*# Import necessary libraries*

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

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

*# Load the dataset (replace 'diabetes.csv' with your dataset file)*

df = pd.read\_csv('/kaggle/input/diabetes-data-set/diabetes.csv')

*# Step 1: Data Cleaning*

*# 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("**\n**Duplicate Rows:")

print(duplicate\_rows)

*# Handle duplicate rows (if any)*

*# For example, drop duplicate rows*

df.drop\_duplicates(inplace=True)

*# Step 2: Data Analysis*

*# Summary Statistics*

summary\_stats = df.describe()

print("**\n**Summary Statistics:")

print(summary\_stats)

*# Class Distribution (for binary classification problems)*

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

print("**\n**Class Distribution:")

print(class\_distribution)

*# Step 3: 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()

Missing Values:

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

Duplicate Rows:

Empty DataFrame

Columns: [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome]

Index: []

Summary Statistics:

Pregnancies Glucose BloodPressure SkinThickness Insulin \

count 768.000000 768.000000 768.000000 768.000000 768.000000

mean 3.845052 120.894531 69.105469 20.536458 79.799479

std 3.369578 31.972618 19.355807 15.952218 115.244002

min 0.000000 0.000000 0.000000 0.000000 0.000000

25% 1.000000 99.000000 62.000000 0.000000 0.000000

50% 3.000000 117.000000 72.000000 23.000000 30.500000

75% 6.000000 140.250000 80.000000 32.000000 127.250000

max 17.000000 199.000000 122.000000 99.000000 846.000000

BMI DiabetesPedigreeFunction Age Outcome

count 768.000000 768.000000 768.000000 768.000000

mean 31.992578 0.471876 33.240885 0.348958

std 7.884160 0.331329 11.760232 0.476951

min 0.000000 0.078000 21.000000 0.000000

25% 27.300000 0.243750 24.000000 0.000000

50% 32.000000 0.372500 29.000000 0.000000

75% 36.600000 0.626250 41.000000 1.000000

max 67.100000 2.420000 81.000000 1.000000

Class Distribution:

Outcome

0 500

1 268

Name: count, dtype: int64

Accuracy: 0.76

precision recall f1-score support

0 0.81 0.82 0.81 99

1 0.67 0.65 0.66 55

accuracy 0.76 154

macro avg 0.74 0.74 0.74 154

weighted avg 0.76 0.76 0.76 154