```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report

df=pd.read_csv('diabetes.csv')

df.isnull()
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin
BMI  \
0          False    False          False          False    False
False
1          False    False          False          False    False
False
2          False    False          False          False    False
False
3          False    False          False          False    False
False
4          False    False          False          False    False
False
..           ...      ...            ...            ...      ...    ..
.
763        False    False          False          False    False
False
764        False    False          False          False    False
False
765        False    False          False          False    False
False
766        False    False          False          False    False
False
767        False    False          False          False    False
False

     DiabetesPedigreeFunction    Age  Outcome
0                       False  False    False
1                       False  False    False
2                       False  False    False
3                       False  False    False
4                       False  False    False
..                        ...    ...      ...
763                     False  False    False
```

```
764                       False  False    False
765                       False  False    False
766                       False  False    False
767                       False  False    False

[768 rows x 9 columns]
```

```
df.isnull().sum()
```

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

```
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
763    False
764    False
765    False
766    False
767    False
Length: 768, dtype: bool
```

```
df.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
```

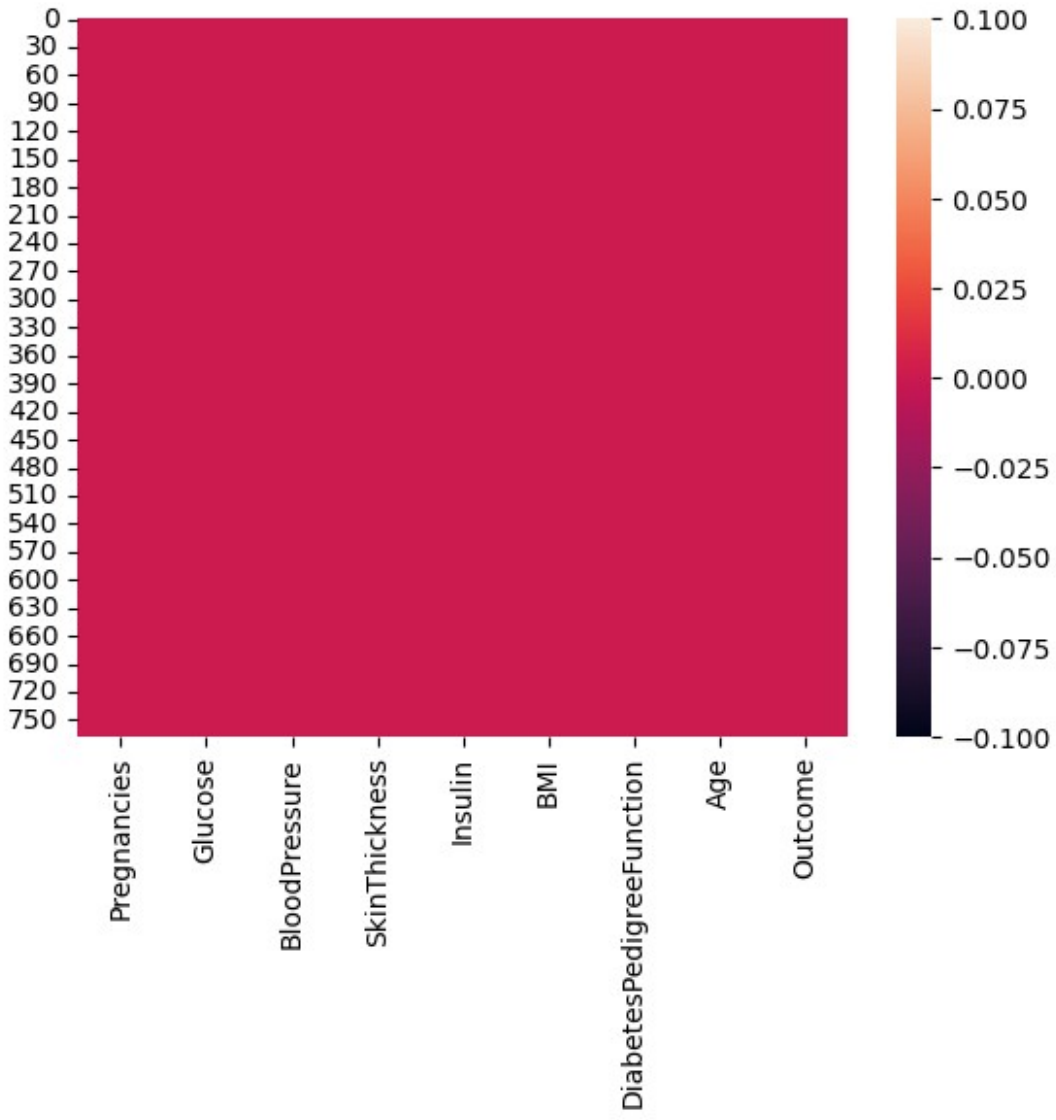
```
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              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
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

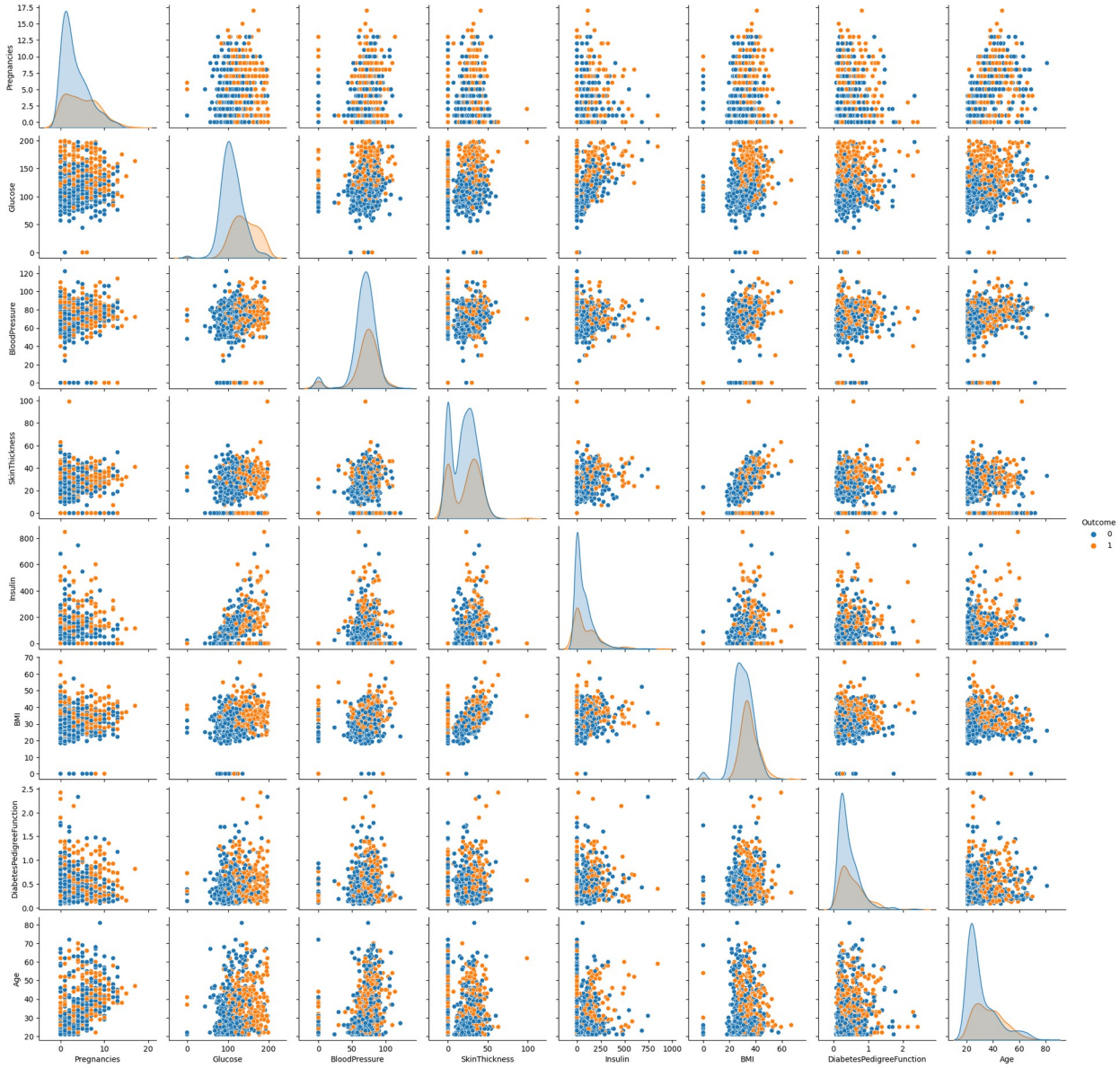
```
sns.heatmap(df.isnull())
```

```
<Axes: >
```

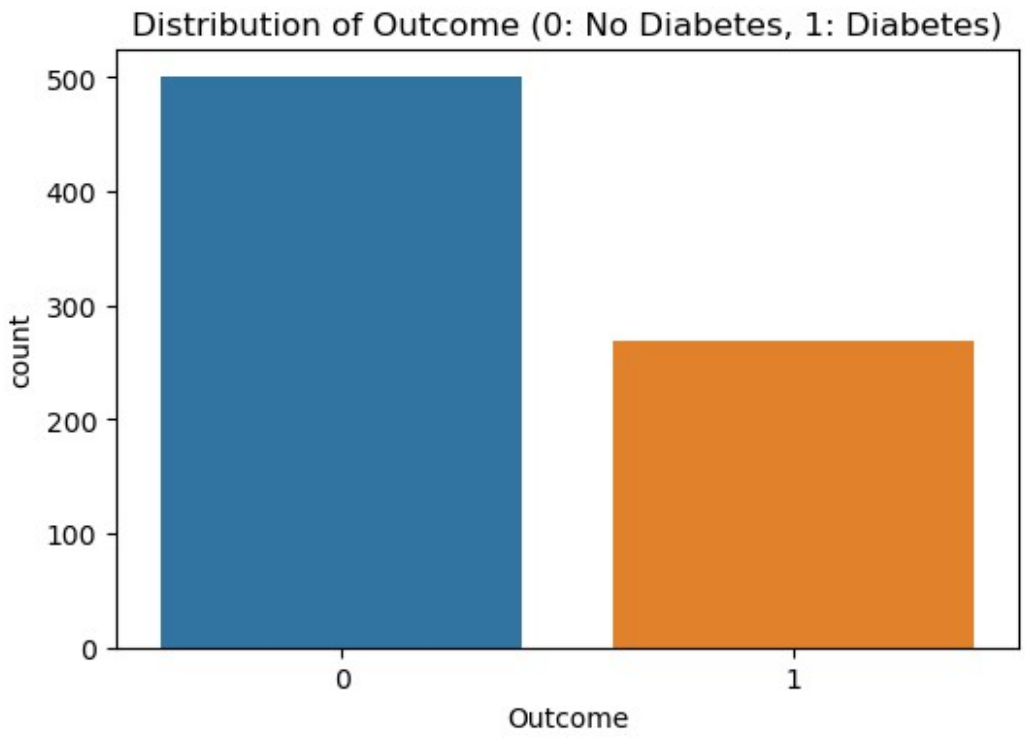



```
sns.pairplot(df,hue="Outcome")
plt.show()
```

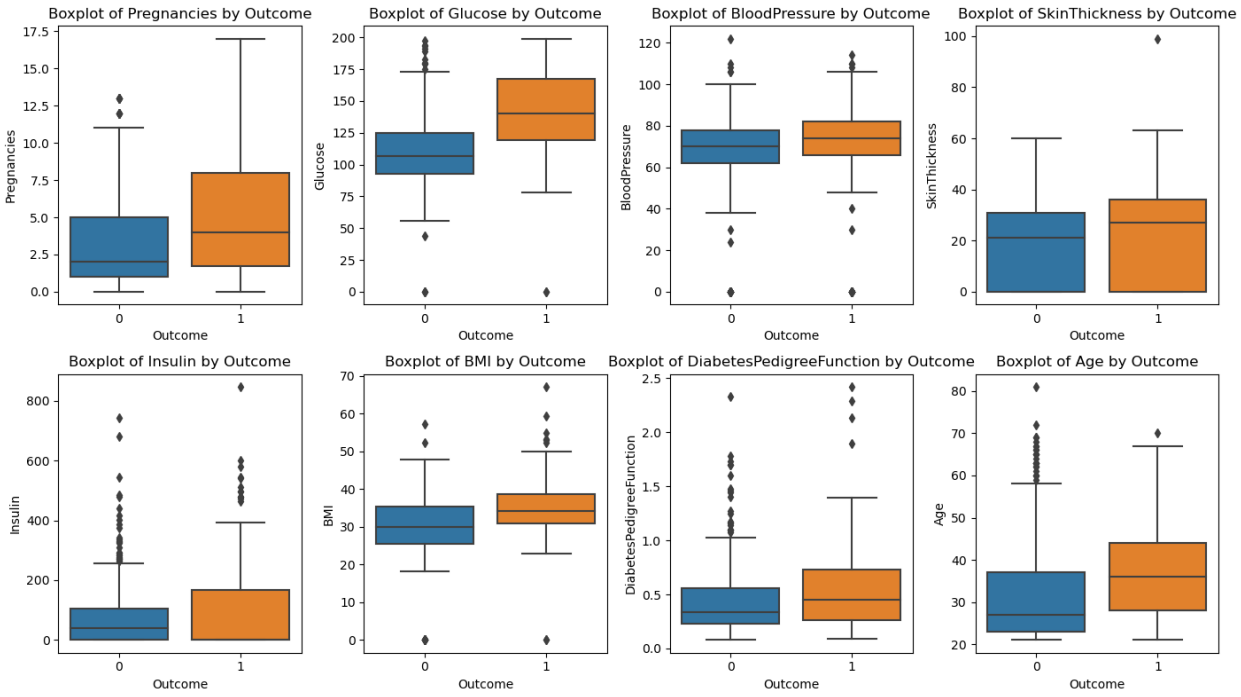
```
C:\Users\1006y\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




```
# Distribution of the Target Variable
plt.figure(figsize=(6, 4))
sns.countplot(x='Outcome', data=df)
plt.title('Distribution of Outcome (0: No Diabetes, 1: Diabetes)')
plt.show()
```




```
# Boxplot for each variable grouped by Outcome
plt.figure(figsize=(14, 8))
for i, column in enumerate(df.columns[:-1], 1):
    plt.subplot(2, 4, i)
    sns.boxplot(x='Outcome', y=column, data=df)
    plt.title(f'Boxplot of {column} by Outcome')
plt.tight_layout()
plt.show()
```




```
Glu = df.Glucose.mean()
Bld =df.BloodPressure.mean()
Ski =df.SkinThickness.mean()
Ins = df.Insulin.mean()
Bmi = df.BMI.mean()

print(f"Average of Glucose is: {Glu}")
print(f"Average of BloodPressure is: {Bld}")
print(f"Average of SkinThickness is: {Ski}")
print(f"Average of Insulin is: {Ins}")
print(f"Average of BMI is: {Bmi}")
```

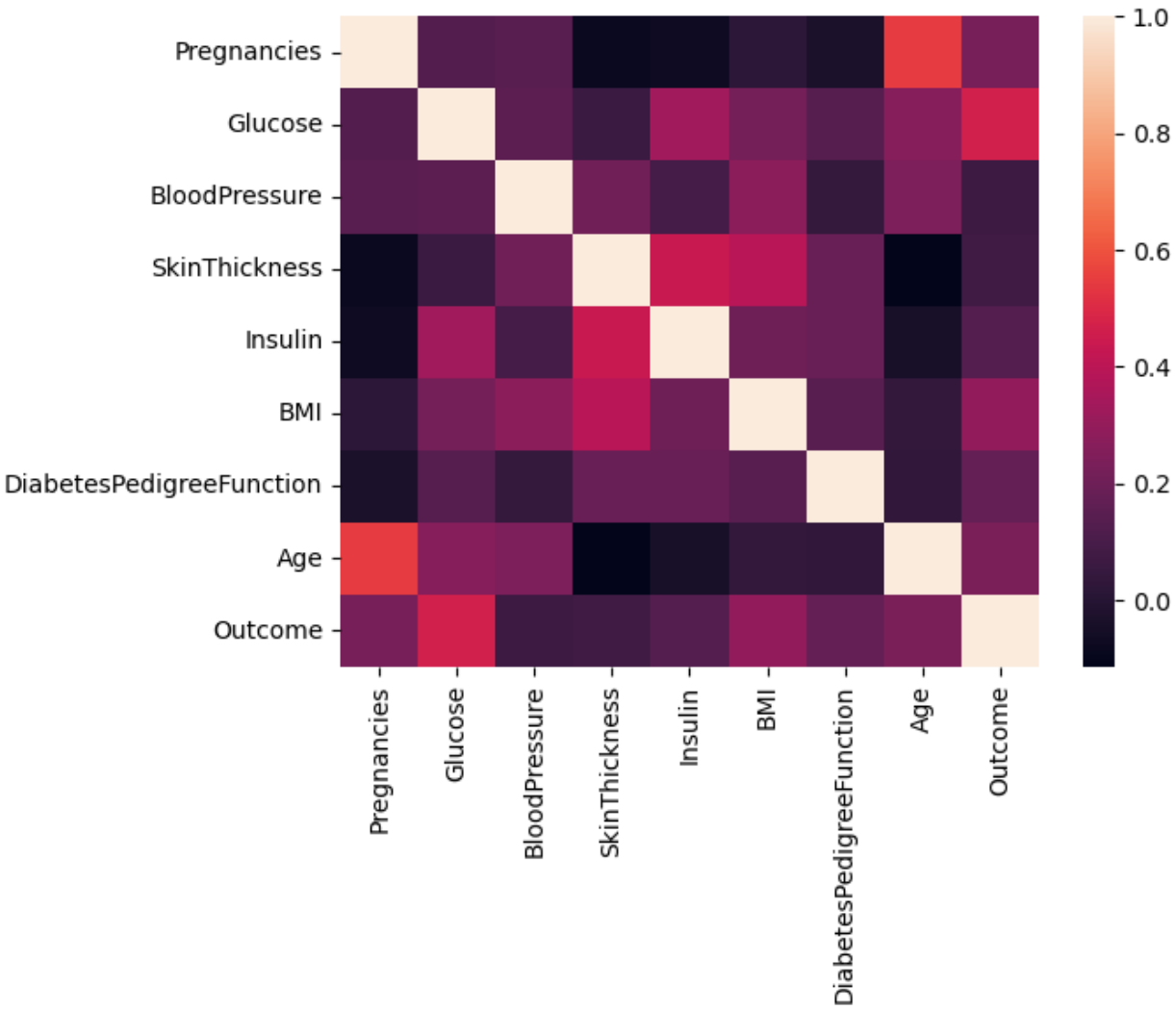
```
Average of Glucose is: 120.89453125
Average of BloodPressure is: 69.10546875
Average of SkinThickness is: 20.536458333333332
Average of Insulin is: 79.79947916666667
Average of BMI is: 31.992578124999998
```

```
# Heatmap of Selected Variables
selected_vars = ['Pregnancies', 'Glucose', 'BloodPressure',
'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age',
'Outcome']
selected_df = df[selected_vars]
plt.figure(figsize=(10, 8))
sns.heatmap(selected_df.corr(), annot=True, cmap='BuPu', fmt=".2f")
plt.title('Correlation Heatmap for Selected Variables')
plt.show()
```


```
correlation=df.corr()
print(correlation)
```

```
                          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

sns.heatmap(correlation)

<Axes: >
```




```
X=df.drop("Outcome",axis=1)
Y=df['Outcome']

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)

model=LogisticRegression()
model.fit(X_train,Y_train)
```

```
C:\Users\1006y\anaconda3\Lib\site-packages\sklearn\linear_model\
_logistic.py:460: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
prediction_model=model.predict(X_test)
```

```
print(prediction_model)
```

```
[0 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0
0 0
 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0
0 0
 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0
0 0
 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0
1 0
 0 0 0 0 0 0]
```

```
accuracy=accuracy_score(prediction_model,Y_test)
print(accuracy*100)
```

```
75.32467532467533
```

```
# Prediction
y_pred = model.predict(X_test)
```

```
# Classification report
print(f'Classification Report: \n{classification_report(Y_test,
y_pred)}')
```

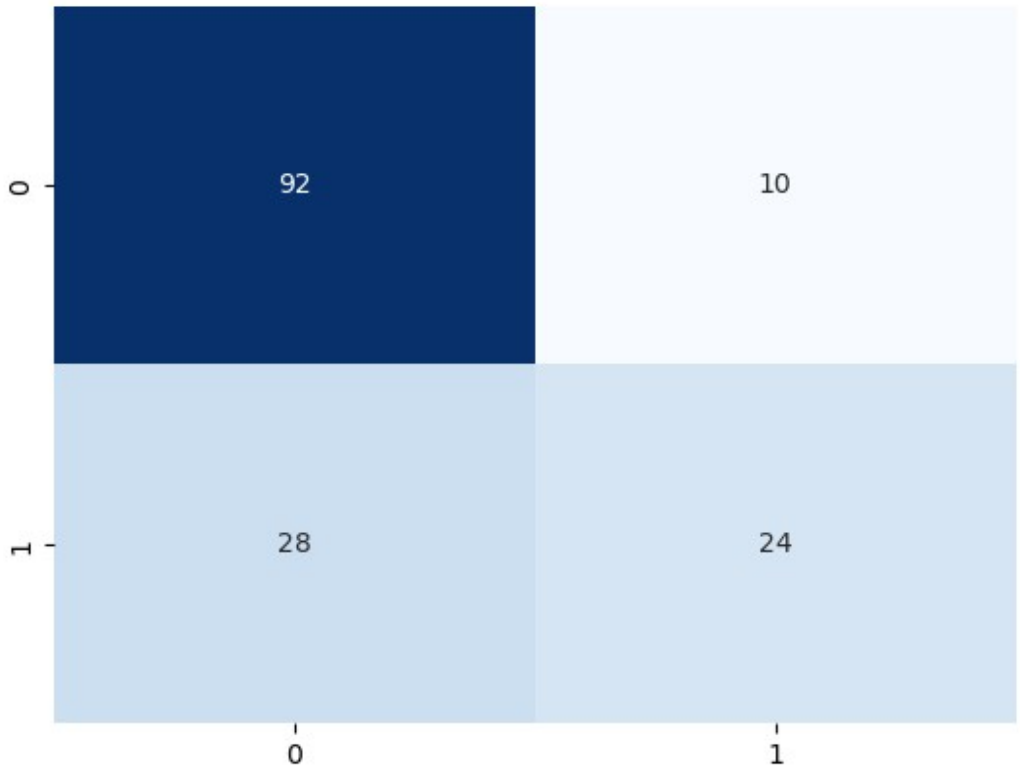
```
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.90      0.83       102
           1       0.71      0.46      0.56        52

    accuracy                           0.75       154
   macro avg       0.74      0.68      0.69       154
weighted avg       0.75      0.75      0.74       154
```

```
# F1 score
print(f"F1 Score : {f1_score(Y_test, y_pred)*100}")
```

```
F1 Score : 55.81395348837211
```

```
# Confusion matrix
cf_matrix = confusion_matrix(Y_test, y_pred)
sns.heatmap(cf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
```

```
<Axes: >
```




```
# Take user input for each value in the input_data tuple
pregnancies = float(input("Enter the number of Pregnancies: "))
glucose = float(input("Enter Glucose level: "))
blood_pressure = float(input("Enter Blood Pressure: "))
skin_thickness = float(input("Enter Skin Thickness: "))
insulin = float(input("Enter Insulin level: "))
bmi = float(input("Enter BMI: "))
diabetes_pedigree = float(input("Enter Diabetes Pedigree Function: "))
age = float(input("Enter Age: "))

# Create a numpy array from the input data
input_data = (pregnancies, glucose, blood_pressure, skin_thickness,
insulin, bmi, diabetes_pedigree, age)
input_data_as_numpy_array = np.asarray(input_data)

# Reshape the input data for prediction
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)

# Standardize the input data
scaler = StandardScaler()
std_data = scaler.fit_transform(input_data_reshaped)

# Make a prediction on the input data
```

```
prediction = model.predict(std_data)

# Print the prediction
if prediction[0] == 0:
    print('The person is not diabetic')
else:
    print('The person is diabetic')
```

```
Enter the number of Pregnancies: 2
Enter Glucose level: 100
Enter Blood Pressure: 70
Enter Skin Thickness: 30
Enter Insulin level: 10
Enter BMI: 25
Enter Diabetes Pedigree Function: 0.35
Enter Age: 30
The person is not diabetic

C:\Users\1006y\anaconda3\Lib\site-packages\sklearn\base.py:464:
UserWarning: X does not have valid feature names, but
LogisticRegression was fitted with feature names
  warnings.warn(
```