PROJECT-2: ANALYSIS REPORT OF DIABETES PATIENTS

Objective:

The main objective of this project is to diagnostically predict whether a patient has diabetes or not.

Note: The dataset used for this analysis is sourced from the National Institute of Diabetes and Digestive and Kidney Diseases.

About the dataset:

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

1. Pregnancies: This variable represents the number of times the patient has been pregnant.

2. Glucose: It is the plasma glucose concentration measured in milligrams per deciliter (mg/dL) of blood. This is a key indicator for diabetes diagnosis.

3. Blood Pressure: This variable represents the diastolic blood pressure (mm Hg) of the patient.

4. Skin Thickness: It indicates the skin thickness (mm) at the triceps area. Skin thickness can sometimes be relevant in diabetes diagnosis.

5. Insulin: This variable represents the serum insulin level (mu U/ml). Insulin is a hormone that regulates blood sugar levels, and its measurement can be important in diabetes diagnosis.

6. BMI (Body Mass Index): BMI is calculated from the weight and height of the patient. It's a measure of body fat and is often used to assess whether a person is underweight, normal weight, overweight, or obese.

7. Diabetes Pedigree Function: This function is used to represent the diabetes pedigree function, which provides information about diabetes mellitus history in relatives and genetic influence.

8. Age: This variable represents the age of the patient in years.

9. Outcome: This is the target variable, and it indicates whether the patient has diabetes or not. It is binary, with values 0 and 1, where:

- 0 typically indicates that the patient does not have diabetes.
- 1 typically indicates that the patient has diabetes.

Steps:

Step-1: load the dataset and import necessary libraries

```
#importing libraries and loading csv file
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.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
data = pd.read_csv("diabetes.csv")
```

Step-2: Data Exploration

```
#Get the summary statistics
print(data.describe())
```

```
       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
```

Step-3: Data Pre-processing

```
#data preprocessing
#check for the missing values
print(data.isnull().sum())
data = data.dropna()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
#Hence,there are no missing values in the data
```

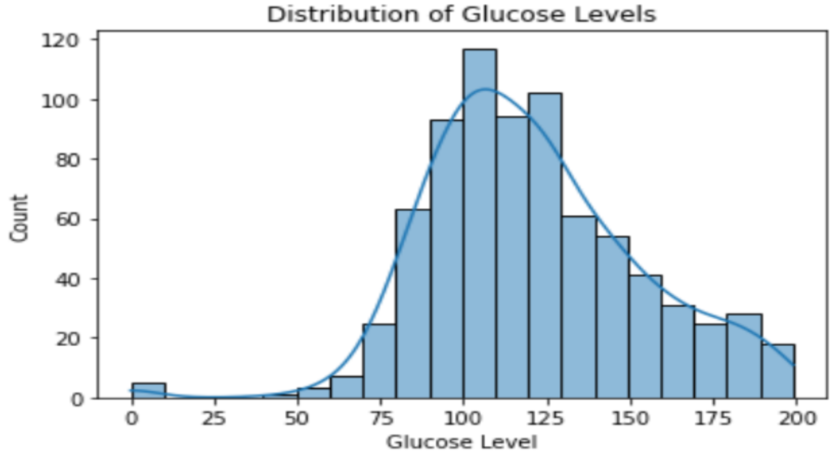
```
#checking for duplicate data
print(data[data.duplicated()])
```

```
Empty DataFrame
Columns: [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction,
Age, Outcome]
Index: []
```

```
#There are no duplicates in the data
```

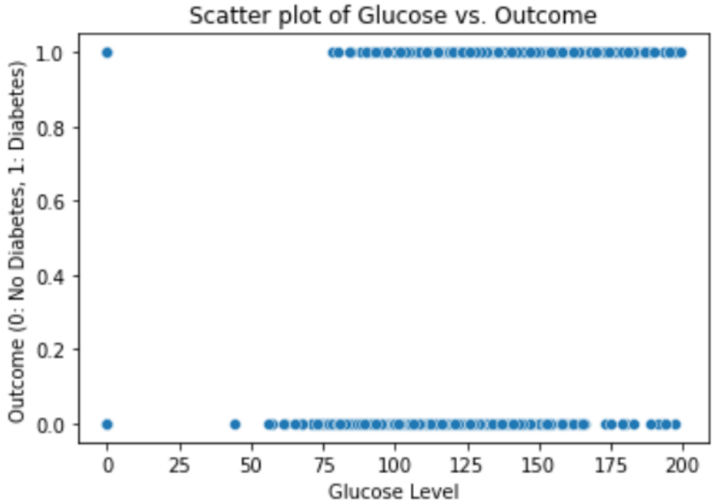
Step-4: Data Visualization

```
#Create histograms for numeric variables
sns.histplot(data['Glucose'], bins=20, kde=True)
plt.xlabel('Glucose Level')
plt.ylabel('Count')
plt.title('Distribution of Glucose Levels')
plt.show()
```

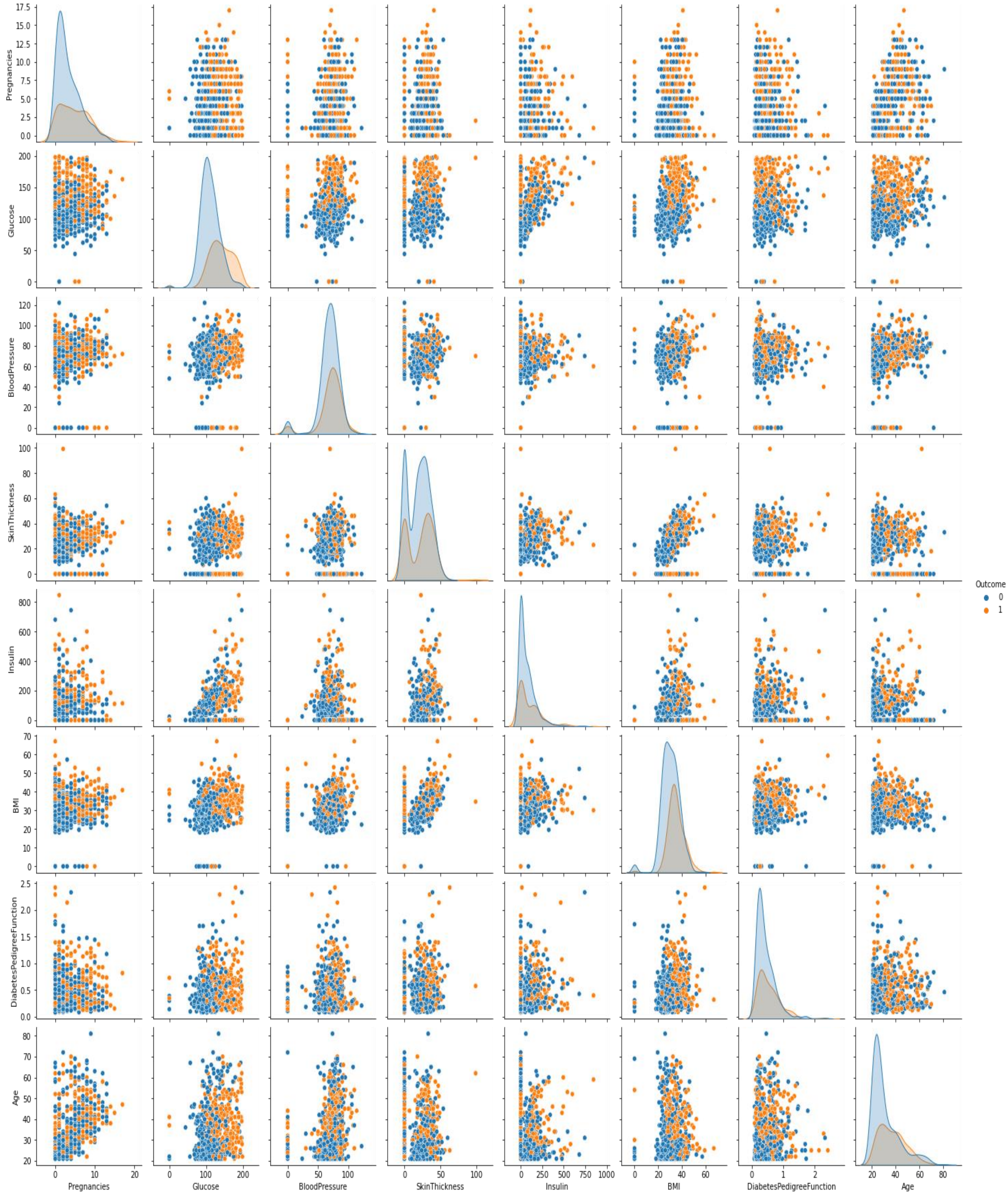


Step-5: Feature analysis

```
In [81]: # Scatter plot between Glucose and Outcome
sns.scatterplot(x='Glucose', y='Outcome', data=data)
plt.xlabel('Glucose Level')
plt.ylabel('Outcome (0: No Diabetes, 1: Diabetes)')
plt.title('Scatter plot of Glucose vs. Outcome')
plt.show()
```

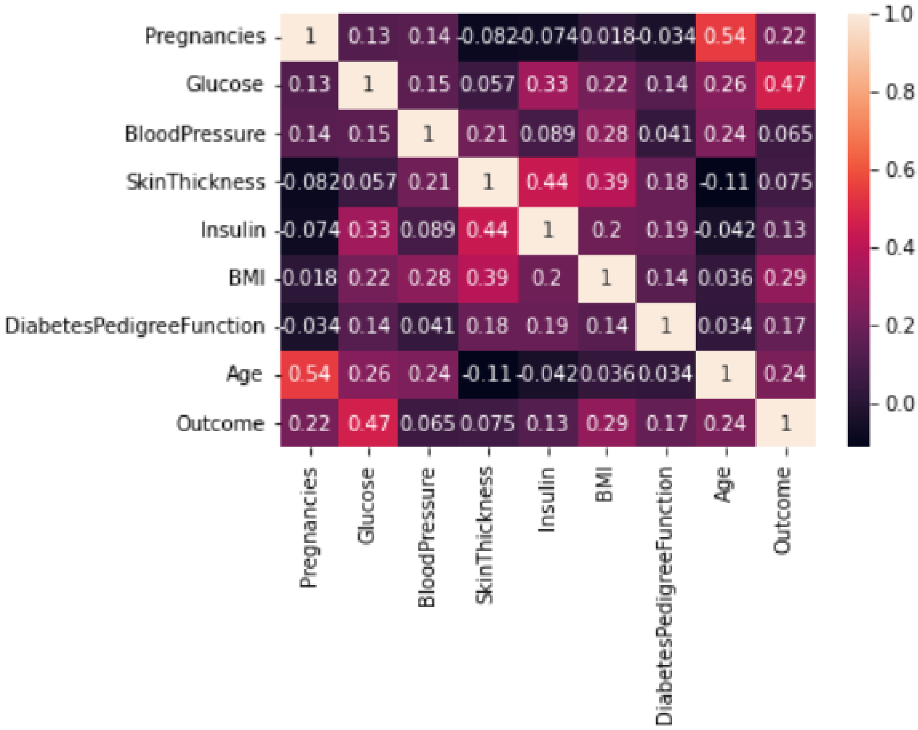



```
#creating visualization to understand the data better
sns.pairplot(data, hue= "Outcome")
plt.show()
```



Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
0
1

```python
#Correlation matrix
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```



Step-6: Model building and evaluation

```python
#splitting the data into features(x) and target (y)
x = data.drop("Outcome", axis=1)
y= data["Outcome"]
```

```python
#importing the libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```python
#splitting into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```python
#feature scaling
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

```python
# training machine learning model using logistic regression
model= LogisticRegression()
model.fit(x_train, y_train)
```

```
LogisticRegression()
```

```
#evaluate the model-accuracy,precision,recall and F1 Score
```

```
#make predictions in the test set
y_pred= model.predict(x_test)
```

```
#evaluate the model
accuracy= accuracy_score(y_test, y_pred)
```

```
#classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.80      0.81        99
           1       0.65      0.67      0.66        55

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154
```

```
#calculating the performance metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall= recall_score(y_test, y_pred)
f1= f1_score(y_test, y_pred)

#report model performance
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1-score: {f1:.2f}')
```

```
Accuracy: 0.75
Precision: 0.65
Recall: 0.67
F1-score: 0.66
```

Conclusion:

- It has a decent level of precision, indicating that when it predicts positive cases (diabetes), it's correct about 65% of the time.
- The recall value indicates that the model is reasonably effective at identifying actual positive cases, capturing about 67% of them.
- The F1-score of 0.66 suggests that the model provides a balanced performance in terms of precision and recall.
- Out of a total of 768 patients, 268 have been diagnosed with diabetes.
- A higher number of pregnancies is associated with a decreased likelihood of diabetes.
- Patients with above-average blood pressure tend to have a lower likelihood of diabetes.
- An increase in blood pressure, BMI, and skin thickness is correlated with an increased likelihood of developing diabetes.
- Rising levels of glucose and insulin are linked to an increased risk of diabetes.