yan-r-task-12-diabetes-prediction

December 25, 2023

ABOUT DATASET This dataset is originally from the National Institute of Diabetes and Digestiv and Kidney Diseases. The objective of the dataset is to diagnostically predi t whether a patient has diabetes based on certain diagnostic measureme ts included in the dataset. Several constraints were placed on the selectio of these instances from a larger database. In particular, all patients her are females at least 21 years old of Pima Indiaheri

tage. COLUMN DESCRIPTION FOR DIABETES DATA: • Pregnancies • Glucose • Blood Pressure • Skin Thickness • Insulin • BM@I • Diabetes Age • Outcome

[6]:
```
# import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[11]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier  # or another classifier of␣
 ↪your choice
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
```

[2]:
```
# loading dataset

import pandas as pd

# Replace this path with the actual path to your dataset
file_path = r'C:\Users\KARTHIK\OneDrive\Desktop\meriskill␣
 ↪intern\projects\project 2\diabetes.csv'

# Read the dataset into a pandas DataFrame
diabetes_df = pd.read_csv(file_path)

# Display the first few rows of the DataFrame to check if the import was␣
 ↪successful
diabetes_df.head()
```

[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[4]:
```
diabetes_df
```

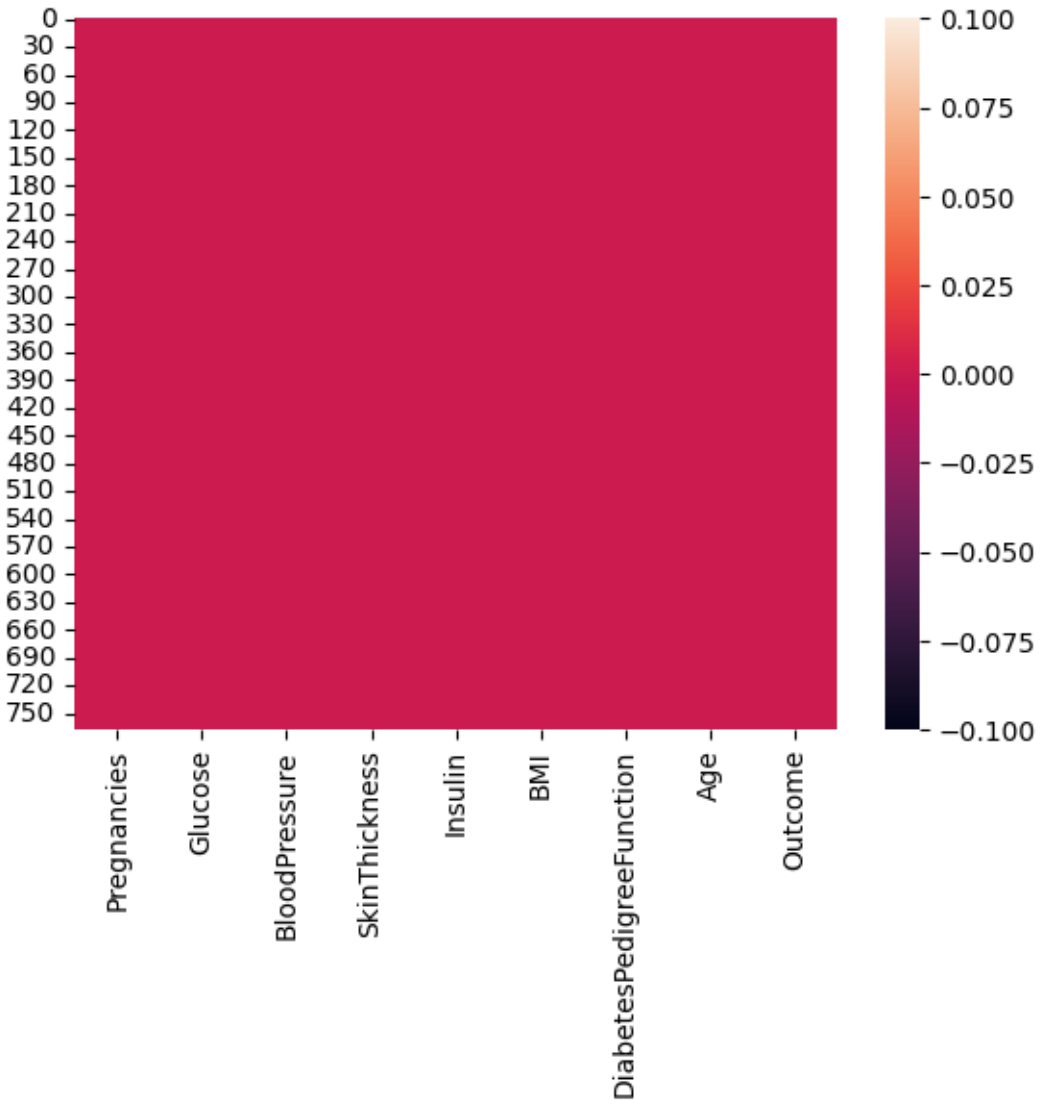
[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
..	…	…	…	…	…	…	…	…	…
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

[768 rows x 9 columns]

[7]:
```
sns.heatmap(diabetes_df.isnull())
```

[7]: <Axes: >

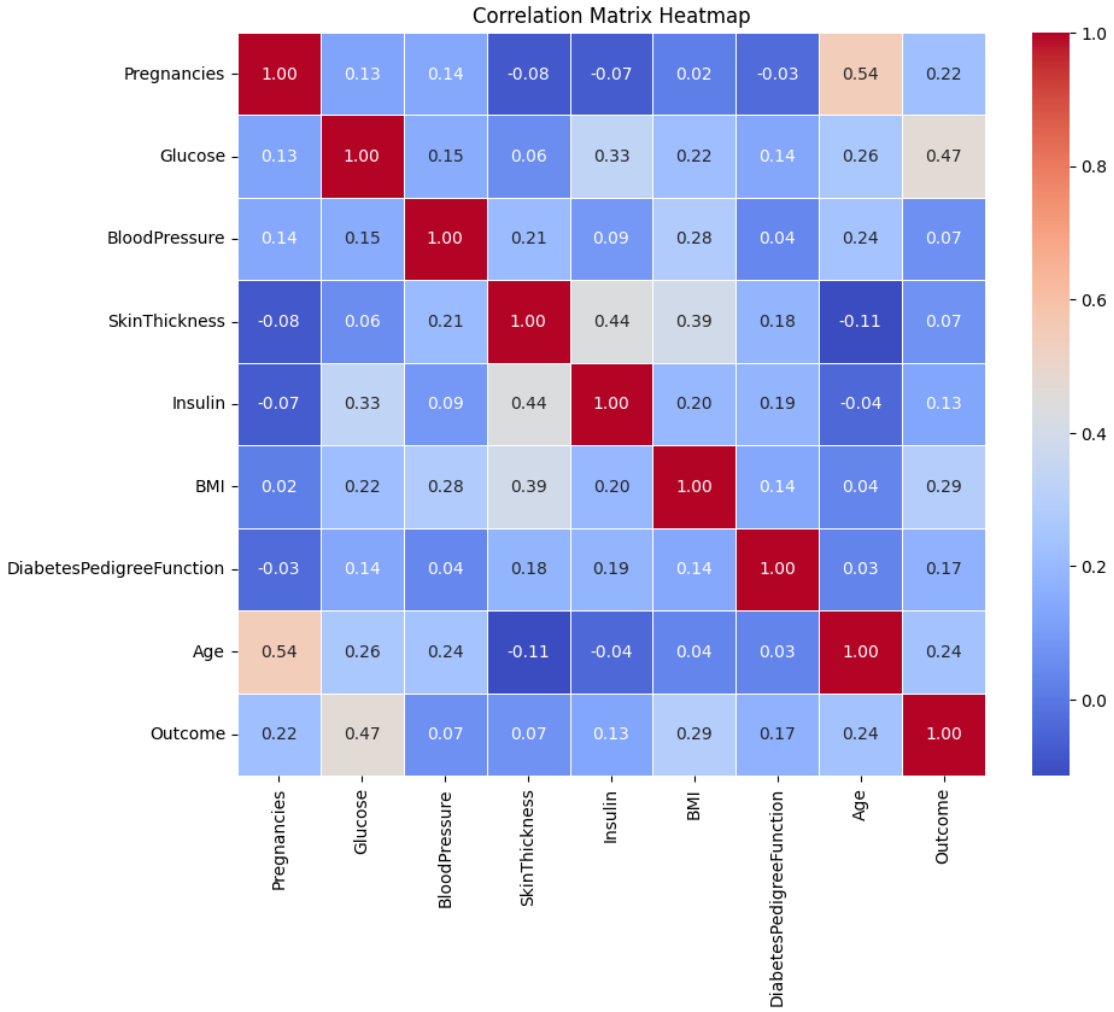


[8]:
```
# Assuming 'diabetes_df' is your DataFrame containing the dataset
correlation_matrix = diabetes_df.corr()

# Display the correlation matrix
print(correlation_matrix)
```

```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       1.000000       0.207371
SkinThickness               -0.081672  0.057328       0.207371       1.000000
Insulin                     -0.073535  0.331357       0.088933       0.436783
BMI                          0.017683  0.221071       0.281805       0.392573
```

```
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183928
Age                          0.544341  0.263514       0.239528      -0.113970
Outcome                      0.221898  0.466581       0.065068       0.074752

                           Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies              -0.073535  0.017683                 -0.033523
Glucose                   0.331357  0.221071                  0.137337
BloodPressure             0.088933  0.281805                  0.041265
SkinThickness             0.436783  0.392573                  0.183928
Insulin                   1.000000  0.197859                  0.185071
BMI                       0.197859  1.000000                  0.140647
DiabetesPedigreeFunction  0.185071  0.140647                  1.000000
Age                      -0.042163  0.036242                  0.033561
Outcome                   0.130548  0.292695                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.263514  0.466581
BloodPressure             0.239528  0.065068
SkinThickness            -0.113970  0.074752
Insulin                  -0.042163  0.130548
BMI                       0.036242  0.292695
DiabetesPedigreeFunction  0.033561  0.173844
Age                       1.000000  0.238356
Outcome                   0.238356  1.000000
```

[9]:
```python
# Assuming 'diabetes_df' is your DataFrame containing the dataset
correlation_matrix = diabetes_df.corr()

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))

# Create a heatmap of the correlation matrix
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪linewidths=0.5)

# Set the title of the plot
plt.title('Correlation Matrix Heatmap')

# Show the plot
plt.show()
```

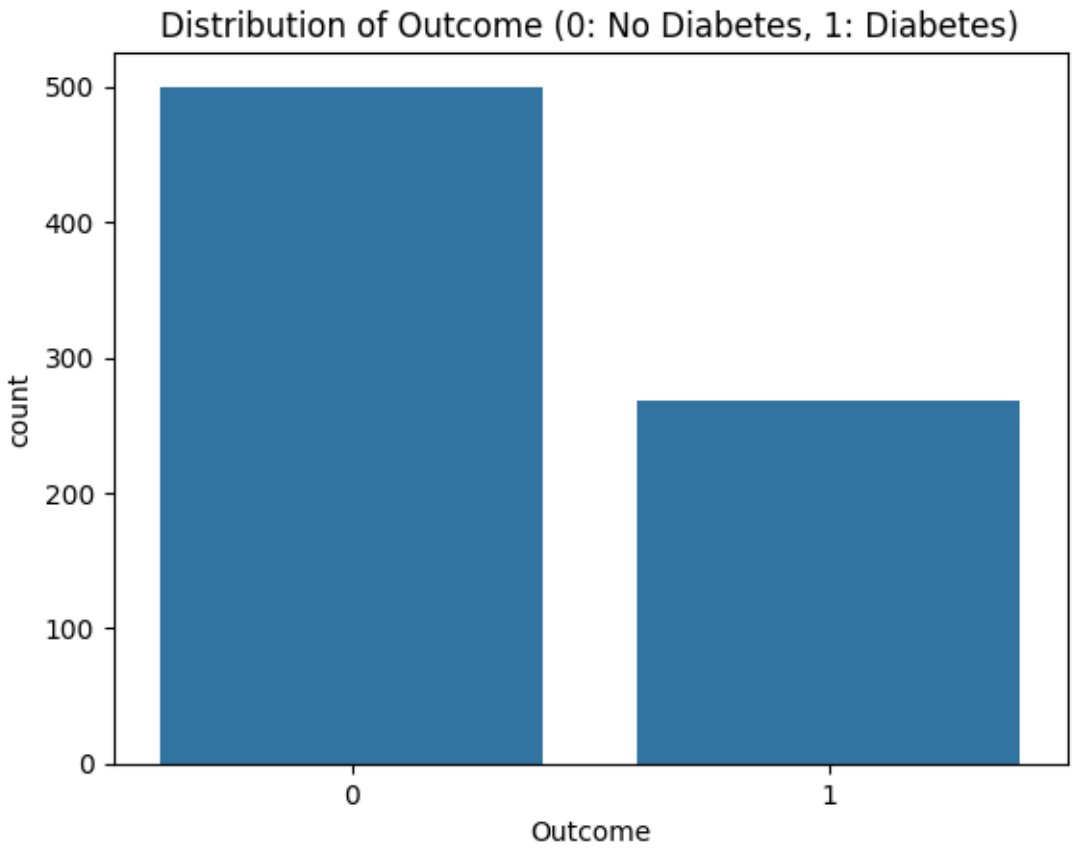


[3]:
```
# Display the first few rows of the dataset
print(diabetes_df.head())

# Check for missing values
print(diabetes_df.isnull().sum())

# Statistical summary of the dataset
print(diabetes_df.describe())
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1
```

```
   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
       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
```

[8]:
```
# Visualize the distribution of the target variable
sns.countplot(x='Outcome', data=diabetes_df)
plt.title('Distribution of Outcome (0: No Diabetes, 1: Diabetes)')
plt.show()
```



[13]:
```
# Prepare the data for training
X = diabetes_df.drop('Outcome', axis=1)
y = diabetes_df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[14]:
```
# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[15]:
```
# Train a RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train_scaled, y_train)
```

[15]:
```
RandomForestClassifier(random_state=42)
```

[16]:
```
# Make predictions on the test set
predictions = model.predict(X_test_scaled)
```

[17]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.72
```

[18]:
```
# Additional evaluation metrics
print('\nClassification Report:')
print(classification_report(y_test, predictions))
```

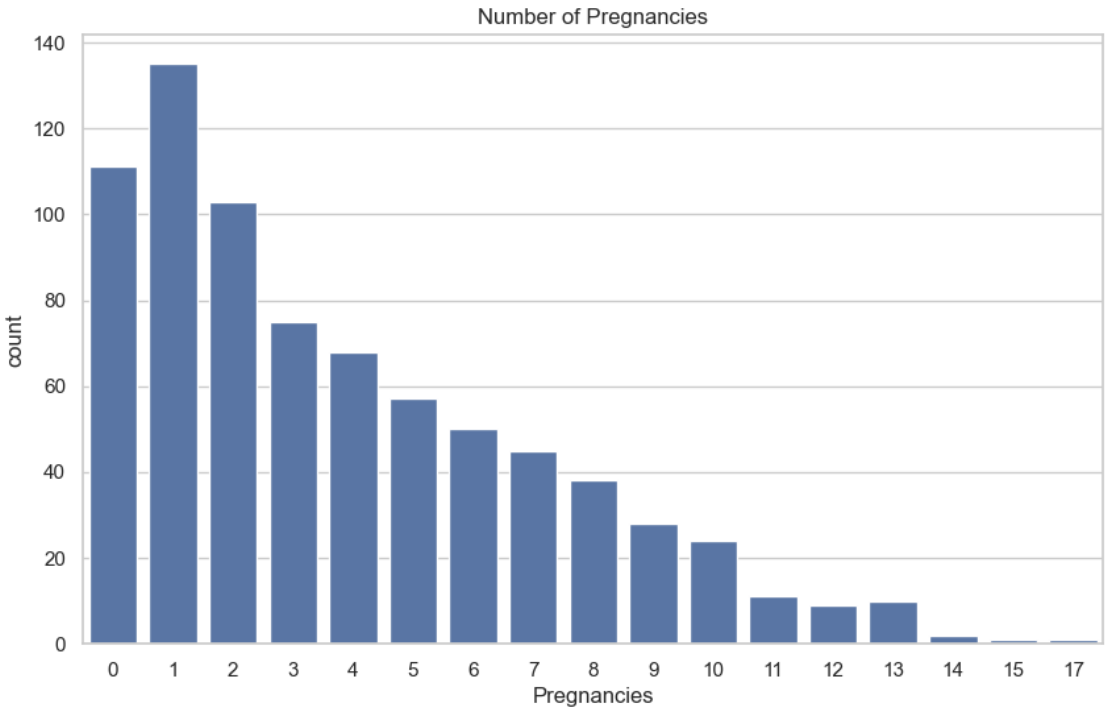
```
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.78      0.78        99
           1       0.61      0.62      0.61        55

    accuracy                           0.72       154
   macro avg       0.70      0.70      0.70       154
weighted avg       0.72      0.72      0.72       154
```

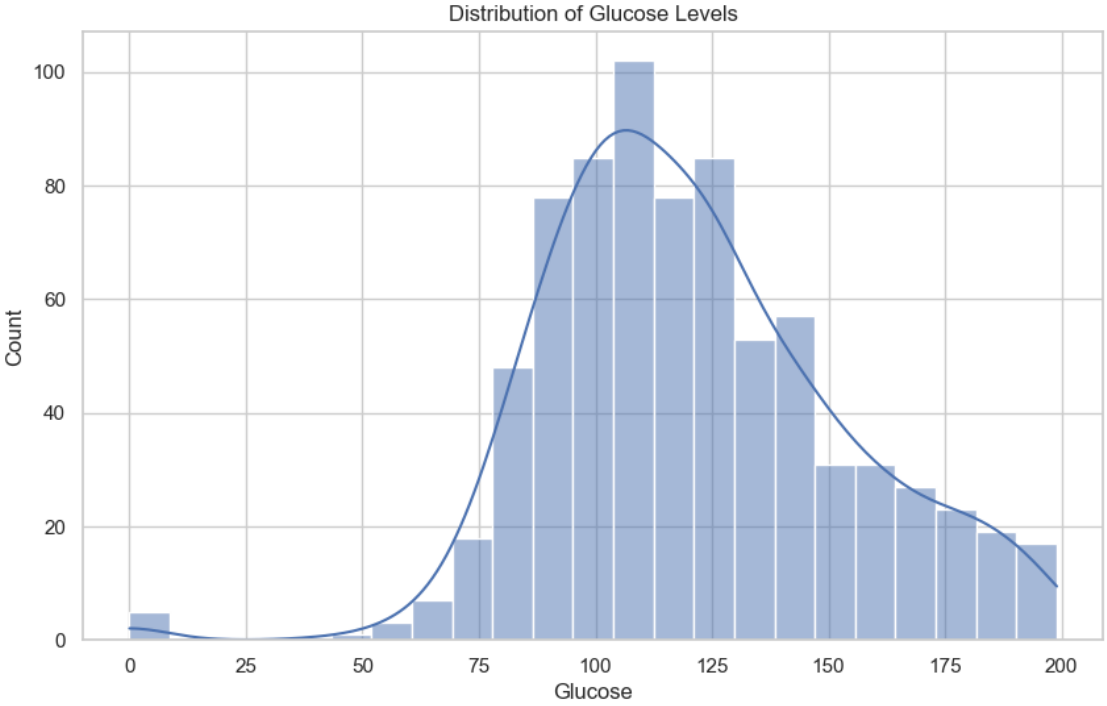
[19]:
```
print('\nConfusion Matrix:')
print(confusion_matrix(y_test, predictions))
```

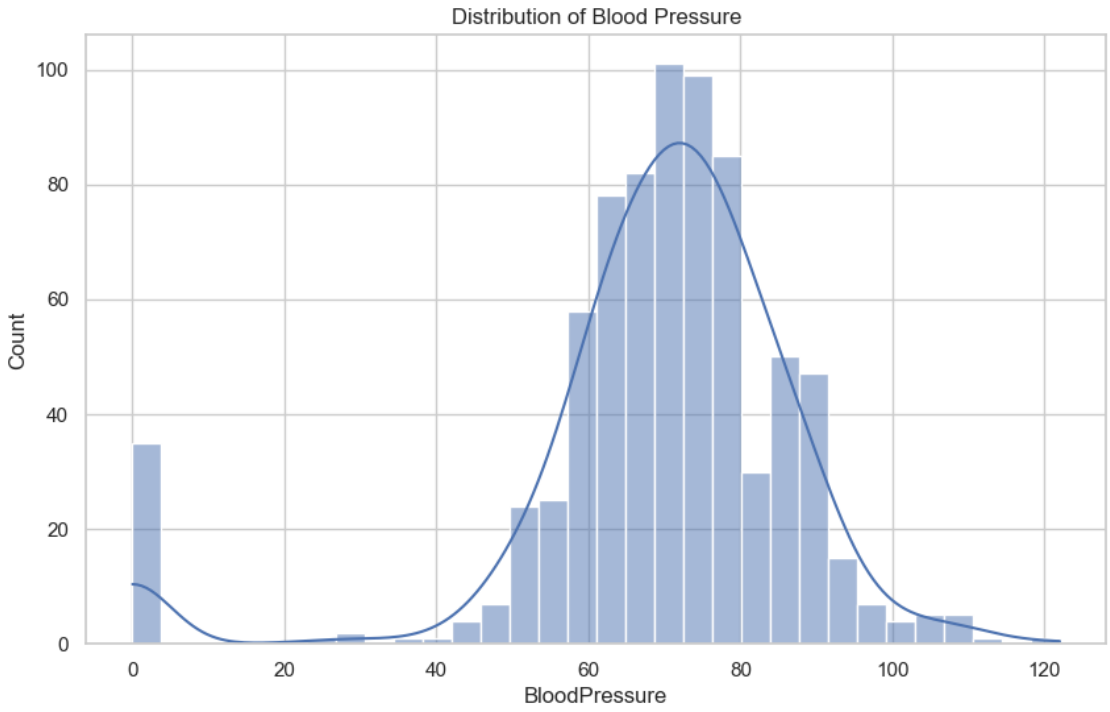
```
Confusion Matrix:
[[77 22]
 [21 34]]
```

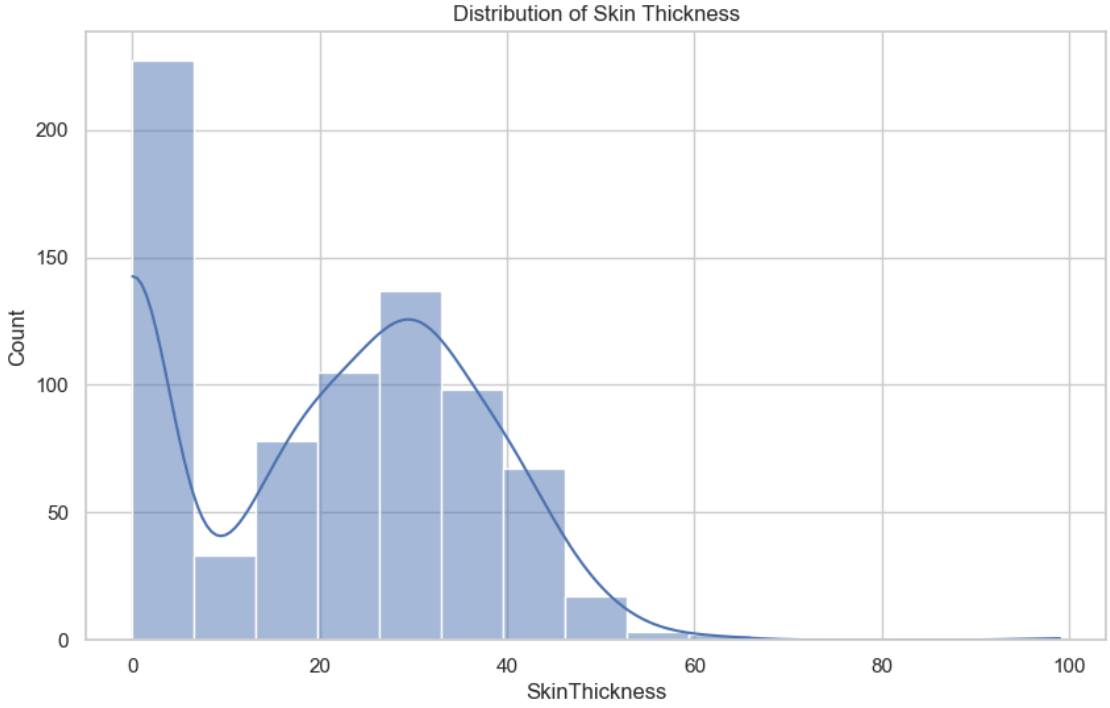
[20]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the style for seaborn plots
sns.set(style="whitegrid")

# Assuming 'diabetes_df' is your DataFrame containing the dataset

# 1. Pregnancies
plt.figure(figsize=(10, 6))
sns.countplot(x='Pregnancies', data=diabetes_df)
plt.title('Number of Pregnancies')
plt.show()

# 2. Glucose
plt.figure(figsize=(10, 6))
sns.histplot(x='Glucose', data=diabetes_df, kde=True)
plt.title('Distribution of Glucose Levels')
```

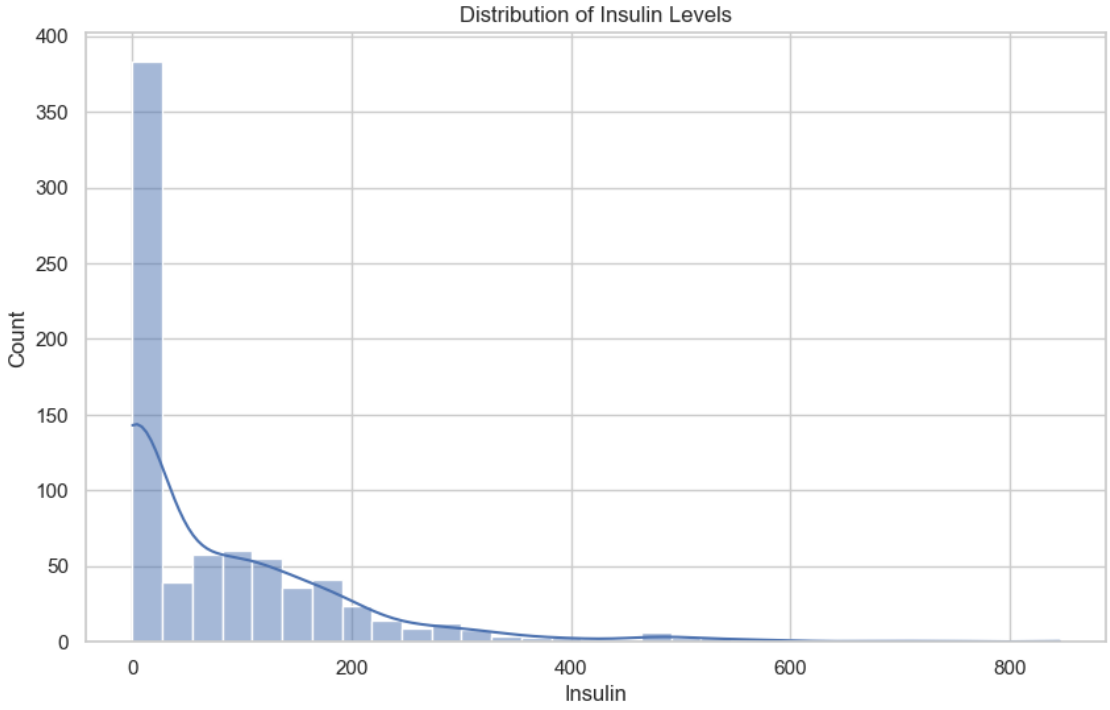
```
plt.show()

# 3. Blood Pressure
plt.figure(figsize=(10, 6))
sns.histplot(x='BloodPressure', data=diabetes_df, kde=True)
plt.title('Distribution of Blood Pressure')
plt.show()

# 4. Skin Thickness
plt.figure(figsize=(10, 6))
sns.histplot(x='SkinThickness', data=diabetes_df, kde=True)
plt.title('Distribution of Skin Thickness')
plt.show()

# 5. Insulin
plt.figure(figsize=(10, 6))
sns.histplot(x='Insulin', data=diabetes_df, kde=True)
plt.title('Distribution of Insulin Levels')
plt.show()

# 6. BMI
plt.figure(figsize=(10, 6))
sns.histplot(x='BMI', data=diabetes_df, kde=True)
plt.title('Distribution of BMI')
plt.show()

# 7. Diabetes Pedigree Function
plt.figure(figsize=(10, 6))
sns.histplot(x='DiabetesPedigreeFunction', data=diabetes_df, kde=True)
plt.title('Distribution of Diabetes Pedigree Function Scores')
plt.show()

# 8. Age
plt.figure(figsize=(10, 6))
sns.histplot(x='Age', data=diabetes_df, kde=True)
plt.title('Distribution of Age')
plt.show()

# 9. Outcome
plt.figure(figsize=(8, 5))
sns.countplot(x='Outcome', data=diabetes_df)
plt.title('Distribution of Outcome (0: No Diabetes, 1: Diabetes)')
plt.show()
```

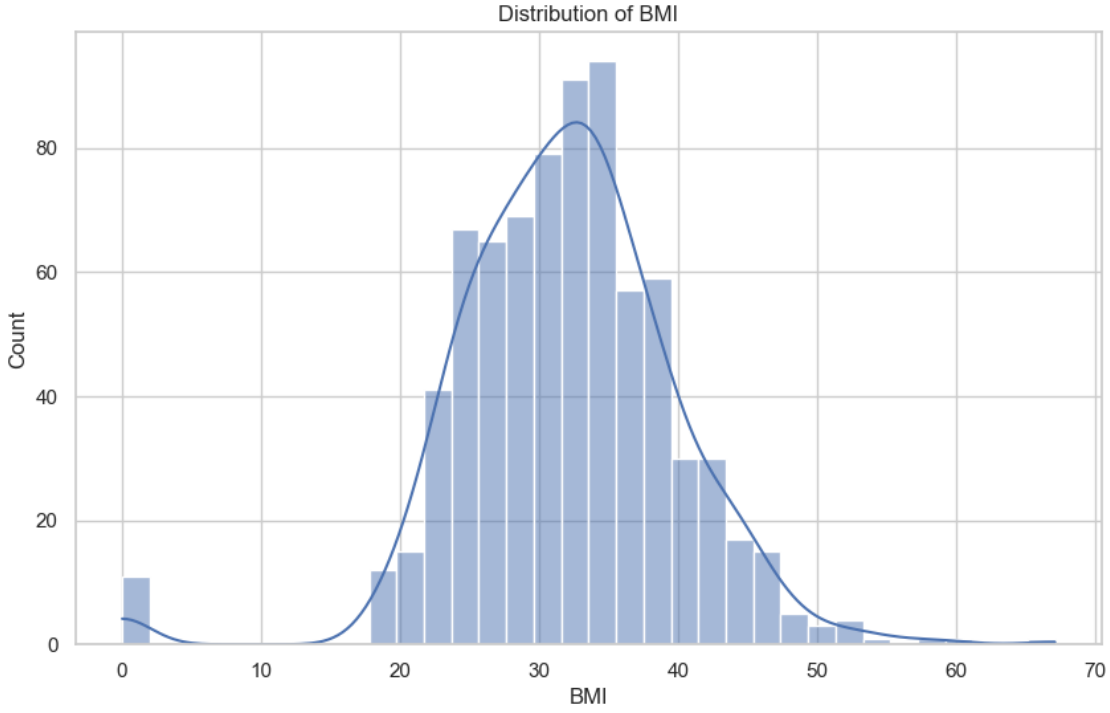

Number of Pregnancies
count
Pregnancies
0
20
40
60
80
100
120
140
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
17

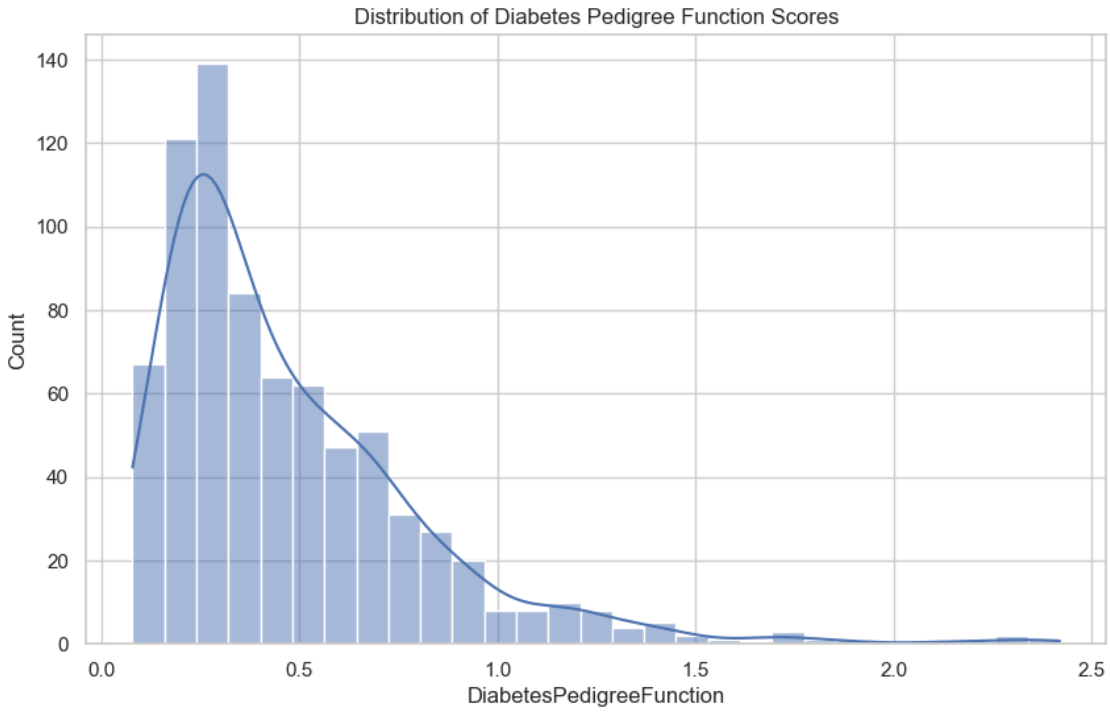

Distribution of Glucose Levels
Count
Glucose
0
20
40
60
80
100
0
25
50
75
100
125
150
175
200

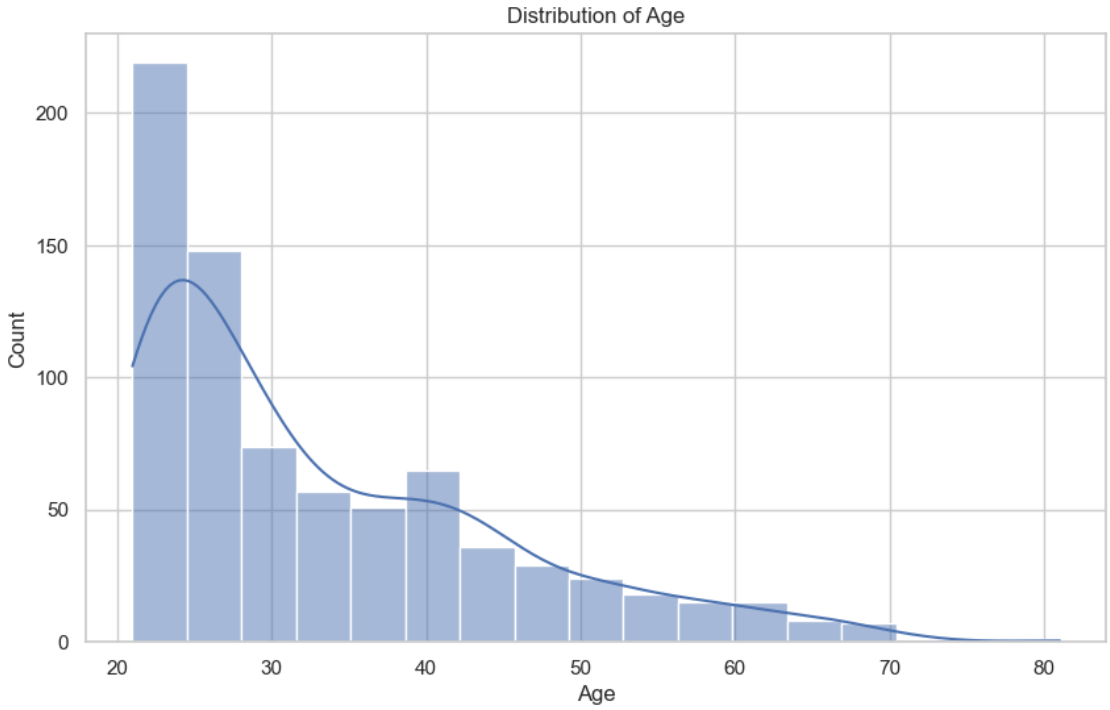

Distribution of Blood Pressure
Count
BloodPressure
0
20
40
60
80
100
120

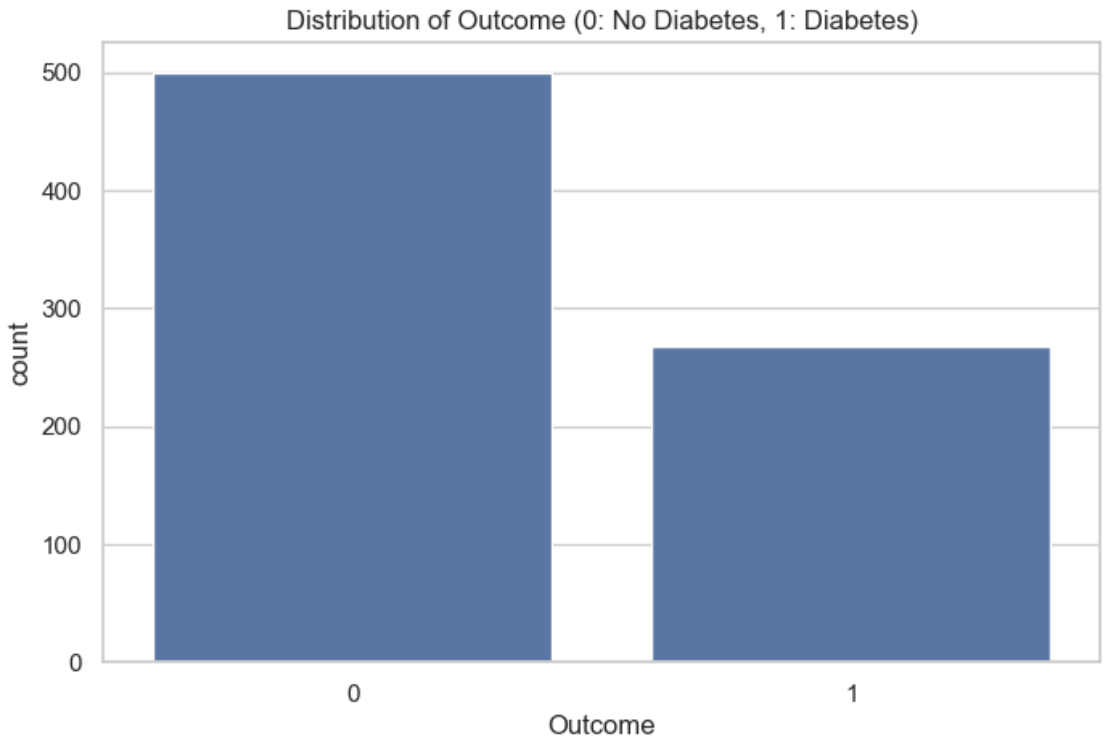

Distribution of Skin Thickness
Count
SkinThickness
0
50
100
150
200
0
20
40
60
80
100


Distribution of Insulin Levels
Count
Insulin
0
50
100
150
200
250
300
350
400
0
200
400
600
800

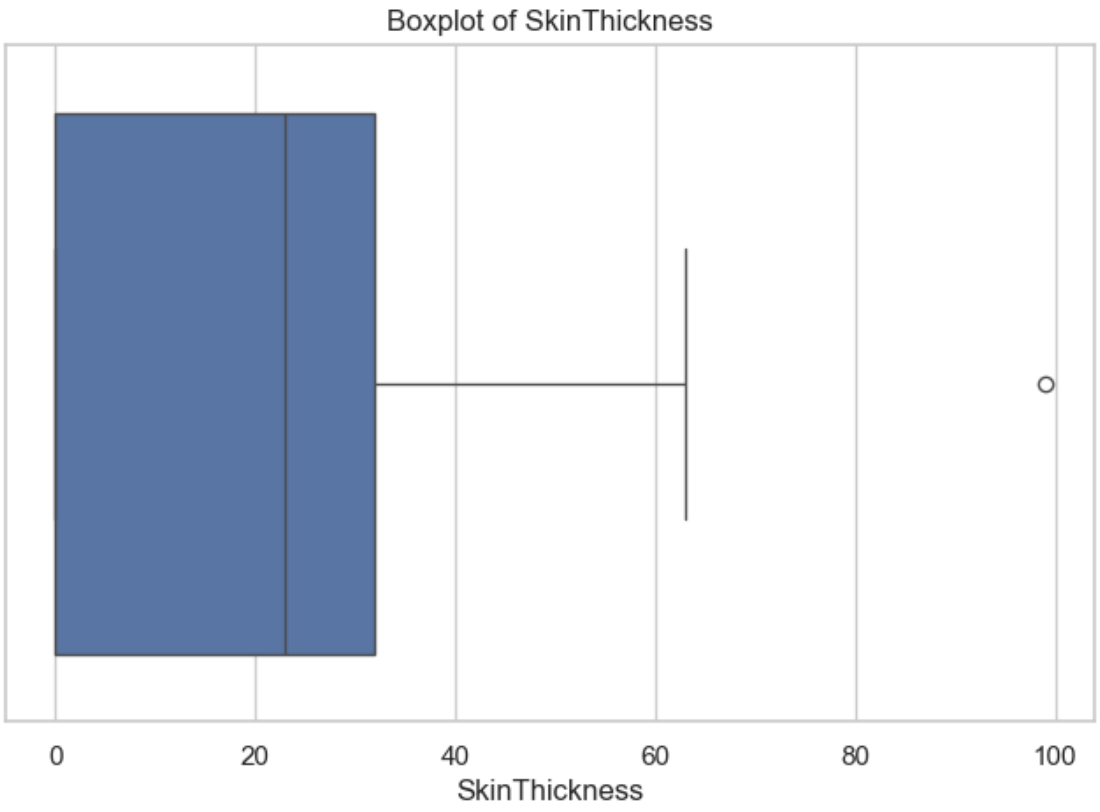

Distribution of BMI
Count
BMI
0
20
40
60
80
0
10
20
30
40
50
60
70


Distribution of Diabetes Pedigree Function Scores
Count
DiabetesPedigreeFunction
0
20
40
60
80
100
120
140
0.0
0.5
1.0
1.5
2.0
2.5


Distribution of Age
Count
Age
0
50
100
150
200
20
30
40
50
60
70
80



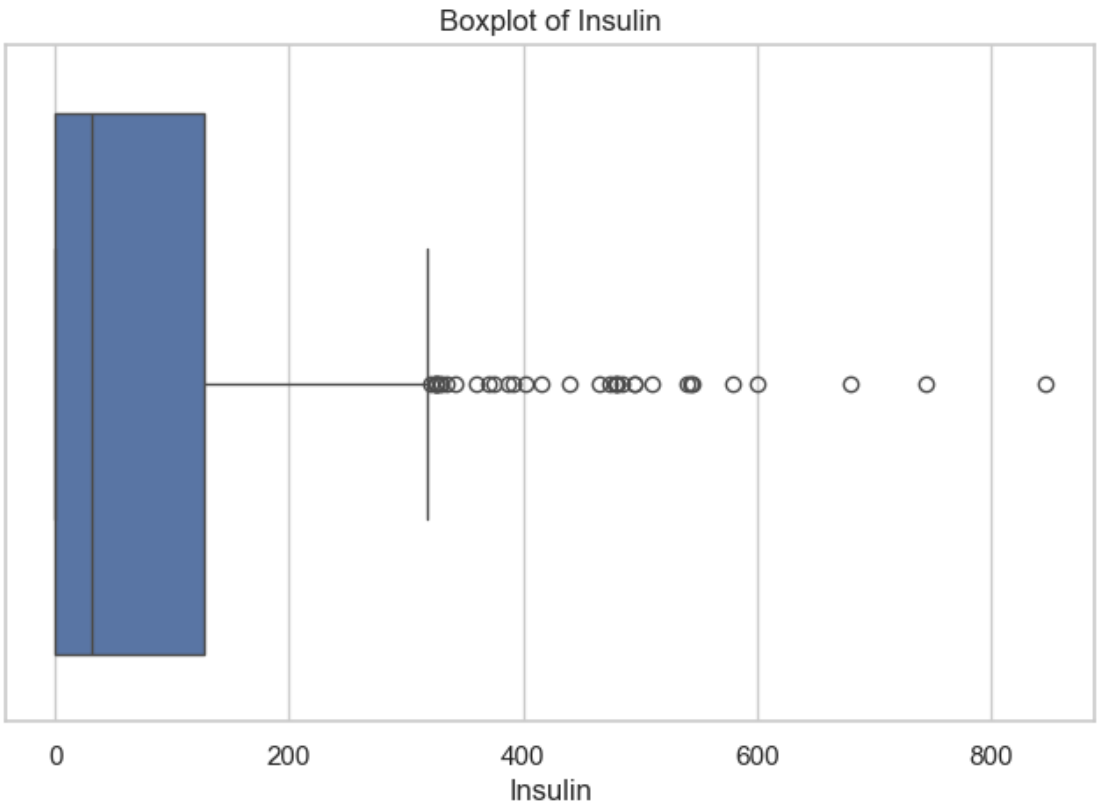
[21]:
```
# Visualize the distribution of 'SkinThickness'
plt.figure(figsize=(8, 5))
sns.boxplot(x='SkinThickness', data=diabetes_df)
plt.title('Boxplot of SkinThickness')
plt.show()
```



[23]:
```
# Checking for outliers in 'SkinThickness'
Q1_skin_thickness = diabetes_df['SkinThickness'].quantile(0.25)
Q3_skin_thickness = diabetes_df['SkinThickness'].quantile(0.75)
IQR_skin_thickness = Q3_skin_thickness - Q1_skin_thickness
lower_bound_skin_thickness = Q1_skin_thickness - 1.5 * IQR_skin_thickness
upper_bound_skin_thickness = Q3_skin_thickness + 1.5 * IQR_skin_thickness
outliers_skin_thickness = diabetes_df[(diabetes_df['SkinThickness'] <␣
 ↪lower_bound_skin_thickness) | (diabetes_df['SkinThickness'] >␣
 ↪upper_bound_skin_thickness)]

print(f'Number of outliers in SkinThickness: {len(outliers_skin_thickness)}')
```

```
Number of outliers in SkinThickness: 1
```

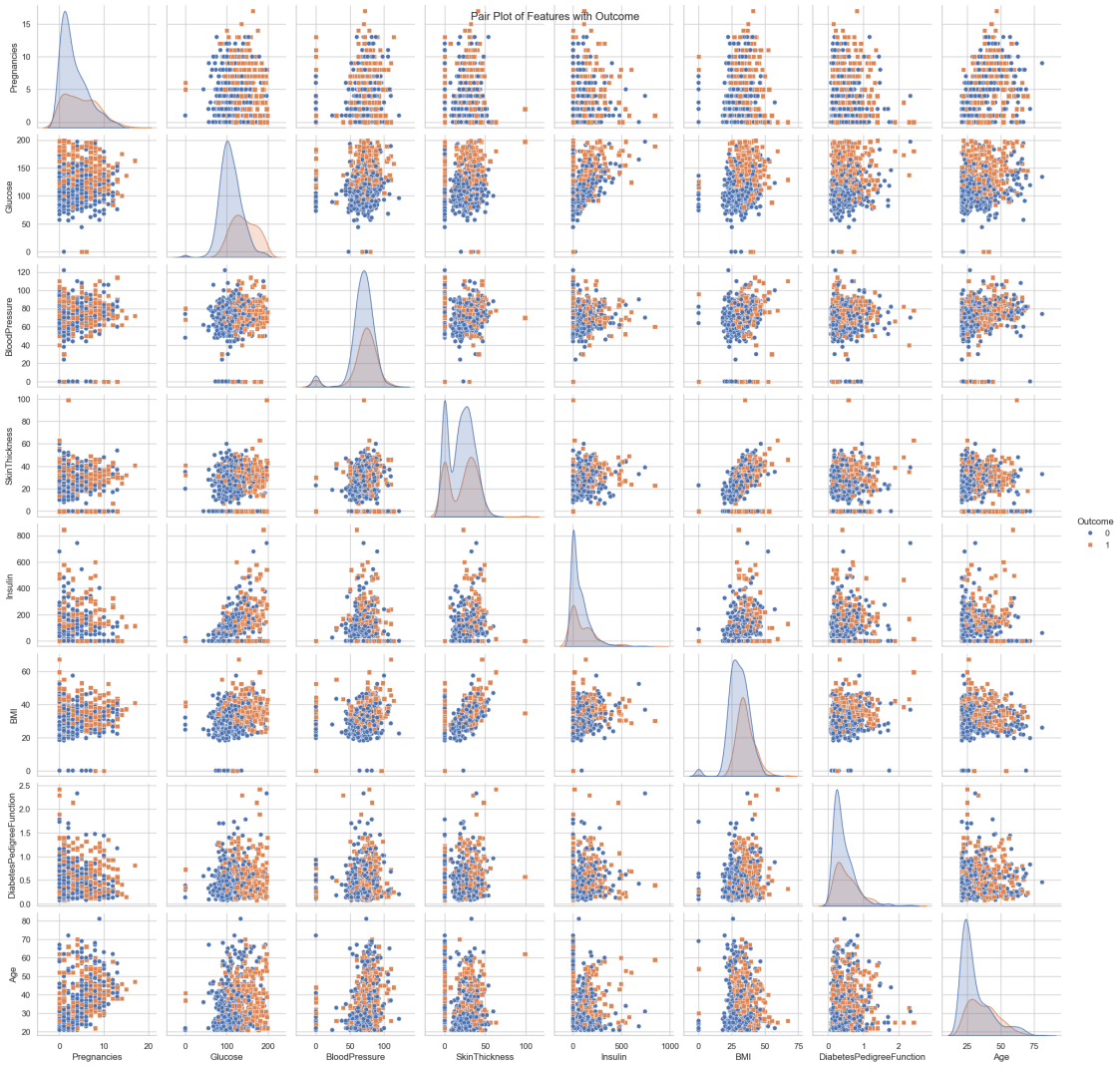
[25]:
```
# Visualize the distribution of 'Insulin'
plt.figure(figsize=(8, 5))
sns.boxplot(x='Insulin', data=diabetes_df)
plt.title('Boxplot of Insulin')
plt.show()
```



[27]:
```
# Checking for outliers in 'Insulin'
Q1_insulin = diabetes_df['Insulin'].quantile(0.25)
Q3_insulin = diabetes_df['Insulin'].quantile(0.75)
IQR_insulin = Q3_insulin - Q1_insulin
lower_bound_insulin = Q1_insulin - 1.5 * IQR_insulin
upper_bound_insulin = Q3_insulin + 1.5 * IQR_insulin
outliers_insulin = diabetes_df[(diabetes_df['Insulin'] < lower_bound_insulin) |␣
↪(diabetes_df['Insulin'] > upper_bound_insulin)]

print(f'Number of outliers in Insulin: {len(outliers_insulin)}')
```

```
Number of outliers in Insulin: 34
```

[29]:
```
# Visualize relationships between features with a pair plot
sns.pairplot(diabetes_df, hue='Outcome', diag_kind='kde', markers=["o", "s"])
plt.suptitle('Pair Plot of Features with Outcome')
plt.show()
```



[36]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
```

[38]:
```
# Initialize and evaluate different models
decision_tree_model = DecisionTreeClassifier(random_state=42)
random_forest_model = RandomForestClassifier(random_state=42)
knn_model = KNeighborsClassifier()
```

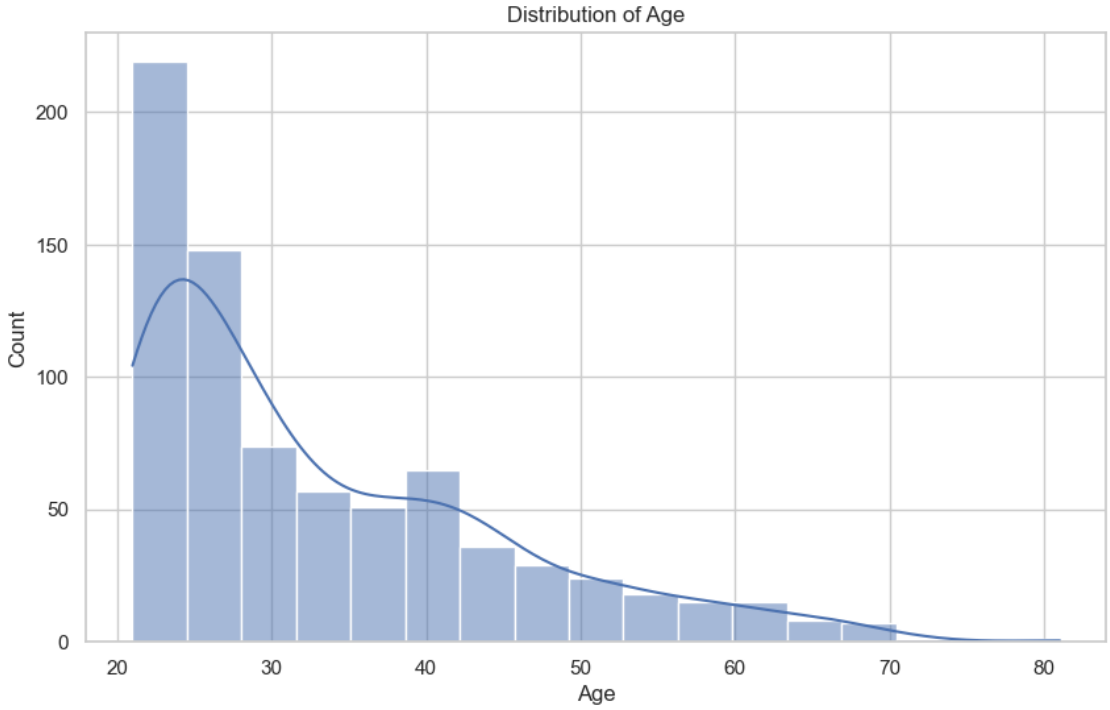
```
xgboost_model = XGBClassifier(random_state=42)
adaboost_model = AdaBoostClassifier(random_state=42)

models = [decision_tree_model, random_forest_model, knn_model, xgboost_model,␣
 ↪adaboost_model]

for model in models:
    train_evaluate_model(model, X_train_scaled, y_train, X_test_scaled, y_test)
```

```
Model: DecisionTreeClassifier
Accuracy: 0.75

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.76      0.79        99
           1       0.62      0.73      0.67        55

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


Confusion Matrix:
[[75 24]
 [15 40]]
--------------------------------------------------
Model: RandomForestClassifier
Accuracy: 0.72

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.78      0.78        99
           1       0.61      0.62      0.61        55

    accuracy                           0.72       154
   macro avg       0.70      0.70      0.70       154
weighted avg       0.72      0.72      0.72       154


Confusion Matrix:
[[77 22]
 [21 34]]
--------------------------------------------------
Model: KNeighborsClassifier
Accuracy: 0.69
```

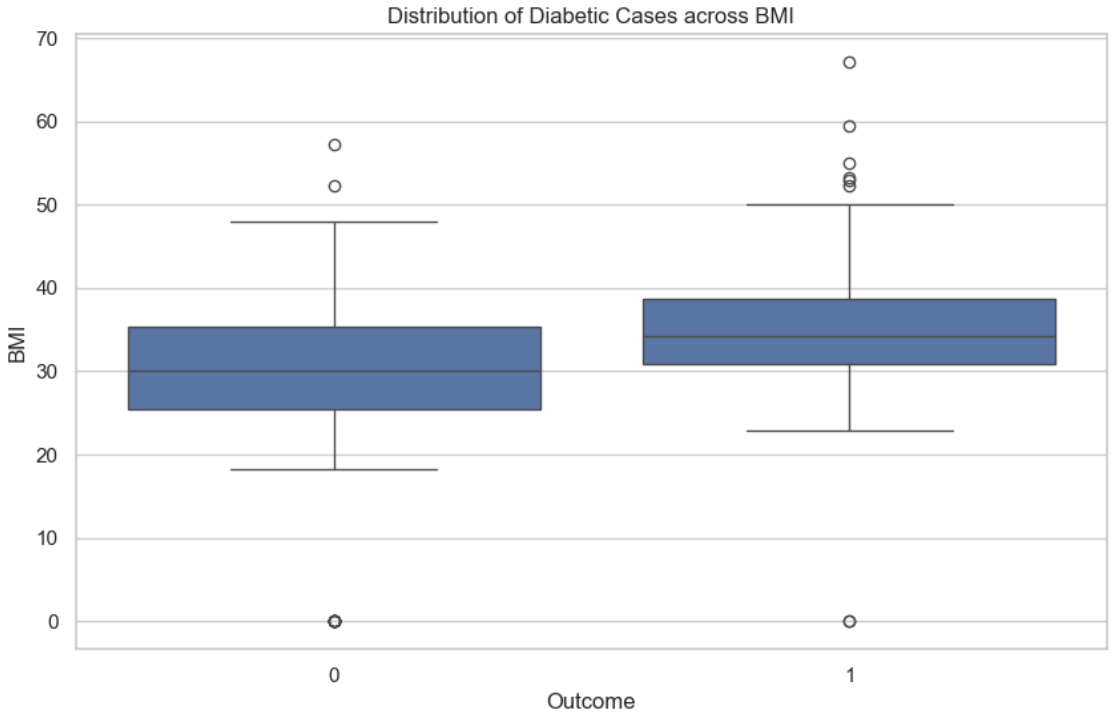
```
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.80      0.77        99
           1       0.58      0.51      0.54        55

    accuracy                           0.69       154
   macro avg       0.66      0.65      0.66       154
weighted avg       0.69      0.69      0.69       154


Confusion Matrix:
[[79 20]
 [27 28]]
--------------------------------------------------
Model: XGBClassifier
Accuracy: 0.71

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.74      0.76        99
           1       0.58      0.65      0.62        55

    accuracy                           0.71       154
   macro avg       0.69      0.70      0.69       154
weighted avg       0.72      0.71      0.71       154


Confusion Matrix:
[[73 26]
 [19 36]]
--------------------------------------------------
Model: AdaBoostClassifier
Accuracy: 0.73

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.79      0.79        99
           1       0.62      0.64      0.63        55

    accuracy                           0.73       154
   macro avg       0.71      0.71      0.71       154
weighted avg       0.73      0.73      0.73       154
```

```
Confusion Matrix:
[[78 21]
 [20 35]]
--------------------------------------------------
```

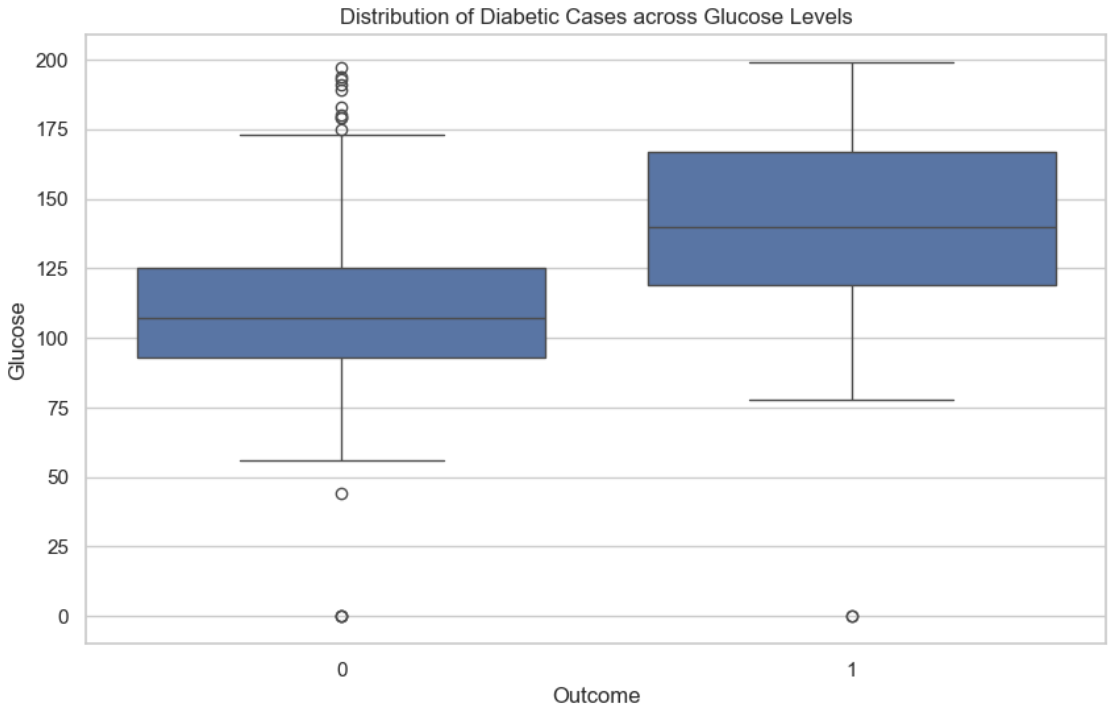
[39]:
```
# Age Distribution
plt.figure(figsize=(10, 6))
sns.histplot(x='Age', data=diabetes_df, kde=True)
plt.title('Distribution of Age')
plt.show()
```



[40]:
```
# Distribution of Diabetic Cases across BMI
plt.figure(figsize=(10, 6))
sns.boxplot(x='Outcome', y='BMI', data=diabetes_df)
plt.title('Distribution of Diabetic Cases across BMI')
plt.show()
```



[41]:
```
# Distribution of Diabetic Cases across Glucose Levels
plt.figure(figsize=(10, 6))
sns.boxplot(x='Outcome', y='Glucose', data=diabetes_df)
plt.title('Distribution of Diabetic Cases across Glucose Levels')
plt.show()
```


Distribution of Diabetic Cases across Glucose Levels
Glucose
200
175
150
125
100
75
50
25
0
0
1
Outcome

[ ]: