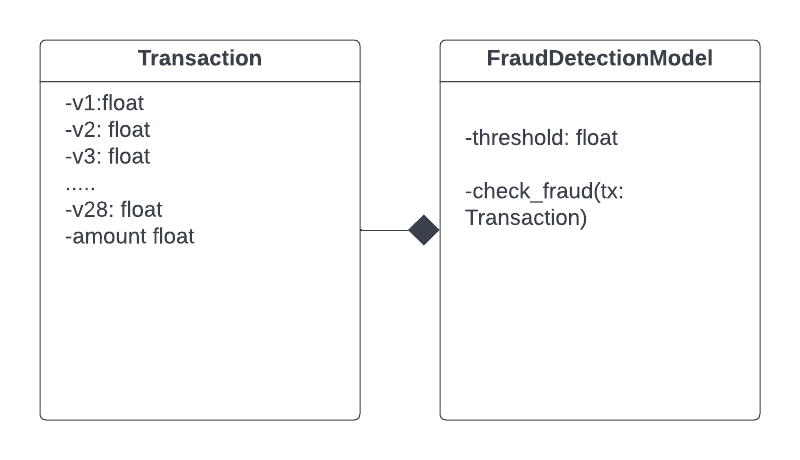
* Why might a credit card fraud detection system be useful?
* What are the potential applications and uses of a credit card fraud detection system and how might it be integrated into existing systems and processes?
* What level of expertise is required to use the system?

# 4. System Design Details

**4.1 Event Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr No.** | **Events** | **Trigger** | **Source** | **Activity** | **Response** |
| 1. | Login | btn\_login | User | Opens the login page | Login the user |
| 2. | register | btn\_register | User | Opens the registration page | Rdirects to registered successfully window |
|  | submit | btn\_submit | User | Submits the entered values | Accepts the value and evaluates result. |

**4.2 Class Diagram:**

figure1: Class Diagram

**4.2 Use Case Diagram:**

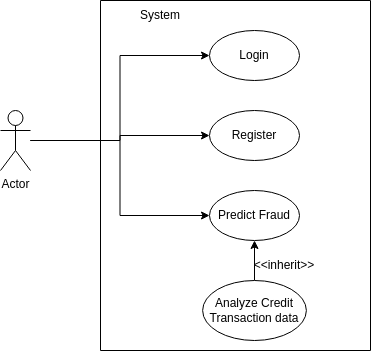
****

figure2: Use Case Diagram

**4.3 Sequence Diagram:**

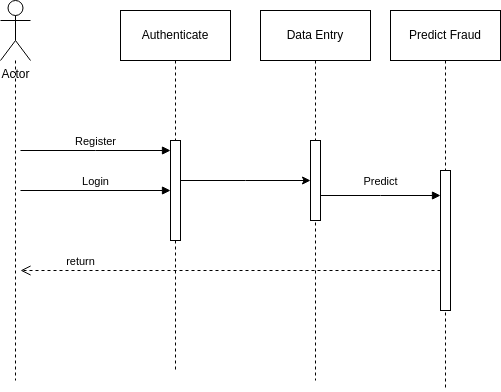


figure3: Sequence Diagram

**4.4 Activity Diagram:**

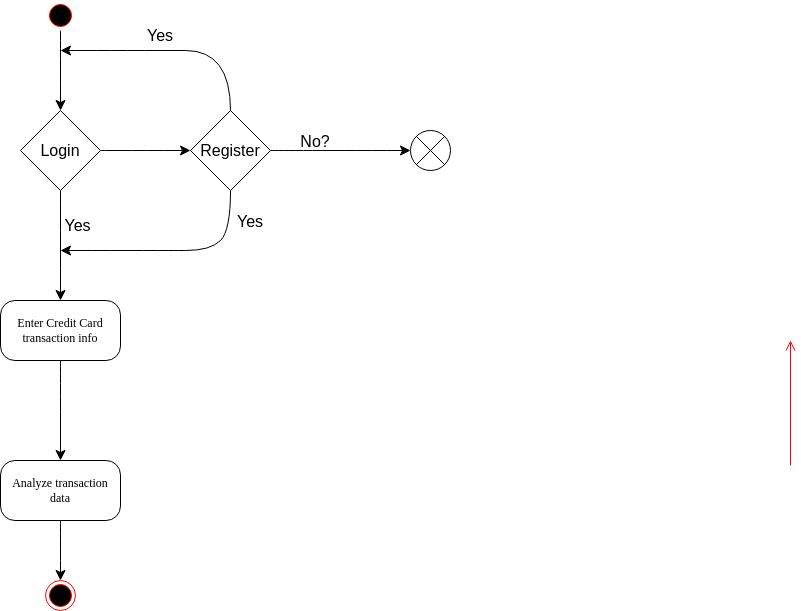
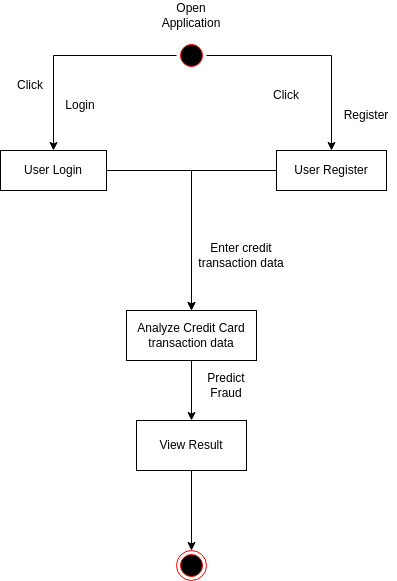
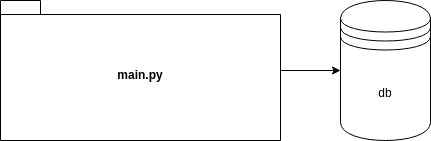


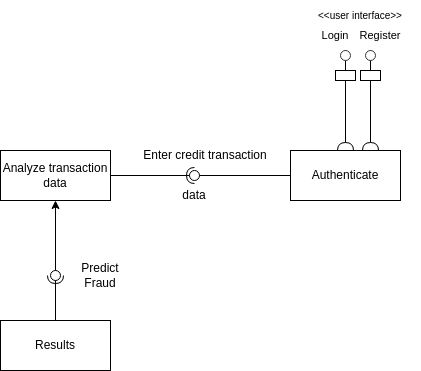
figure 4: Activity Diagram

**4.5 State Diagram:**

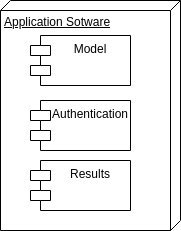
****

**4.6 Package Diagram:**

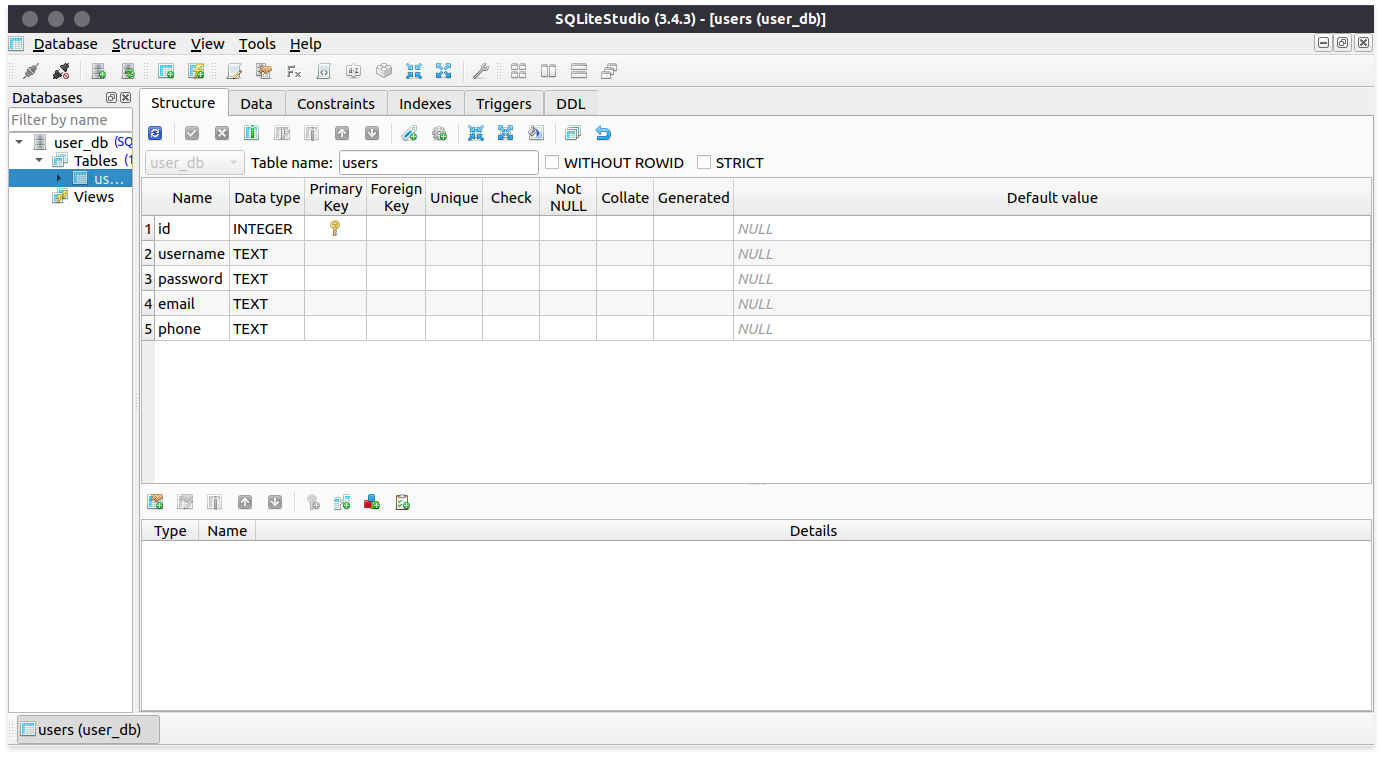
****

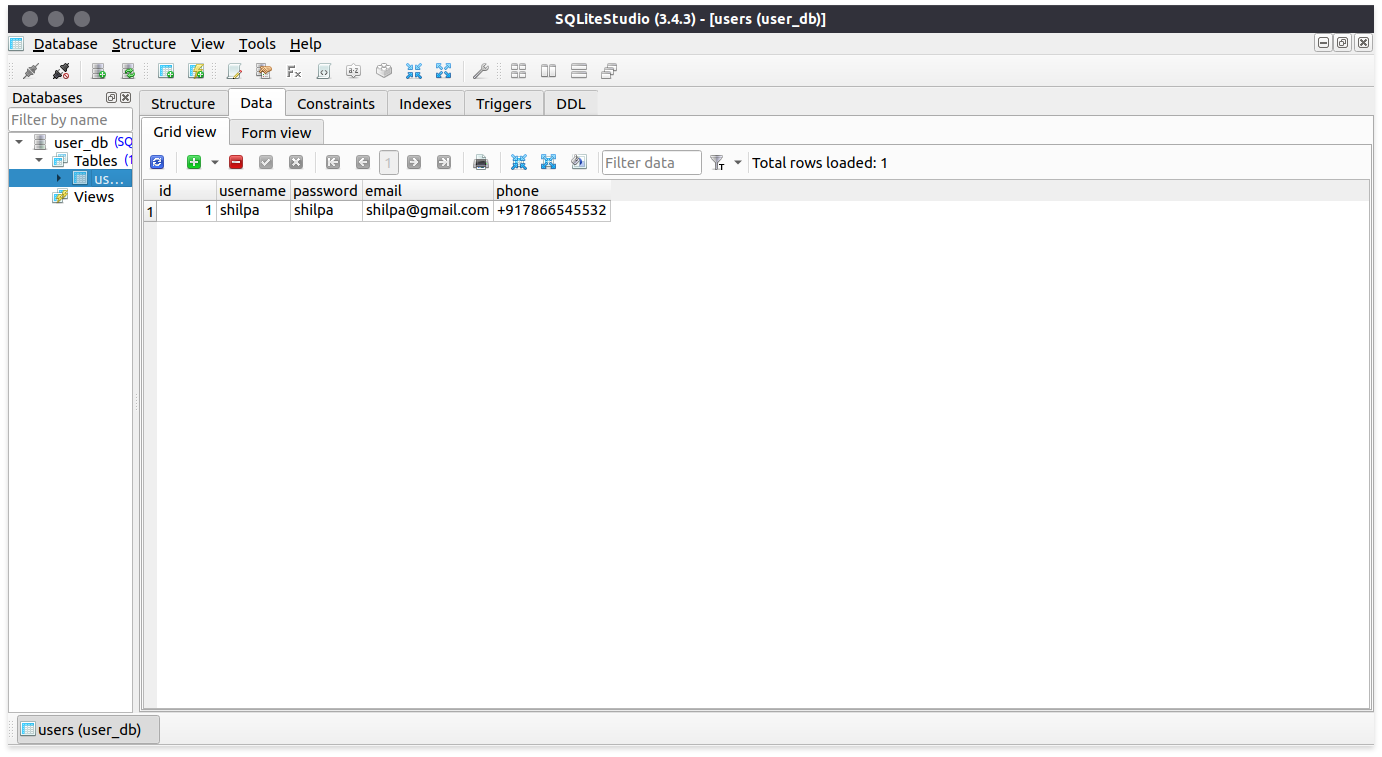
**4.7 Component Diagram:**

**4.8 Deployment Diagram:**

****

**4.9. Database Design:**





**5. System Implementation**

Creditsystem.ipynb

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# loading the dataset to a Pandas dataframe

credit\_card\_data = pd.read\_csv('/content/creditcard.csv')

# first 5 rows of the dataset

credit\_card\_data.head()

#last 5 rows of the dataset

credit\_card\_data.tail()

#dataset informations

credit\_card\_data.info()

credit\_card\_data.isnull().sum()

#Distribution of the legit transaction and fraudulent transaction

credit\_card\_data['Class'].value\_counts()

#Check how many datapoints we have

credit\_card\_data.shape

#This dataset is highly unbalanced

#0 --> Normal transaction 1--> fraudulent transaction

#separating data for analysis

legit = credit\_card\_data[credit\_card\_data.Class == 0]

fraud = credit\_card\_data[credit\_card\_data.Class == 1]

print(legit.shape)

print(fraud.shape)

(284315, 31)

(492, 31)

#statistical measures of the data

legit.Amount.describe()

fraud.Amount.describe()

#compare the values for both legit and fraud transaction classes

credit\_card\_data.groupby('Class').mean()

#Under-Sampling

#Buid a sampl dataset containing similar distribution of normal transaction and the fraudulent transactions

#Number of fraudulent transactions = 492

legit\_sample = legit.sample(n=492)

#Concatinating two dataframes(legit\_sample and fraud transaction)

new\_dataset = pd.concat([legit\_sample, fraud], axis=0)

new\_dataset.head()

new\_dataset.tail()

new\_dataset['Class'].value\_counts()

new\_dataset.groupby('Class').mean()

#Splitting the data into features and targets

X = new\_dataset.drop(columns='Class', axis=1)

Y = new\_dataset['Class']

print(X)

print(Y)

#Split the data into Training and Testing data

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size= 0.2, stratify= Y, random\_state=2)

print(X.shape, X\_train.shape,X\_test.shape)

#Model Training

#Logistic Regression Model

model = LogisticRegression()

#training the logiatic regression model with training data

model.fit(X\_train, Y\_train)

import pickle

pkl\_filename = "lr\_model.pkl"

with open(pkl\_filename,'wb') as file:

pickle.dump(model, file)

#accuracy on training data

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

print('Accuracy on trining data: ', training\_data\_accuracy)

Accuracy on trining data: 0.9097839898348158

#Accuracy on test data

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print('Accuracy on test data: ',test\_data\_accuracy)

Accuracy on test data: 0.8984771573604061

main.py

from tkinter import \*

from tkinter import messagebox

import sqlite3

import pickle

import pandas as pd

import sklearn

def predict\_fraud(data):

data = [float(x) for x in data]

df = pd.DataFrame([data], columns=['Time', 'V1','V2','V3', 'V4', 'V5', 'V6', 'V7', 'V8','V9','V10', 'V11', 'V12', 'V13', 'V14',

'V15','V16','V17', 'V18', 'V19', 'V20', 'V21',

'V22','V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount'])

with open('lr\_model.pkl', 'rb') as file:

model =pickle.load(file)

result = model.predict(df)

print(result)

if result[0] == 0:

return "No Fraud"

else:

return "Fraud"

# create the main window

root = Tk()

root.title("Credit Card Fraud Detection")

root.geometry("900x500")

root.configure(bg='#333333')

def init\_db():

conn = sqlite3.connect('user\_db.sqlite')

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS users

(id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT,

password TEXT,

email TEXT,

phone TEXT)''')

conn.commit()

conn.close()

# initialize the database

init\_db()

def credit\_entry\_window():

# create a new window

# data\_entry = Toplevel(root)

data\_entry = Tk()

data\_entry.title("Credit Entry Form")

data\_entry.geometry("1000x600")

data\_entry.configure(bg='#333333')

v\_entry\_list = []

# create a label and entry for the time

time\_label = Label(data\_entry, fg='#FFFFFF',bg='#333333', text="Time:", font=("Arial",13))

time\_label.grid(row=0, column=0, pady= 10)

time\_entry = Entry(data\_entry,font=("Arial",13))

time\_entry.grid(row=0, column=1)

# for i in range(1, 29):

# v\_label = Label(data\_entry, text="V" + str(i) + ":",font=("Arial",13), fg='#FFFFFF',bg='#333333')

# v\_label.grid(row=i, column=0, pady= 10)

# v\_entry = Entry(data\_entry,font=("Arial",13))

# v\_entry.grid(row=i, column=1, pady=10)

# v\_entry\_list.append(v\_entry)

row = 1

for i in range(1, 29, 3):

v\_label = Label(data\_entry, text="V" + str(i) + ":",font=("Arial",13), fg='#FFFFFF',bg='#333333')

v\_label.grid(row=row, column=0, pady= 10)

v\_entry = Entry(data\_entry,font=("Arial",13))

v\_entry.grid(row=row, column=1, pady=10)

v\_label\_2 = Label(data\_entry, text="V" + str(i+1) + ":",font=("Arial",13), fg='#FFFFFF',bg='#333333')

v\_label\_2.grid(row=row, column=3, pady= 10)

v\_entry\_2 = Entry(data\_entry,font=("Arial",13))

v\_entry\_2.grid(row=row, column=4, pady=10)

v\_label\_3 = Label(data\_entry, text="V" + str(i+2) + ":",font=("Arial",13), fg='#FFFFFF',bg='#333333')

v\_label\_3.grid(row=row, column=5, pady= 10)

v\_entry\_3 = Entry(data\_entry,font=("Arial",13))

v\_entry\_3.grid(row=row, column=6, pady=10)

v\_entry\_list.append(v\_entry)

v\_entry\_list.append(v\_entry\_2)

v\_entry\_list.append(v\_entry\_3)

row += 1

# create a label and entry for the amount

amount\_label = Label(data\_entry, text="Amount:",font=("Arial",13),fg='#FFFFFF',bg='#333333')

amount\_label.grid(row=0, column=3, pady= 10)

amount\_entry = Entry(data\_entry,font=("Arial",13))

amount\_entry.grid(row=0, column=4, pady=10)

def save\_data():

data = [time\_entry.get()]

for i in range(1, 29):

data.append(v\_entry\_list[i-1].get())

data.append(amount\_entry.get())

result = predict\_fraud(data)

# show a message box to confirm the data has been saved

messagebox.showinfo("Data Saved", "Your data has been saved." + str(result))

save\_button = Button(data\_entry, text="Submit",bg='#FF3399', fg='#FFFFFF', font=("Arial",16), command=save\_data)

save\_button.grid(row=33, column=4)

# create the login/register window

def login\_window():

# destroy the main window

root.destroy()

# create a new window

login = Tk()

login.title("Login/Register Form")

login.geometry("900x500")

login.configure(bg='#333333')

# create a label and entry for the username

username\_label = Label(login, text="Username:",bg='#333333', fg='#FFFFFF', font=("Arial",16 ))

username\_label.grid(row=1, column=0)

username\_entry = Entry(login, font=("Arial",16))

username\_entry.grid(row=1, column=1, pady=20)

# create a label and entry for the password

password\_label = Label(login, text="Password:", bg='#333333', fg='#FFFFFF', font=("Arial",16))

password\_label.grid(row=2, column=0)

password\_entry = Entry(login, show="\*", font=("Arial",16))

password\_entry.grid(row=2, column=1, pady=20)

# create a function to check if the login is valid

def login\_check():

# check if the username and password match

conn = sqlite3.connect('user\_db.sqlite')

cursor = conn.cursor()

cursor.execute("SELECT \* FROM users WHERE username=? AND password=?", (username\_entry.get(), password\_entry.get()))

user = cursor.fetchone()

conn.close()

if user is not None:

messagebox.showinfo("Login Successful", "Welcome back, " + user[1] + "!")

login.destroy()

credit\_entry\_window()

else:

messagebox.showerror("Login Failed", "Incorrect username or password.")

# create a login button

login\_button = Button(login, text="Login",bg='#FF3399', fg='#FFFFFF', font=("Arial",16), command=login\_check)

login\_button.grid(row=3, column=0, pady=30, columnspan=1)

def register\_window():

# create a new window

register = Toplevel(login)

register.title("Register Form")

register.configure(bg='#333333')

register.geometry("900x500")

# create a label and entry for the username

new\_username\_label = Label(register, text="New Username:",bg='#333333',fg="#FFFFFF",font=("Arial",16))

new\_username\_label.grid(row=1, column=0)

new\_username\_entry = Entry(register, font=("Arial",16))

new\_username\_entry.grid(row=1, column=1,pady=20)

# create a label and entry for the password

new\_password\_label = Label(register, text="New Password:",bg='#333333',fg="#FFFFFF", font=("Arial",16))

new\_password\_label.grid(row=2, column=0)

new\_password\_entry = Entry(register, show="\*", font=("Arial",16))

new\_password\_entry.grid(row=2, column=1, pady=20)

# create a label and entry for the email

new\_email\_label = Label(register, text="New Email:", bg='#333333', fg="#FFFFFF", font=("Arial",16))

new\_email\_label.grid(row=3, column=0)

new\_email\_entry = Entry(register, font=("Arial",16))

new\_email\_entry.grid(row=3, column=1, pady=20)

# create a label and entry for the phone number

new\_phone\_label = Label(register, text="New Phone Number:", bg='#333333', fg="#FFFFFF", font=("Arial",16))

new\_phone\_label.grid(row=4, column=0)

new\_phone\_entry = Entry(register, font=("Arial",16))

new\_phone\_entry.grid(row=4, column=1, pady=20)

def add\_user():

conn = sqlite3.connect('user\_db.sqlite')

cursor = conn.cursor()

cursor.execute("INSERT INTO users (username, password, email, phone) VALUES (?, ?, ?, ?)", (new\_username\_entry.get(), new\_password\_entry.get(), new\_email\_entry.get(), new\_phone\_entry.get()))

conn.commit()

conn.close()

messagebox.showinfo("Registration Successful", "Your account has been created!")

register.destroy()

register\_button = Button(register, text="Register", bg='#FF3399', fg='#FFFFFF', font=("Arial",16),command=add\_user)

register\_button.grid(row=6, column=0, columnspan=2, pady=30)

register\_button = Button(login, text="Register", bg='#FF3399', fg='#FFFFFF', font=("Arial",16), command=register\_window)

register\_button.grid(row=3, column=1, columnspan=3, pady=40)

# run the window

login.mainloop()

# create a label for the main window

welcome\_label = Label(root, text="Welcome to the CCF System", bg='#333333', fg='#FF3399', font=("Arial",30))

welcome\_label.pack()

# create a button to open the login/register window

login\_button = Button(root, text="Login/Register", fg='#FF3399', font=("Arial",16),command=login\_window, activebackground="grey")

login\_button.pack()

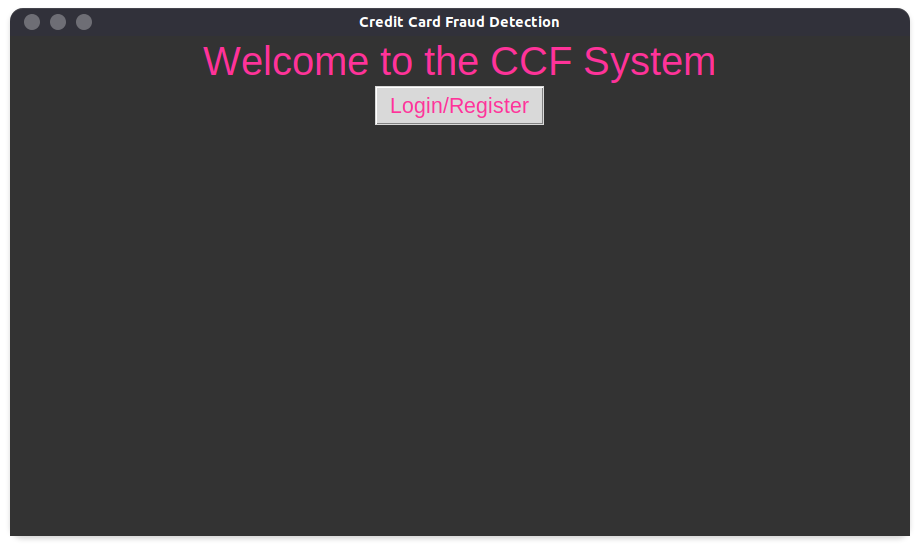
# run the main window

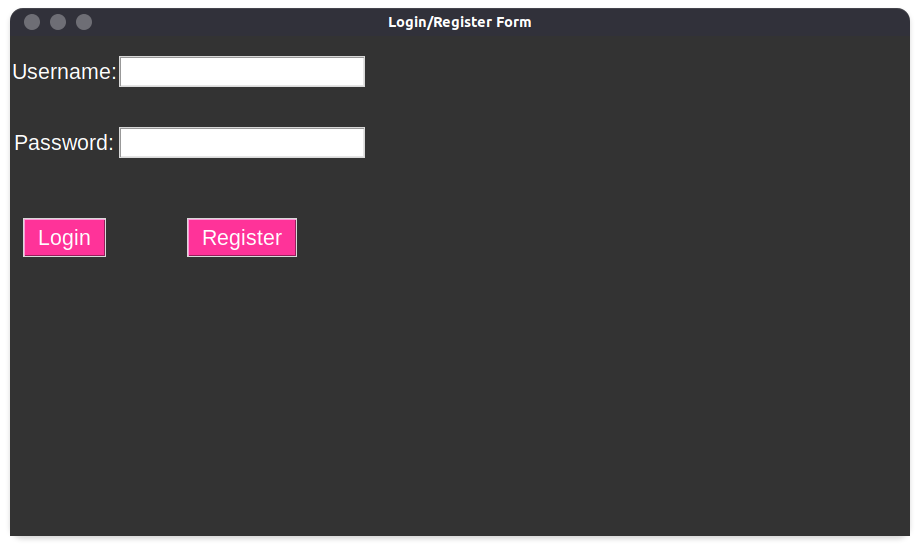
root.mainloop()

**6.Results**

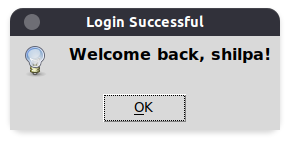
**Screenshots:**

**User Interface:**

****

**L****ogin:**

**Login Successful:**

****

**Transaction detail:**

****

**Fraud Detection:**

****

**7.Conclusion and Future Enhancements**

**Conclusion:**

* The system provides a unique and innovative approach to detect credit card transaction frauds.
* Overall, the project represents an exciting and promising application of machine learning technology to the field of mathematics, and it will be exciting to see how it continues to evolve and develop in the future.

**Future Enhancements:**

1. Integration with Machine Learning: Currently, the system uses a set of pre-defined rules to analyze transaction data for possible fraud. In the future, the system could be enhanced by incorporating machine learning algorithms to improve the accuracy and speed of fraud detection.

2. Real-Time Monitoring: The current system operates on a batch processing model, meaning that transactions are analyzed in batches at periodic intervals. A future enhancement could involve implementing real-time monitoring of transactions, where transactions are analyzed and processed as they occur.

**8.References**

1. [**https://www.sciencedirect.com/science/article/pii/S2212017314003051**](https://www.sciencedirect.com/science/article/pii/S2212017314003051)
2. [**https://www.sciencedirect.com/science/article/pii/S2212017314003051**](https://www.sciencedirect.com/science/article/pii/S2212017314003051)
3. [**https://www.datacamp.com/community/tutorials/credit-card-fraud-detection-python**](https://www.datacamp.com/community/tutorials/credit-card-fraud-detection-python)
4. [**https://www.analyticsvidhya.com/blog/2021/02/credit-card-fraud-detection-a-practical-guide-in-2021/**](https://www.analyticsvidhya.com/blog/2021/02/credit-card-fraud-detection-a-practical-guide-in-2021/)

**9.Annexure**

**Figure List:**

|  |  |  |
| --- | --- | --- |
| Figure Number | Name of the figure | Page number |
| 1 | Class Diagram | 6 |
| 2 | Use Case Diagram | 6 |
| 3 | Sequence Diagram | 7 |
| 4 | Activity Diagram | 7 |
| 5 | State Diagram | 8 |
| 6 | Package Diagram | 8 |
| 7 | Component Diagram | 9 |
| 8 | Deployment Diagram | 9 |

**Table List:**

|  |  |  |
| --- | --- | --- |
| Table Number | Name of the table | Page number |
| 1 | Event Table | 5 |
| 2 | History Table Structure | 10 |