**Program Code:**

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)

# 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)

#Data Visualization

Sns.pairplot(df, hue=’Outcome’)

Plt.show()

#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

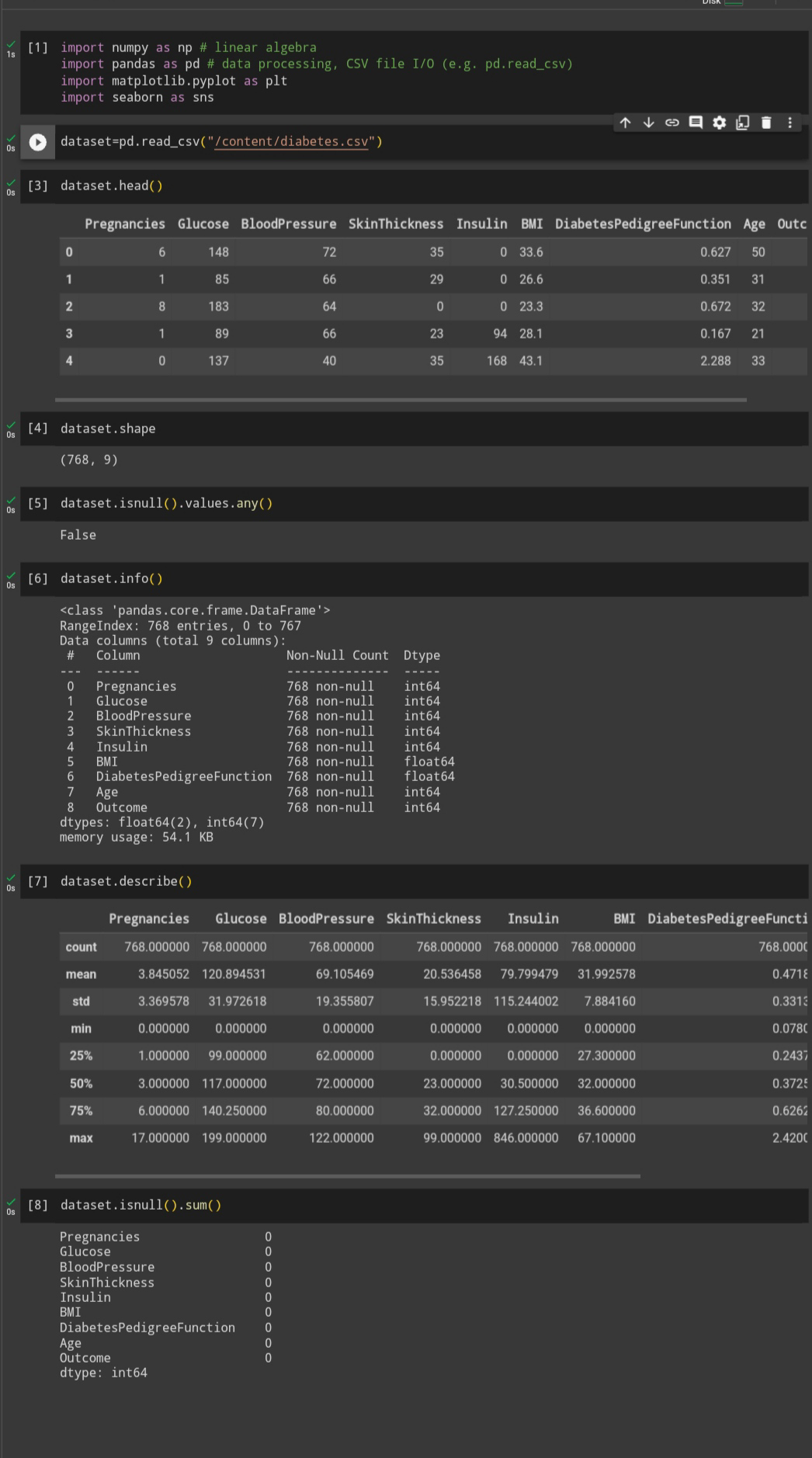
Print(classification\_report(y\_test, y\_pred))

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

Sns.heatmap(cm, annot=True, fmt=’d’)

Plt.show()

**OUTPUT:**

****

**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

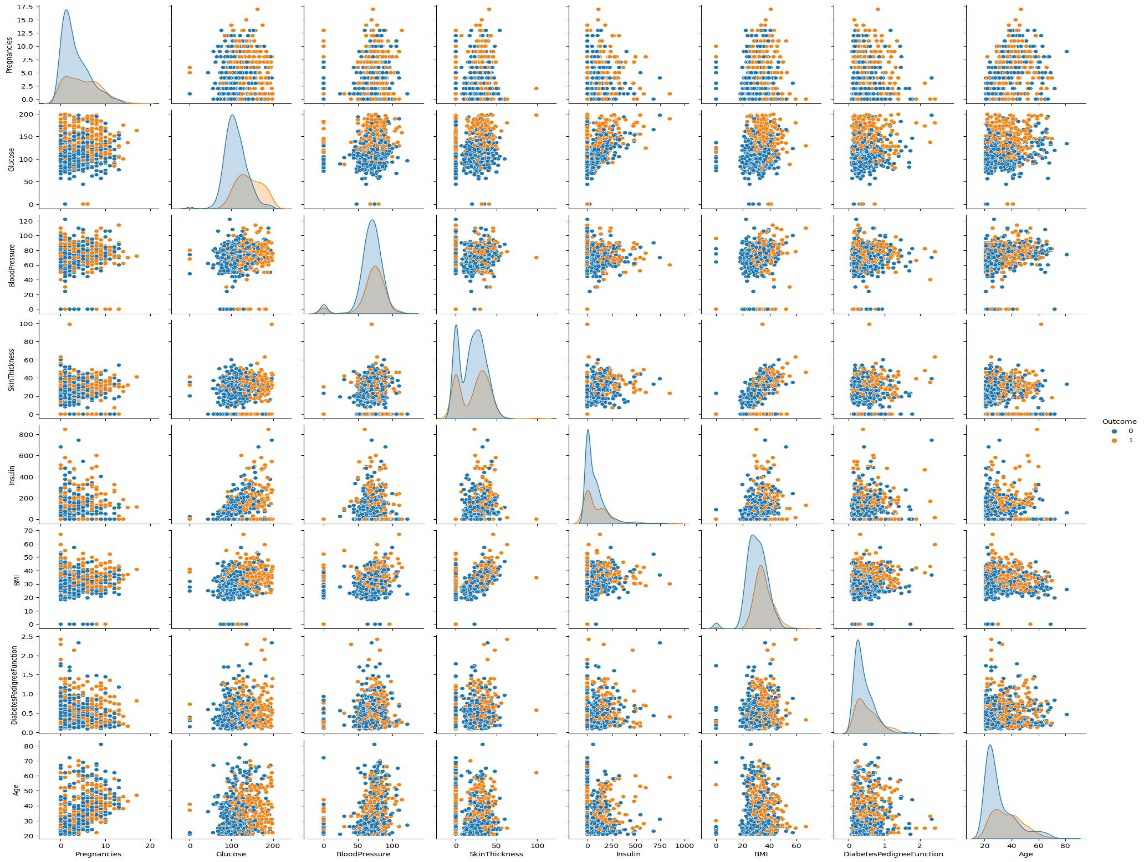
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Count | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| Mean | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| Std | 7.884160 | 0.331329 | 11.760232 | 0.476951 |
| Min | 0.000000 | 0.078000 | 21.000000 | 0.000000 |
| 25% | 27.300000 | 0.243750 | 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

1. 500
2. 1 268

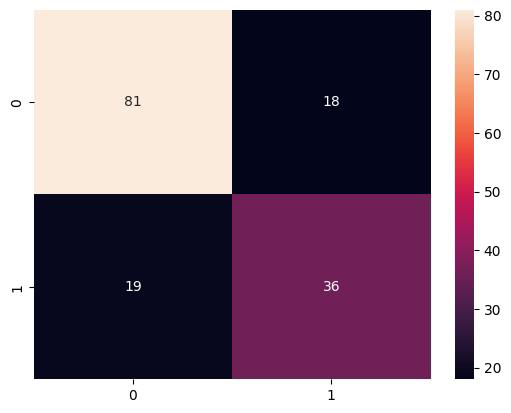
**Name: count, dtype: int64**

****

**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 |

****