import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

sns.set(style = "whitegrid")

df = pd.read\_csv("C:\\Users\\USER\\Downloads\\AIML Dataset.csv")

df.head()

df.info()

df.columns

df["isFraud"].value\_counts()

df["isFlaggedFraud"].value\_counts()

df.isnull().sum().sum()

df.shape[0]

round((df["isFraud"].value\_counts()[1]/df.shape[0]) \* 100,2)

df["type"].value\_counts().plot(kind="bar",title = "Transaction Types",color= "skyblue")

plt.xlabel("Transaction Type")

plt.ylabel("Count")

plt.show()

fraud\_by\_type = df.groupby("type")["isFraud"].mean().sort\_values(ascending = False)

fraud\_by\_type.plot(kind= "bar", title = "Fraud Rate by Type",color = "salmon")

plt.ylabel("Fraud Rate")

plt.show()

df["amount"].describe().astype(int)

sns.histplot(np.log1p(df["amount"]),bins = 100, kde = True, color = "green")

plt.title("Transaction Amount Distribution(log scale)")

plt.xlabel("log(Amount + 1)")

plt.show()

sns.boxplot(data = df[df["amount"] < 50000], x = "isFraud", y = "amount")

plt.title("Amount vs isFraud(Filtered under 50k)")

plt.show()

df["balanceDiffOrig"] = df["oldbalanceOrg"] - df["newbalanceOrig"]

df["balanceDiffDest"] = df["newbalanceDest"] - df["oldbalanceDest"]

(df["balanceDiffOrig"] < 0).sum()

df.head(2)

fraud\_per\_step = df[df["isFraud"] == 1]["step"].value\_counts().sort\_index()

plt.plot(fraud\_per\_step.index,fraud\_per\_step.values,label = "Frauds per step")

plt.xlabel("step(Time)")

plt.ylabel("Number of Frauds")

plt.title("Frauds Over Time")

plt.grid(True)

plt.show()

df.drop(columns = "step", inplace = True)

df.head()

top\_senders = df["nameOrig"].value\_counts().head(10)

top\_senders

top\_receivers = df["nameDest"].value\_counts().head(10)

top\_receivers

fraud\_users = df[df["isFraud"] == 1]["nameOrig"].value\_counts().head(10)

fraud\_users

fraud\_types = df[df["type"].isin(["TRANSFER","CASH\_OUT"])]

fraud\_types["type"].value\_counts()

sns.countplot(data = fraud\_types, x = "type", hue = "isFraud")

plt.title("Fraud Distribution in Transfer and Cash\_out")

plt.show()

corr = df[["amount","oldbalanceOrg","newbalanceOrig","oldbalanceDest","newbalanceDest","isFraud"]].corr()

corr

sns.heatmap(corr,annot = True, cmap = "coolwarm",fmt = ".2f")

plt.title("Correlation Matrix")

plt.show()

Zero\_after\_transfer = df[(df["oldbalanceOrg"] > 0) &

(df["newbalanceOrig"] == 0)&

(df["type"].isin(["TRANSFER","CASH\_OUT"]))]

len(Zero\_after\_transfer)

Zero\_after\_transfer.head()

df["isFraud"].value\_counts()

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report,confusion\_matrix

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

df.head()

df\_model = df.drop(["nameOrig","nameDest","isFlaggedFraud"], axis = 1)

df\_model.head()

categorical = ["type"]

numeric = ["amount","oldbalanceOrg","newbalanceOrig","oldbalanceDest","newbalanceDest"]

y = df\_model["isFraud"]

X = df\_model.drop("isFraud", axis = 1)

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size = 0.3, stratify = y)

preprocessor = ColumnTransformer(

transformers =[

("num",StandardScaler(),numeric),

("cat",OneHotEncoder(drop="first"),categorical)

],

remainder = "drop"

)

pipeline = Pipeline([("prep",preprocessor),

("clf",LogisticRegression(class\_weight = "balanced", max\_iter = 1000))

])

pipeline.fit(X\_train, y\_train)

y\_pred = pipeline.predict(X\_test)

print(classification\_report(y\_test,y\_pred))

confusion\_matrix(y\_test,y\_pred)

pipeline.score(X\_test,y\_test) \* 100

import joblib

joblib.dump(pipeline,"fraud\_detection\_pipeline.pkl")

import streamlit as st

import pandas as pd

import joblib

model = joblib.load("fraud\_detection\_pipeline.pkl")

st.title("Fraud Detection Prediction App")

st.markdown("Please enter the transcation details and use the predict button")

st.divider()

transaction\_type = st.selectbox("Transaction Type",["PAYMENT","TRANSFER","CASH\_OUT","DEPOSIT"])

amount = st.number\_input("Amount",min\_value = 0.0, value = 1000.0)

oldbalanceOrg = st.number\_input("Old Balance(Sender)", min\_value = 0.0, value = 10000.0)

newbalanceOrig = st.number\_input("New Balance(Sender)", min\_value = 0.0, value = 9000.0)

oldbalanceDest = st.number\_input("Old Balance(Receiver)", min\_value = 0.0,value = 0.0)

newbalanceDest = st.number\_input("New Balance(Receiver)", min\_value = 0.0, value = 0.0)

if st.button("Predict"):

    input\_data = pd.DataFrame([{

        "type": transaction\_type,

        "amount": amount,

        "oldbalanceOrg" : oldbalanceOrg,

        "newbalanceOrig": newbalanceOrig,

        "oldbalanceDest": oldbalanceDest,

        "newbalanceDest": newbalanceDest

    }])

prediction = model.predict(input\_data)[0]

st.subheader(f"Prediction:'{int(prediction)}'")

if prediction == 1:

    st.error("This transaction can be fraud")

else:

    st.sucess("This transaction looks like it is not a fraud")