diabetes-prediction-project

January 11, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[4]:
```
# Import file
df = pd.read_csv("diabetes.csv")
```

[5]:
```
df.head()
```

[5]:

	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

[6]:
```
# Understanding Data and Data cleaning

df.shape
```

[6]:
```
(768, 9)
```

[7]:
```
df.describe()
```

[7]:

	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]:
```
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
```

[9]:
```
# checking null values

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

[9]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
```

```
Age                         0
Outcome                     0
dtype: int64
```

[10]:
```
# checking duplicate value in dataset
df[df.duplicated()]
```

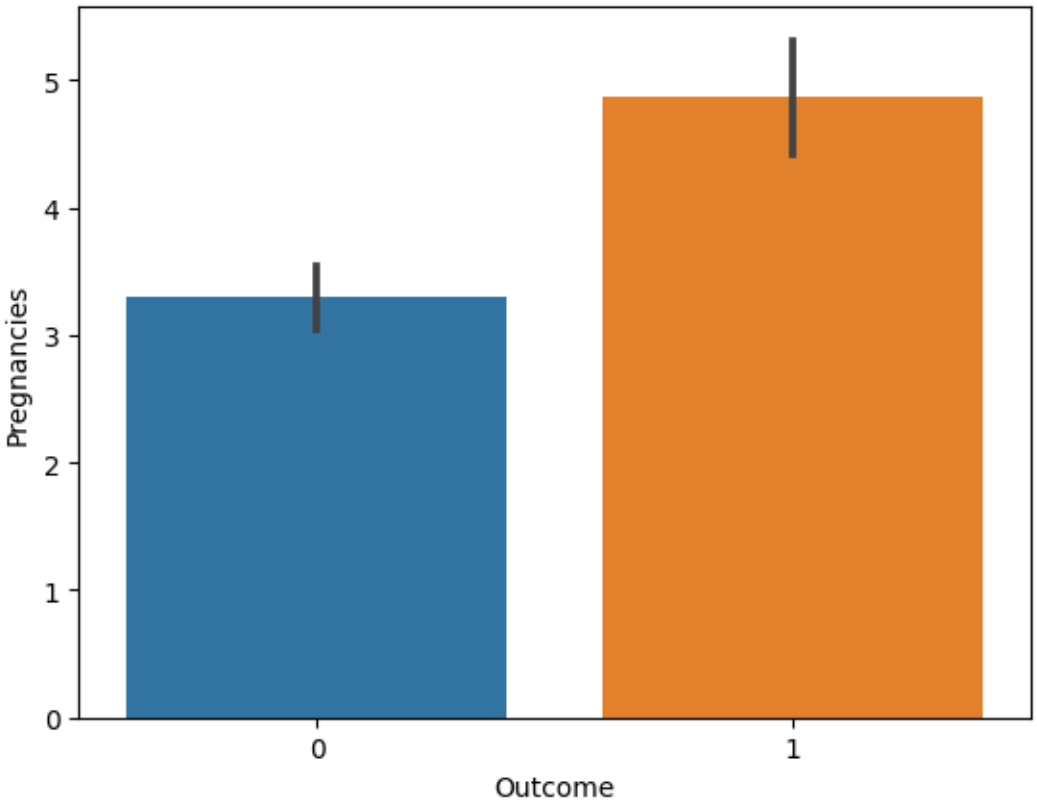
[10]:
```
Empty DataFrame
Columns: [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI,
DiabetesPedigreeFunction, Age, Outcome]
Index: []
```

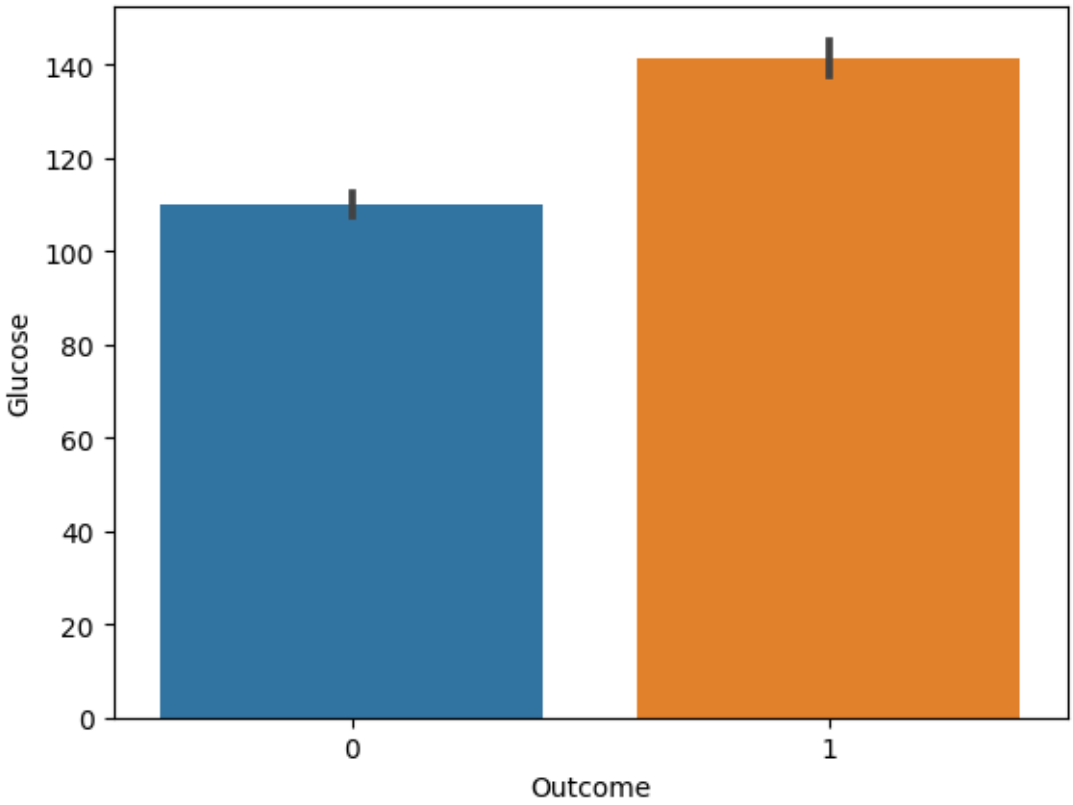
[11]:
```
# Calculating probability
df['Outcome'].value_counts()/len(df)*100
```

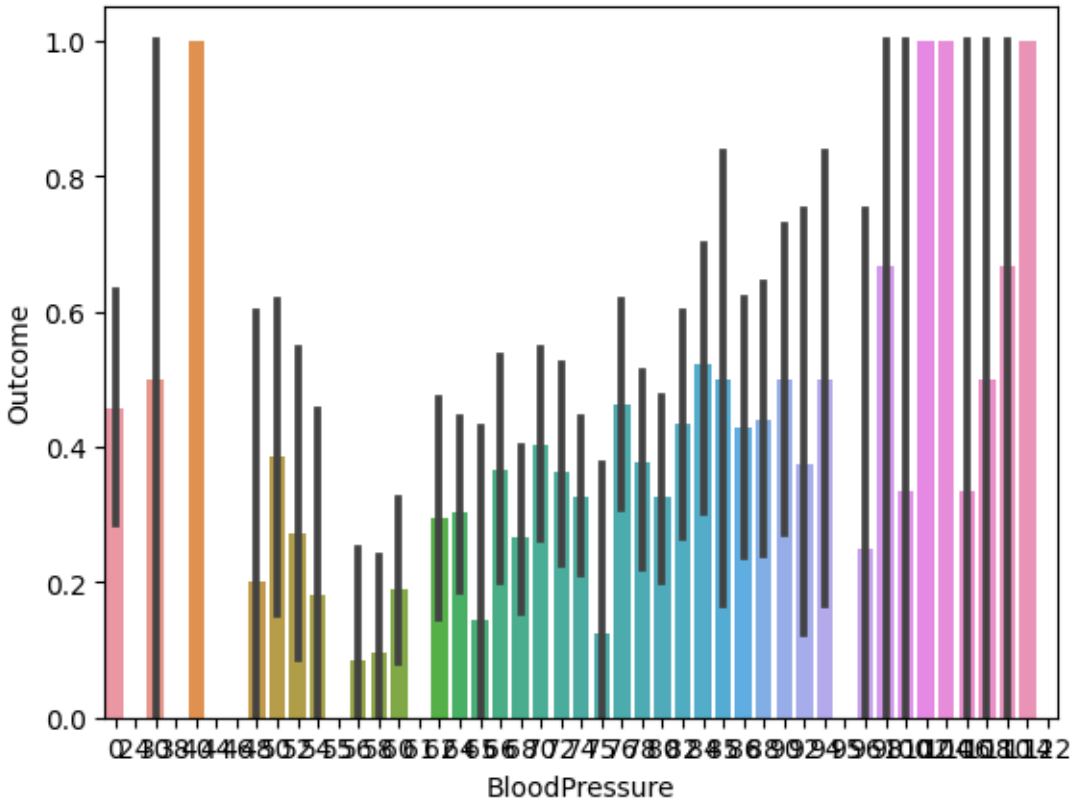
[11]:
```
0    65.104167
1    34.895833
Name: Outcome, dtype: float64
```

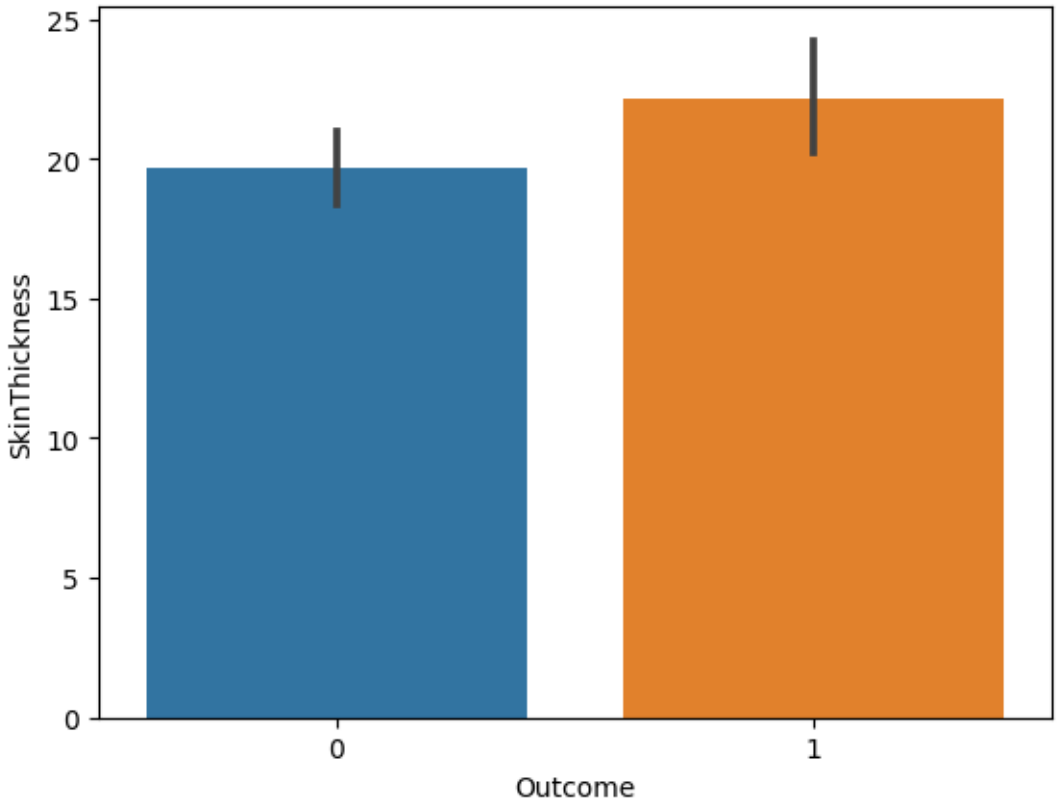
[ ]:
```
# EXPLORATORY DATA ANALYSIS
```

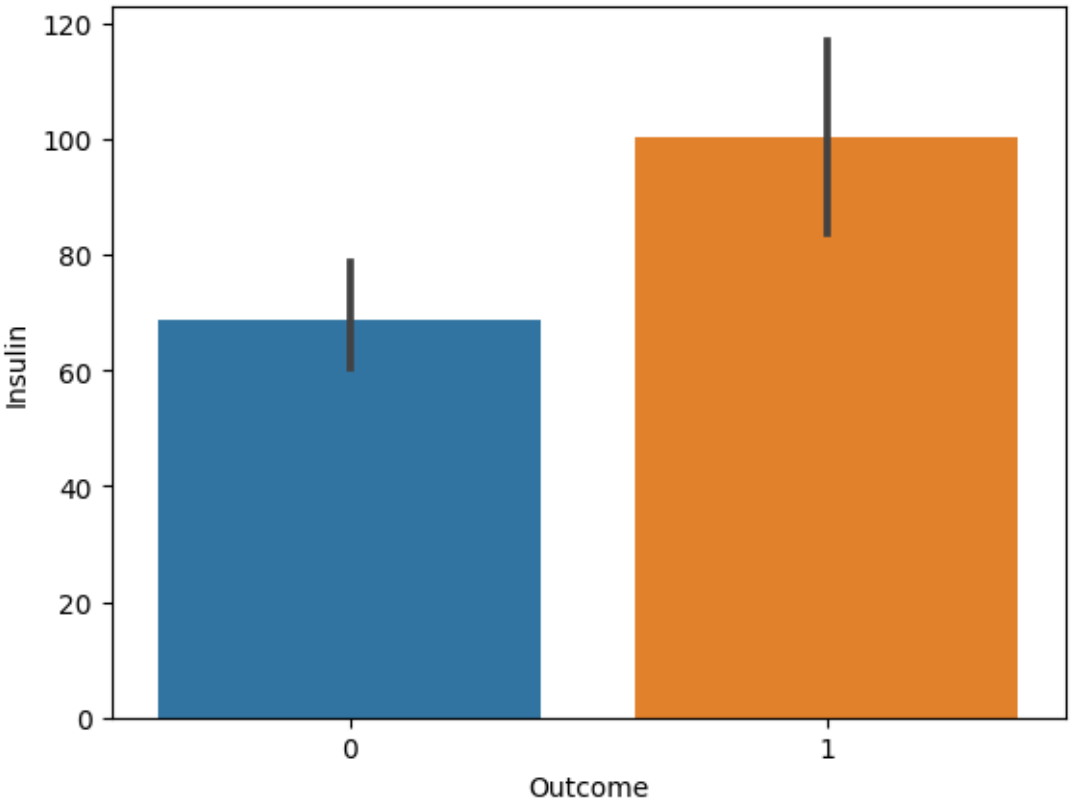
[12]:
```
plt.figure()
sns.barplot(x='Outcome', y='Pregnancies',data=df)
plt.show()
sns.barplot(x='Outcome' , y='Glucose' , data=df)
plt.show()
plt.figure()
sns.barplot(y='Outcome', x='BloodPressure', data=df)
plt.figure()
sns.barplot(x='Outcome', y='SkinThickness' , data=df)
plt.show()
plt.figure()
sns.barplot(x='Outcome', y='Insulin' , data=df)
plt.show()
plt.figure()
sns.barplot(x='Outcome', y='BMI' , data=df)
plt.show()
plt.figure()
sns.barplot(x='Outcome', y='DiabetesPedigreeFunction' , data=df)
plt.figure()
sns.barplot(x='Outcome', y='Age' , data=df)
plt.show()
```

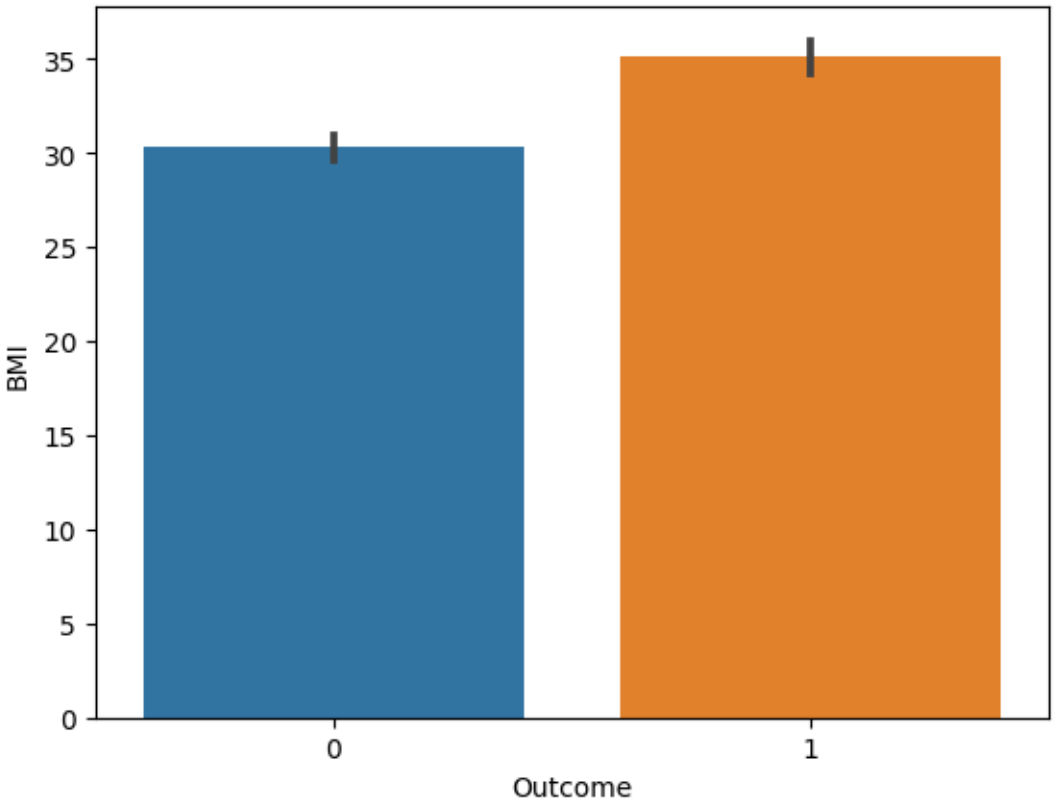

Pregnancies
0
1
2
3
4
5
0
1
Outcome

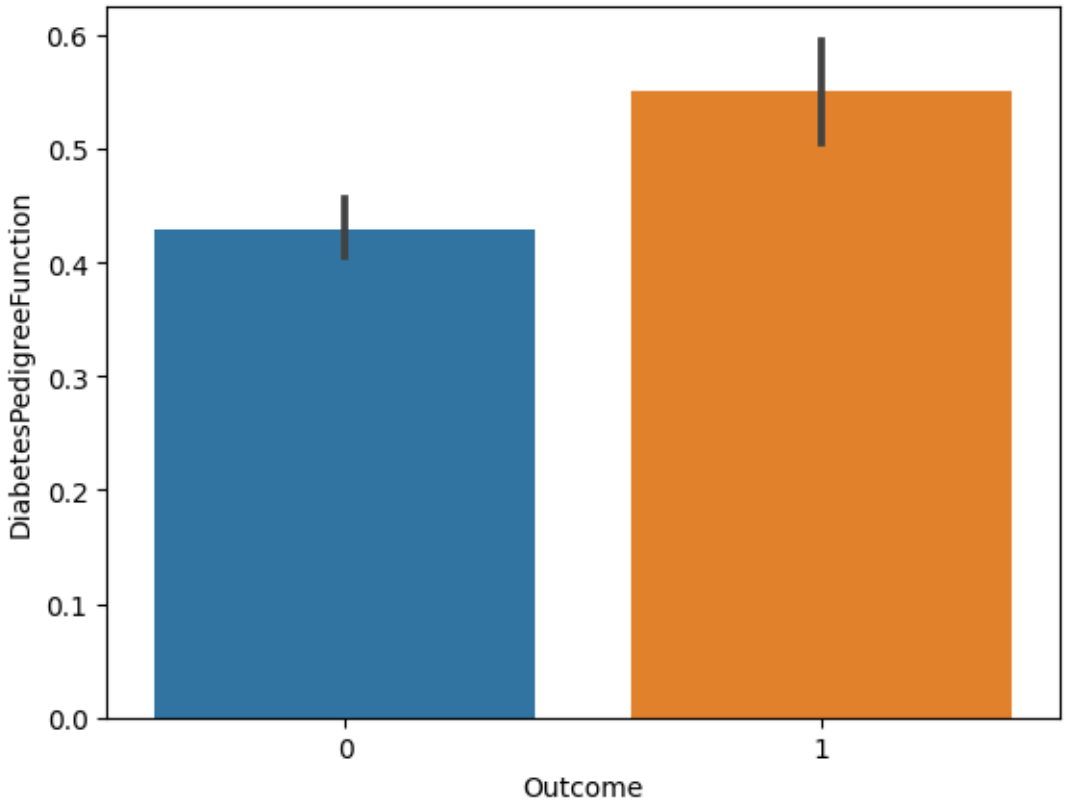

140
120
100
80
60
40
20
0
Glucose
0
1
Outcome

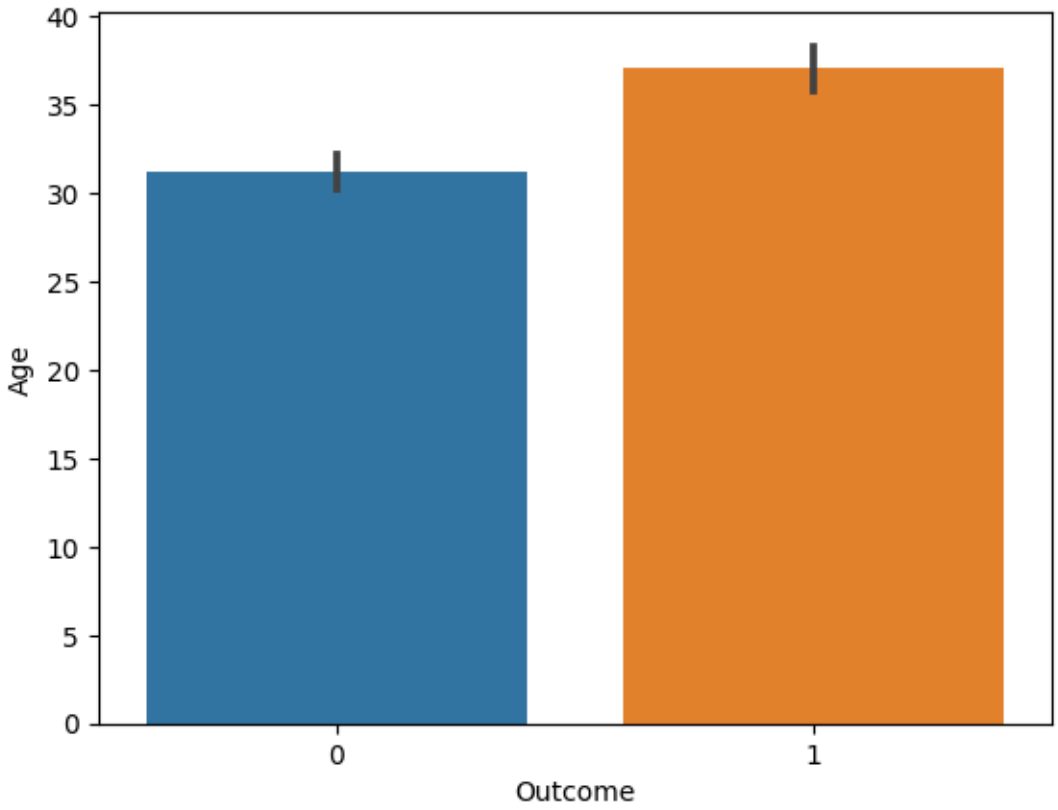

Outcome
1.0
0.8
0.6
0.4
0.2
0.0
BloodPressure


25
20
15
10
5
0
SkinThickness
0
1
Outcome


120
100
80
60
40
20
0
Insulin
0
1
Outcome

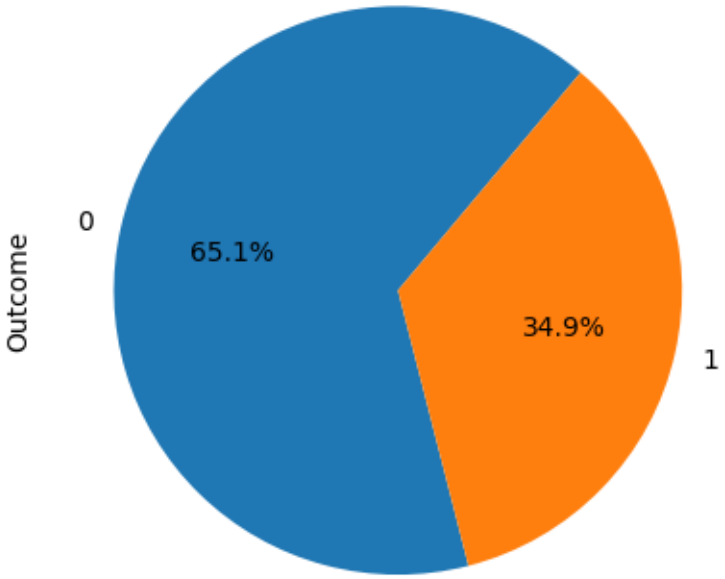

35
30
25
20
15
10
5
0
BMI
0
1
Outcome


0.6
0.5
0.4
0.3
0.2
0.1
0.0
DiabetesPedigreeFunction
0
1
Outcome



[13]:
```
df['Outcome'].value_counts().plot.pie(startangle=50, autopct='%1.1f%%')

plt.show()
```



[ ]:
```
# Data preprocessing
```

[14]:
```
df.isnull().sum()
```

[14]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[16]:
```
# Replacing Nan with mean values

df["Glucose"].fillna(df["Glucose"].mean(), inplace = True)
df["BloodPressure"].fillna(df["BloodPressure"].mean(), inplace = True)
df["SkinThickness"].fillna(df["SkinThickness"].mean(), inplace = True)
df["Insulin"].fillna(df["Insulin"].mean(), inplace = True)
df["BMI"].fillna(df["BMI"].mean(), inplace = True)
```

[17]:
```
df.describe().T
```

[17]:

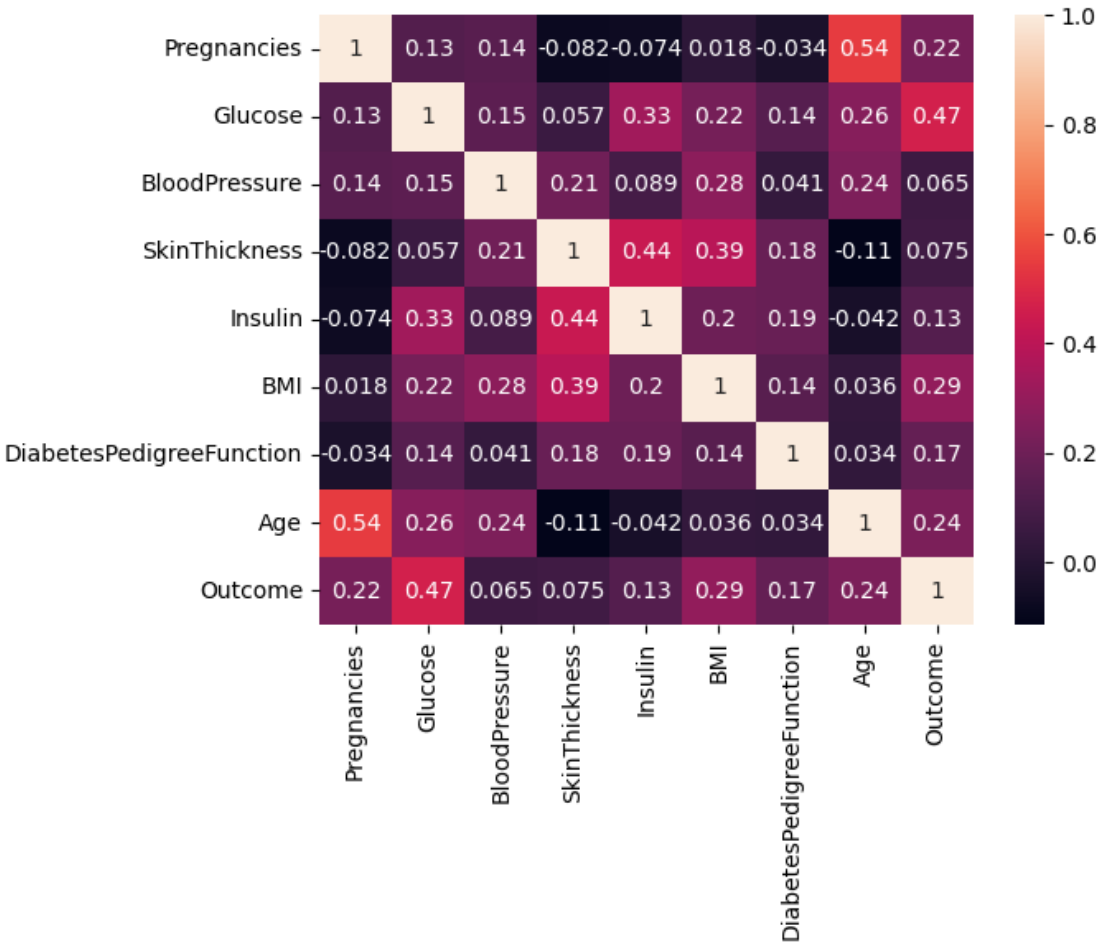
	count	mean	std	min	25%	\
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	
Glucose	768.0	120.894531	31.972618	0.000	99.00000	
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	
Insulin	768.0	79.799479	115.244002	0.000	0.00000	
BMI	768.0	31.992578	7.884160	0.000	27.30000	
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	
Age	768.0	33.240885	11.760232	21.000	24.00000	
Outcome	768.0	0.348958	0.476951	0.000	0.00000	

	50%	75%	max
Pregnancies	3.0000	6.00000	17.00
Glucose	117.0000	140.25000	199.00
BloodPressure	72.0000	80.00000	122.00
SkinThickness	23.0000	32.00000	99.00
Insulin	30.5000	127.25000	846.00
BMI	32.0000	36.60000	67.10
DiabetesPedigreeFunction	0.3725	0.62625	2.42
Age	29.0000	41.00000	81.00
Outcome	0.0000	1.00000	1.00

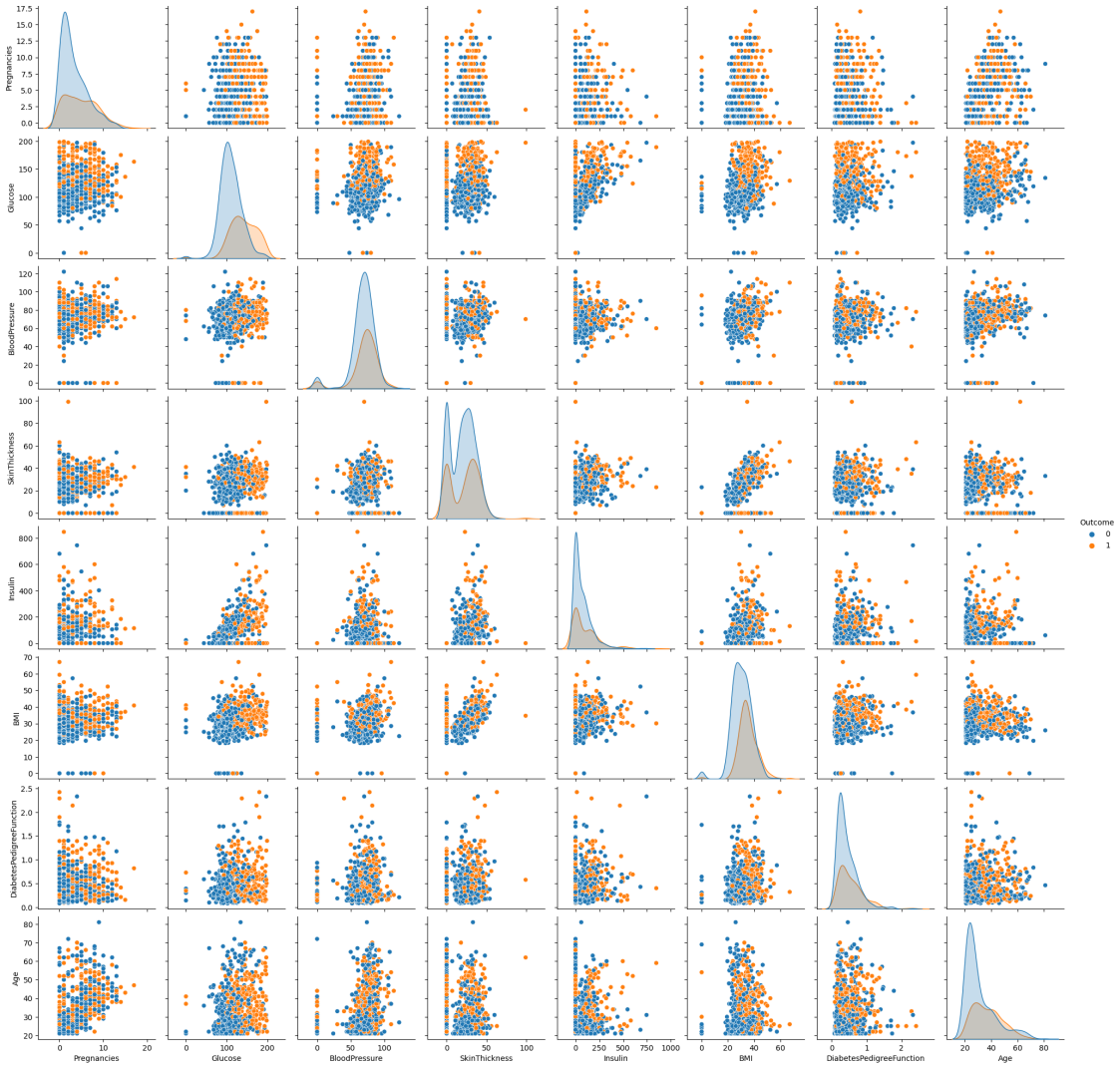
[18]:
```
df.isnull().sum()
```

[18]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[19]:
```
#Heatmap
sns.heatmap(df.corr() , annot = True)
plt.show()
```



[20]:
```
# Pairplot
sns.pairplot(data = df, hue = 'Outcome')
plt.show()
```



[21]:
```
# Replacing zero with Nan
df_N = df
df_N.isnull().sum()
```

[21]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[23]:
```
# Logistic Regression
y = df_N['Outcome']
X = df_N.drop('Outcome', axis=1)
```

[24]:
```
# Count of NaN
df_N.isnull().sum()
```

[24]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[26]:
```
# Replacing NaN with mean values
df_N["Glucose"].fillna(df_N["Glucose"].mean(), inplace = True)
df_N["BloodPressure"].fillna(df_N["BloodPressure"].mean(), inplace = True)
df_N["SkinThickness"].fillna(df_N["SkinThickness"].mean(), inplace = True)
df_N["Insulin"].fillna(df_N["Insulin"].mean(), inplace = True)
df_N["BMI"].fillna(df_N["BMI"].mean(), inplace = True)
```

[28]:
```
df_N.isnull().sum()
```

[28]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[32]:
```
# Logistic Regression

y = df_N['Outcome']
X = df_N.drop('Outcome', axis=1)

# Splitting X and Y
from sklearn.model_selection    import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2,
random_state = 42, stratify = df_N['Outcome'] )
```

[34]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)
y_predict = model.predict(X_test)
```

[35]:
```
y_predict
```

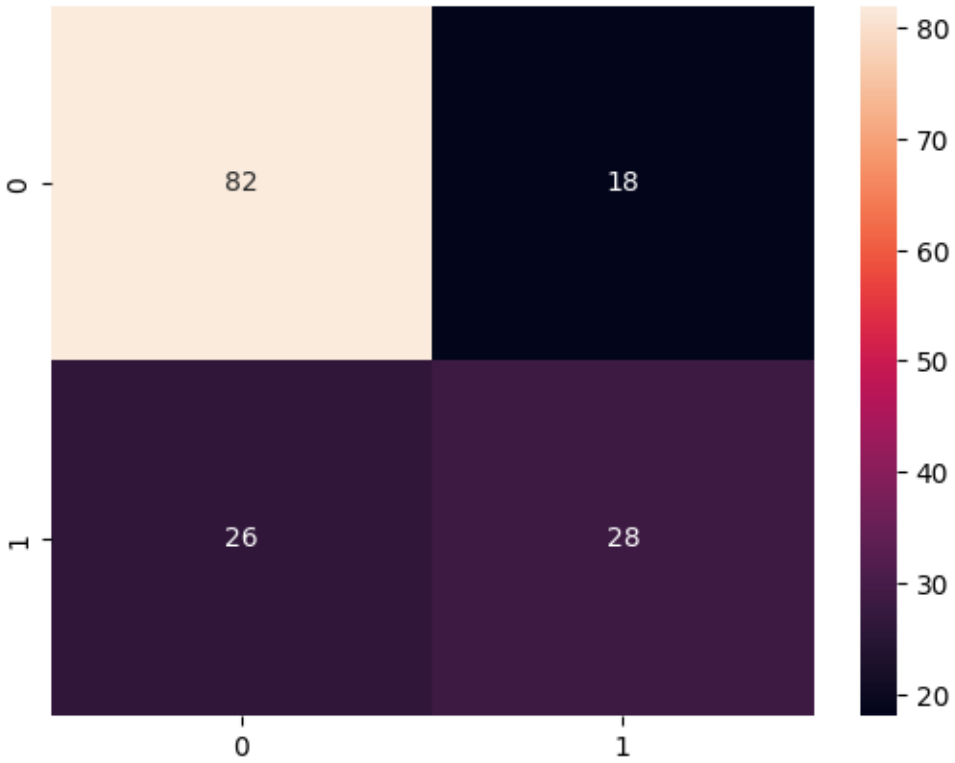
[35]:
```
array([1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0])
```

[36]:
```
# Confusion Matrix - comparing with the actual outcome
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, y_predict)
cm
```

[36]:
```
array([[82, 18],
       [26, 28]])
```

[37]:
```
# Heatmap of Confusion matrix
sns.heatmap(pd.DataFrame(cm), annot=True)
```

[37]:
```
<Axes: >
```



[39]:
```
from sklearn.metrics import accuracy_score    #compare the predictive true␣
↪label under correct predictions
```

[40]:
```
accuracy =accuracy_score(Y_test, y_predict)
accuracy
```

[40]: 0.7142857142857143

[44]:
```
y_predict = model.predict([[1,148,72,35,79.799,33.6,0.627,50]])
print(y_predict)
if y_predict==1:
   print("Diabetic")
else:
  print("Non Diabetic")
```

```
[1]
Diabetic
```

[ ]: