Source code : Grauding Transactions with AI-Powered Credit Card fraud Detection and Prevention

import pandas as pd

# Load data

data = pd.read\_csv("cards\_data.csv")

# Drop sensitive identifiers

data = data.drop(columns=["card\_number", "cvv", "expires", "acct\_open\_date"])

import matplotlib.pyplot as plt

import seaborn as sns

# Quick overview

print(data.info())

print(data.describe())

print(data.isnull().sum())

# Bar plot for categorical features

categorical\_cols = data.select\_dtypes(include='object').columns.tolist()

for col in categorical\_cols:

    plt.figure(figsize=(5, 3))

    sns.countplot(data[col])

    plt.title(f"Distribution of {col}")

    plt.xticks(rotation=45)

    plt.tight\_layout()

    plt.show()

# Histogram for numerical features

numeric\_cols = data.select\_dtypes(include=['int64', 'float64']).columns.tolist()

data[numeric\_cols].hist(figsize=(12, 8), bins=20)

plt.suptitle("Distributions of Numeric Features")

plt.tight\_layout()

plt.show()

# Correlation heatmap

plt.figure(figsize=(8, 6))

sns.heatmap(data[numeric\_cols].corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Between Numeric Features")

plt.tight\_layout()

plt.show()

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.impute import SimpleImputer

from sklearn.compose import ColumnTransformer

# Define columns again

numeric\_cols = data.select\_dtypes(include=['int64', 'float64']).columns.tolist()

categorical\_cols = data.select\_dtypes(include='object').columns.tolist()

# Pipelines

numeric\_pipeline = Pipeline([

    ("imputer", SimpleImputer(strategy="mean")),

    ("scaler", StandardScaler())

])

categorical\_pipeline = Pipeline([

    ("imputer", SimpleImputer(strategy="most\_frequent")),

    ("encoder", OneHotEncoder(handle\_unknown="ignore"))

])

# Combine preprocessors

preprocessor = ColumnTransformer([

    ("num", numeric\_pipeline, numeric\_cols),

    ("cat", categorical\_pipeline, categorical\_cols)

])

from sklearn.ensemble import IsolationForest

model\_pipeline = Pipeline([

    ("preprocessing", preprocessor),

    ("isolation\_forest", IsolationForest(n\_estimators=100, contamination=0.05, random\_state=42))

])

# Train model

model\_pipeline.fit(data)

# Predict (-1 = fraud-like, 1 = normal)

preds = model\_pipeline.named\_steps["isolation\_forest"].predict(

    model\_pipeline.named\_steps["preprocessing"].transform(data)

)

# Add to dataset

data["fraud\_prediction"] = preds

data["fraud\_label"] = data["fraud\_prediction"].map({1: "Normal", -1: "Fraud-like"})

from sklearn.decomposition import PCA

# Countplot

plt.figure(figsize=(6, 4))

sns.countplot(x="fraud\_label", data=data, palette="Set2")

plt.title("Fraud Detection Results")

plt.xlabel("Record Type")

plt.ylabel("Count")

plt.tight\_layout()

plt.show()

# PCA for 2D visualization

X\_transformed = model\_pipeline.named\_steps["preprocessing"].transform(data.drop(columns=["fraud\_prediction", "fraud\_label"]))

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_transformed)

plt.figure(figsize=(8, 6))

sns.scatterplot(x=X\_pca[:, 0], y=X\_pca[:, 1], hue=data["fraud\_label"], palette={"Normal": "green", "Fraud-like": "red"}, alpha=0.6)

plt.title("PCA of Card Records with Fraud Detection")

plt.xlabel("PCA Component 1")

plt.ylabel("PCA Component 2")

plt.legend(title="Prediction")

plt.tight\_layout()

plt.show()