import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

dataset=pd.read\_csv(‘C:\\Users\\STUDENT\\Desktop\\diabetes.csv’)

dataset.head()

dataset.shape

dataset.isnull().values.any()

dataset.info()

dataset.describe()

dataset.isnull().sum()

sns.countplot(x = ‘Outcome’,data = dataset)

sns.pairplot(data = dataset, hue = ‘Outcome’)

plt.show()

sns.heatmap(dataset.corr(), annot = True)

plt.show()

dataset\_new = dataset

dataset\_new[[“Glucose”, “BloodPressure”, “SkinThickness”, “Insulin”, “BMI”]] = dataset\_new[[“Glucose”, “BloodPressure”, “SkinThickness”, “Insulin”, “BMI”]].replace(0, np.NaN)

dataset\_new.isnull().sum()

# Check for Missing Values

Missing\_Values = df.isnull().sum()

print(“Missing Values:”)

print(missing\_values)

# Handle missing values (if any)

# For example, fill missing values with the mean of the column

mean\_fill = df.mean()

df.fillna(mean\_fill, inplace=True)

# Check for Duplicate Rows

duplicate\_rows = df[df.duplicated()]

print(“\nDuplicate Rows:”)

print(duplicate\_rows)

# Handle duplicate rows (if any)

# For example, drop duplicate rows

df.drop\_duplicates(inplace=True)

# Step 4: Data Analysis

# Summary Statistics

summary\_stats = df.describe()

print(“\nSummary Statistics:”)

print(summary\_stats)

# Class Distribution (for binary classification problems)

class\_distribution = df[‘Outcome’].value\_counts()

print(“\nClass Distribution:”)

print(class\_distribution)

# Step 5: Data Visualization

sns.pairplot(df, hue=’Outcome’)

plt.show()

# Step 6: Support Vector Machine (SVM) Modeling

# Separate features and target variable

X = df.drop(‘Outcome’, axis=1)

Y = df[‘Outcome’]

# Split the dataset into a training and testing set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Initialize and train the SVM model

Model = SVC(kernel=’linear’, random\_state=42)

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

# Make predictions

Y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

Print(f’Accuracy: {accuracy:.2f}’)

# Classification report and confusion matrix

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

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

sns.heatmap(cm, annot=True, fmt=’d’)

plt.show()