Importing Libraries

In [93]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [94]:
```
df = pd.read_csv('diabetes.csv')
```

In [95]:
```
df.head()
```

Out[95]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
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	

This is a Diabetes dataset, we are using to understand its importance to the significance of health-related features and the role they might play in predicting the onset of diabetes. Lets dig deeper into these features.

Pregnancies: This feature represents the number of times an individual has been pregnant. Its important as several studies have indicated that the risk of developing diabetes increases with the number of pregnancies.

Glucose: This represents the plasma glucose concetration of a 2 hour oral glucose tolerance. Elevated levels of glucose in the blood, or hyperglycemia, is a common effect of uncontrolled diabetes and overtime might result to serious damage to many of the body's systems, especially the nerves and blood vessels.

BloodPressure: This feature signifies diastolic blood pressure(in mm Hg). Persistent high blood pressure, also known as hypertension, can lead to various health problems including heart disease, kidney disease, stroke, and can also be a risk factor for the development of diabetes.

SkinThickness: This refers to the triceps skin fold thickness(in mm). It's a measure of body fat, and higher values may indicate overwieght or obesity, which are known risk factors for diabetes.

Insulin: This is the 2-hour serum insulin(in mu U/ml). Insulin is a hormone that regulates blood sugar, and problems with insulin production or function can lead to the development of diabetes.

BMI: This feature is the Body Mass Index (weight in kg/(height in m)^2). Like skin thickness, it's a measure of body fat, and high BMI values (overwieght or obesity) are associated with an increased risk of diabetes.

DiabetesPedigreeFucntions: This is a function that scores likelihood of diabetes based on family history. It's based on the premise that the genetic predispostion to the disease can be quantified, and that a family history of the disease increases the risk.

Age: This represents the age in years. Aging is associated with changes in body compostion, insulin secretion and action, and glucose metabolism, all of which can increase the risk of developing diabetes.

Outcome: This is the class variable(0 or 1). In this dataset, 268 of 768 instances are 1(representing diabetes), and the rest are 0(no diabetes). This is our target variable which we aim to predict based on the other features.

Checking for null values

In [96]:
```
df.isnull().sum()
```

Out[96]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [97]:
```
# Loop through each column and count the number of distinct val

for column in df.columns:
    num_distinct_values = len(df[column].unique())
    print(f"{column}: {num_distinct_values} distinct values")
```

```
Pregnancies: 17 distinct values
Glucose: 136 distinct values
BloodPressure: 47 distinct values
SkinThickness: 51 distinct values
Insulin: 186 distinct values
BMI: 248 distinct values
DiabetesPedigreeFunction: 517 distinct values
Age: 52 distinct values
Outcome: 2 distinct values
```

In [98]:
```
df.describe().style.format("{:.2f}")
```

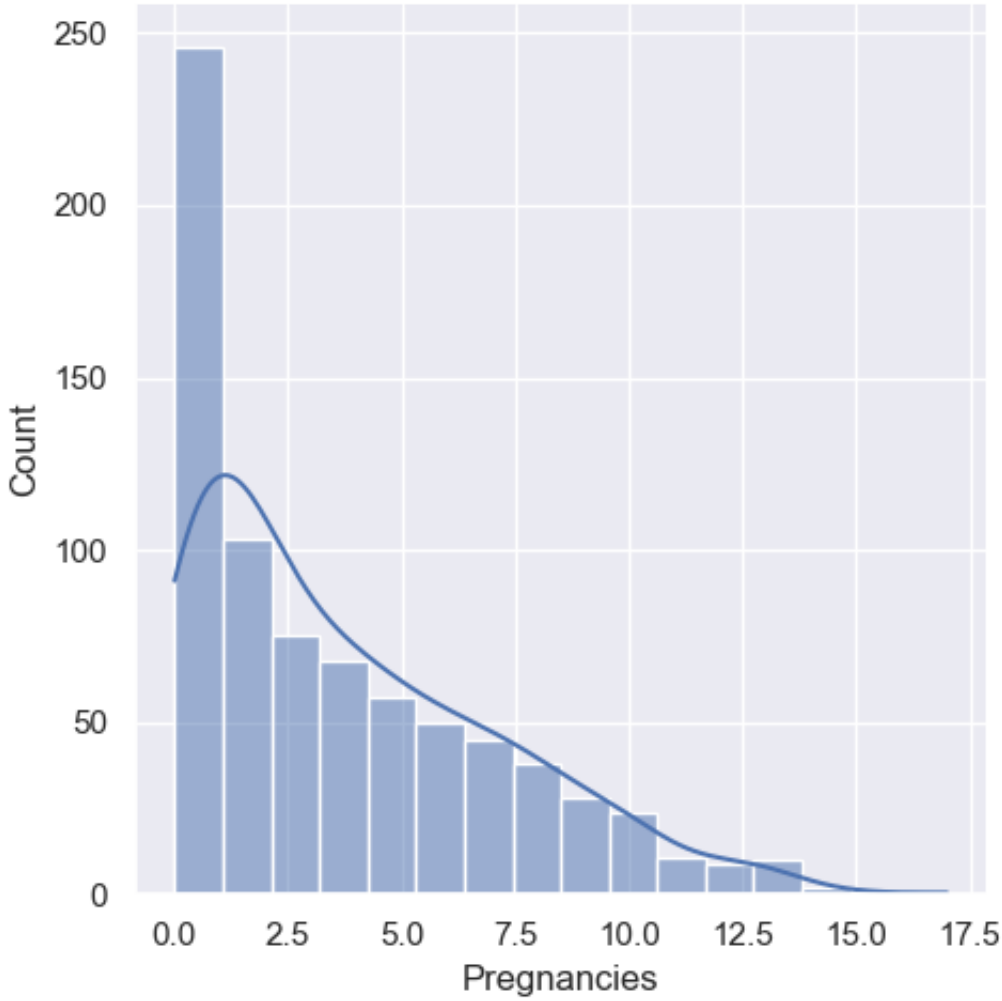
Out[98]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
count	768.00	768.00	768.00	768.00	768.00	768.00	
mean	3.85	120.89	69.11	20.54	79.80	31.99	
std	3.37	31.97	19.36	15.95	115.24	7.88	
min	0.00	0.00	0.00	0.00	0.00	0.00	
25%	1.00	99.00	62.00	0.00	0.00	27.30	
50%	3.00	117.00	72.00	23.00	30.50	32.00	
75%	6.00	140.25	80.00	32.00	127.25	36.60	
max	17.00	199.00	122.00	99.00	846.00	67.10	

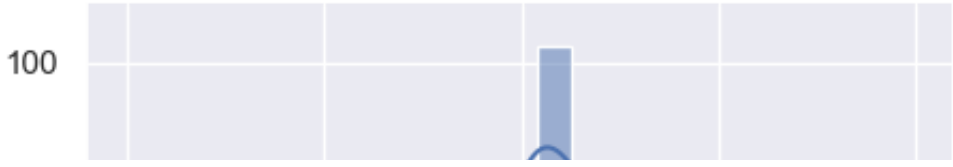
1. This dataset consists of 768 rows and 8 columns.
2. The target variable in Outcome, which contains categorical binary values 0 and 1.
3. The variables other than Outcome are also numberical.
4. They are technically no missing values because of lack of NaN values, however when we examine closely, some 0's in the dataset indicate they are actually missing values.

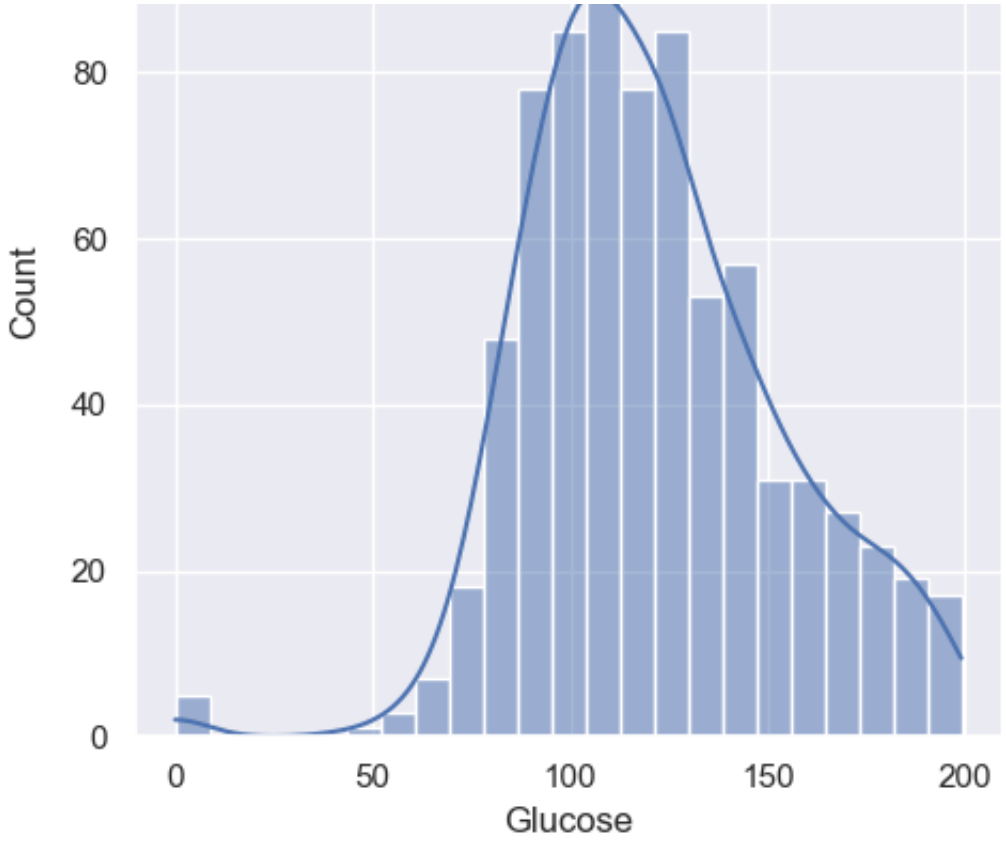
In [99]:
```
# Explore the distribution of numerical variables

for col in df.columns:
    plt.figure(figsize=(6,4))
    sns.displot(data=df[col], kde=True)
    plt.show()
```

```
<Figure size 600x400 with 0 Axes>
```

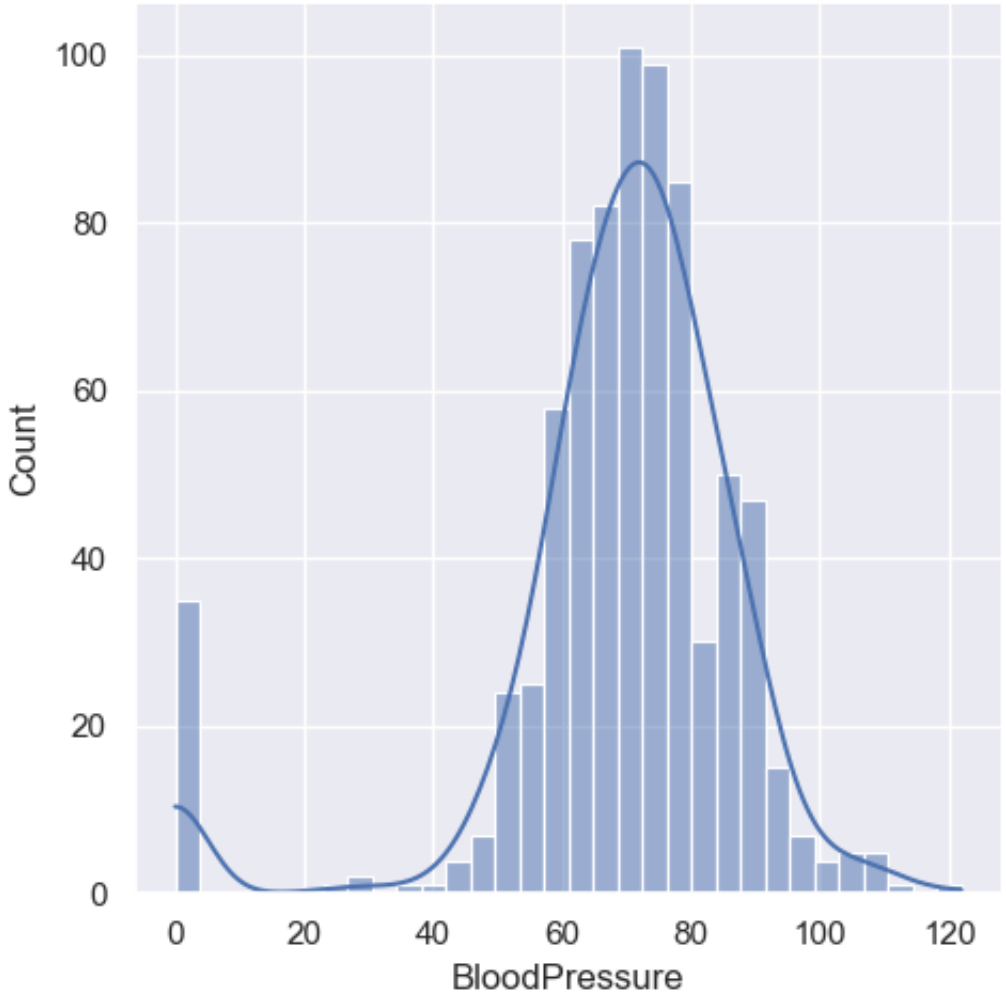



```
<Figure size 600x400 with 0 Axes>
```

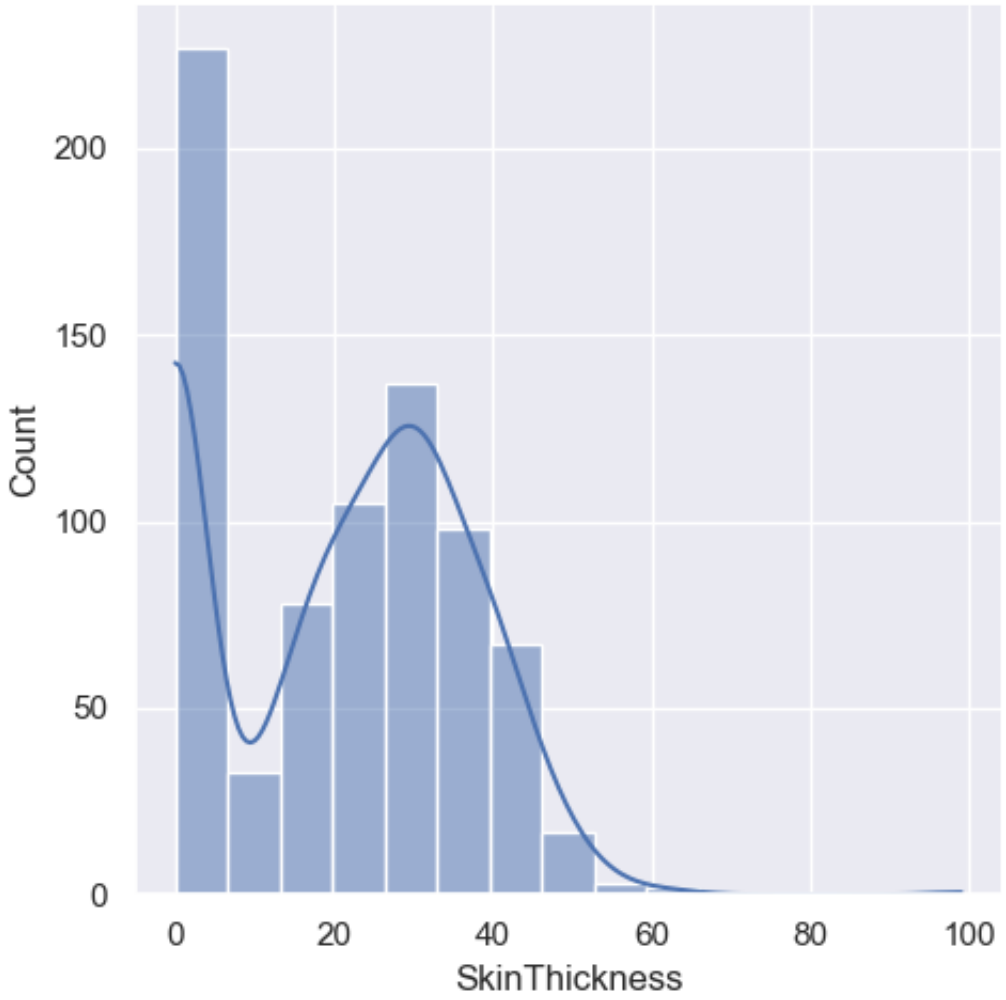




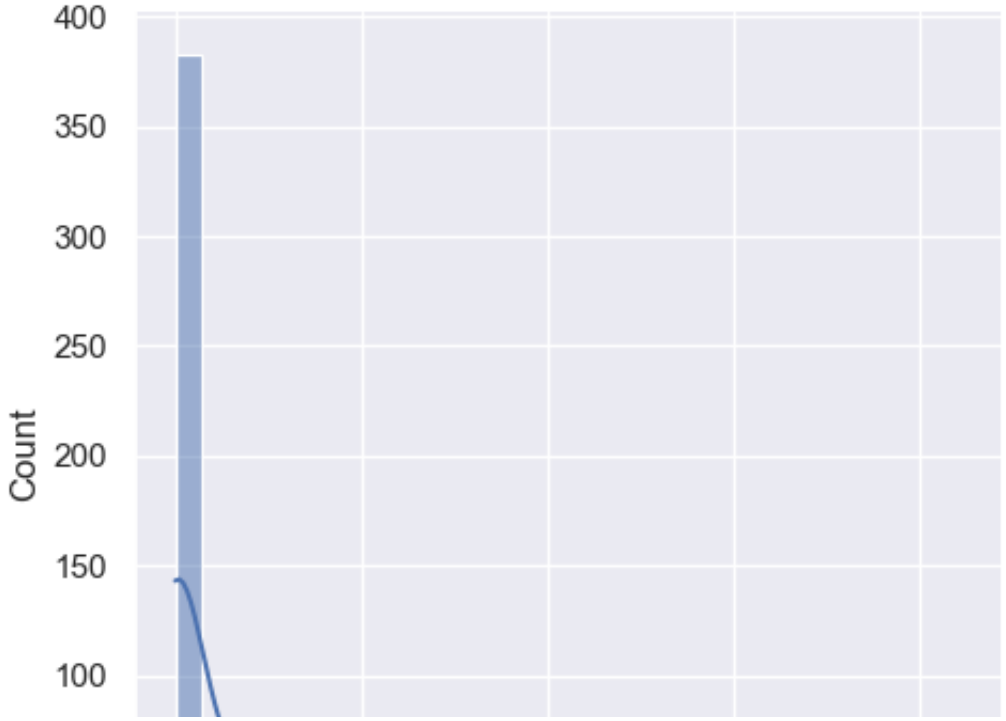

```
<Figure size 600x400 with 0 Axes>
```

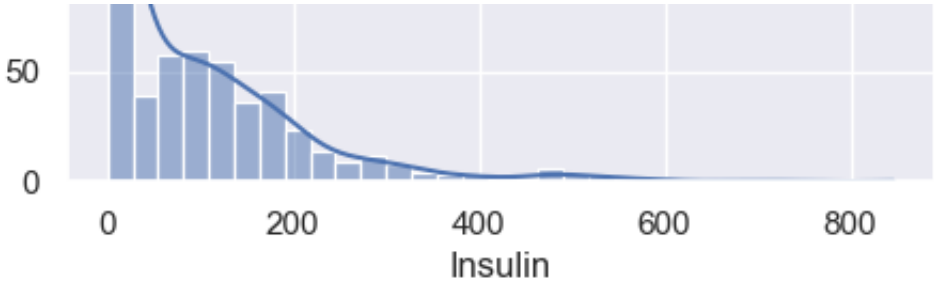


```
<Figure size 600x400 with 0 Axes>
```

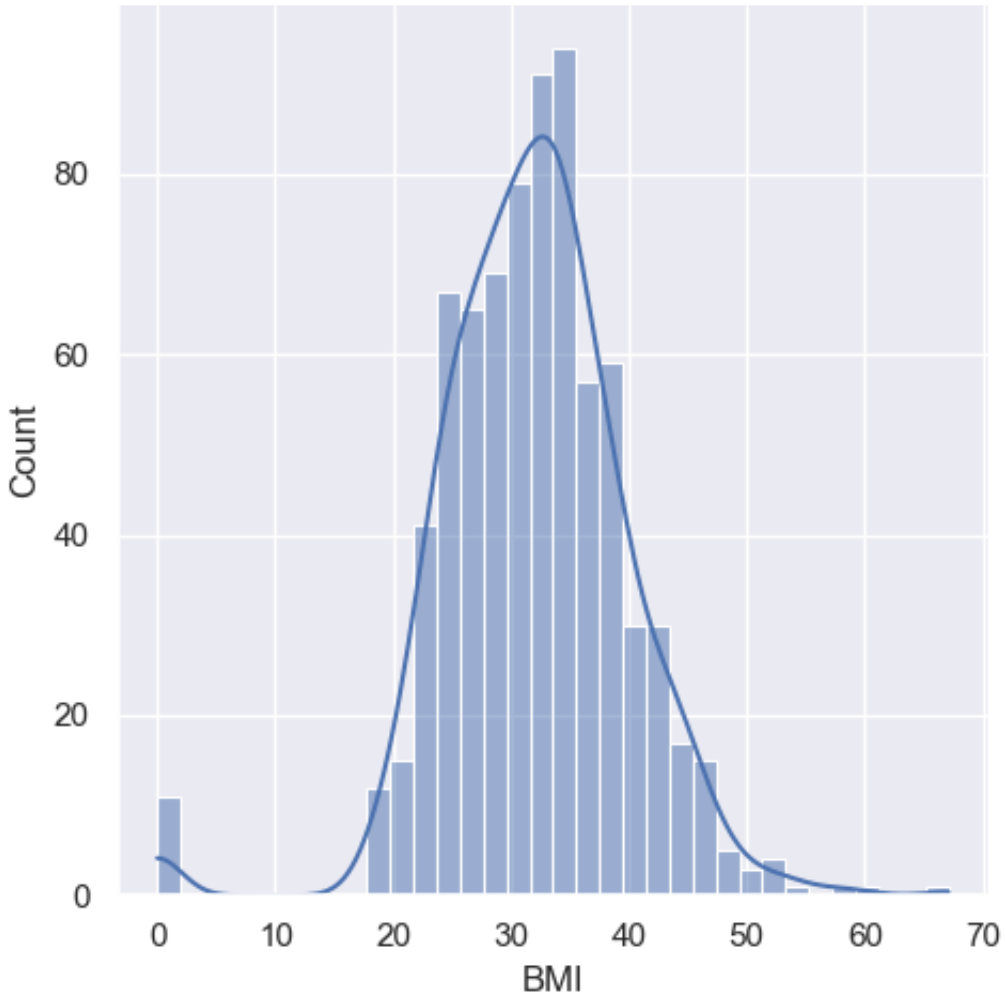



```
<Figure size 600x400 with 0 Axes>
```

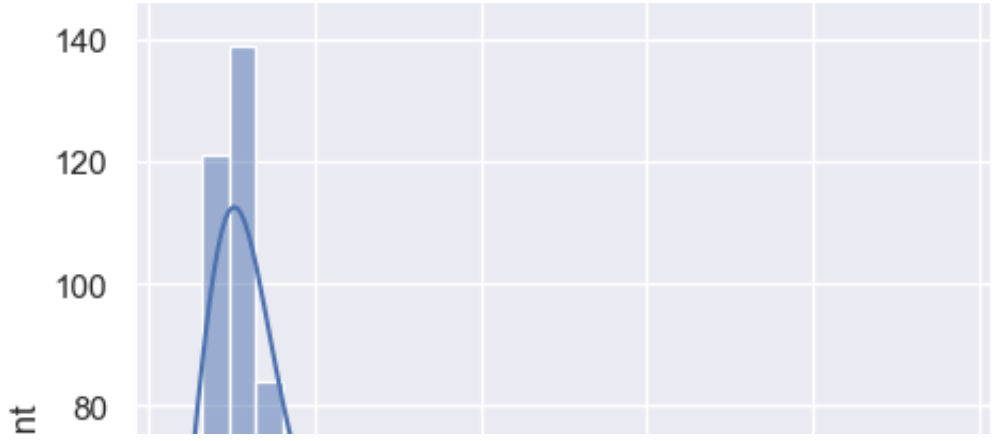


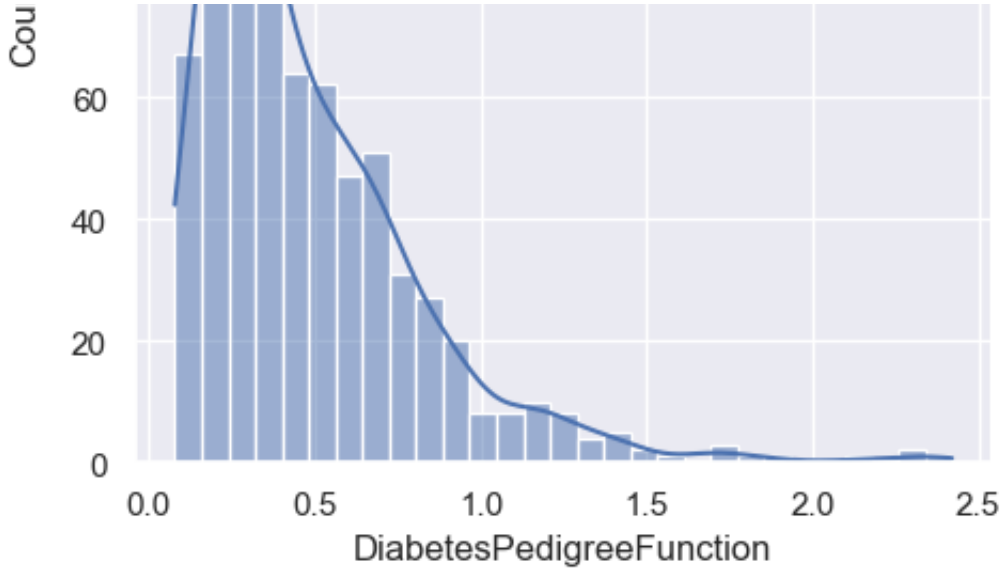



```
<Figure size 600x400 with 0 Axes>
```

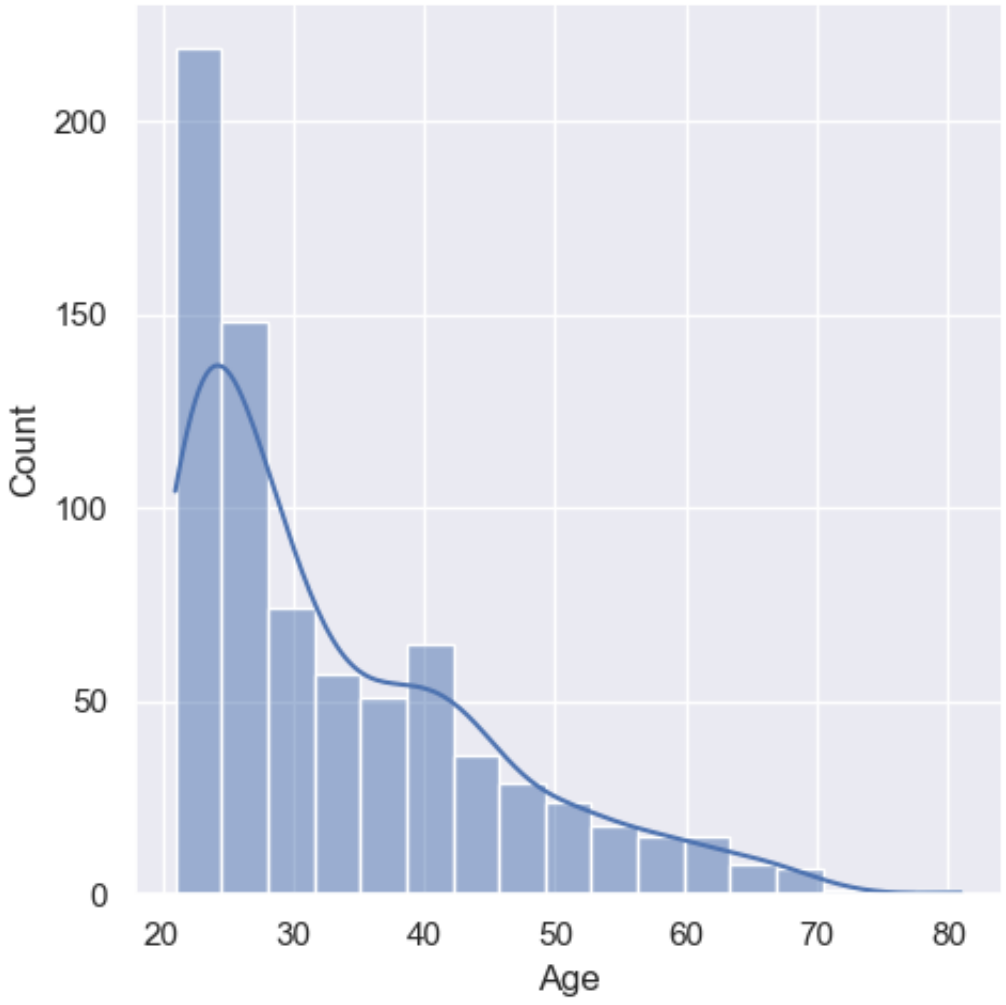



```
<Figure size 600x400 with 0 Axes>
```

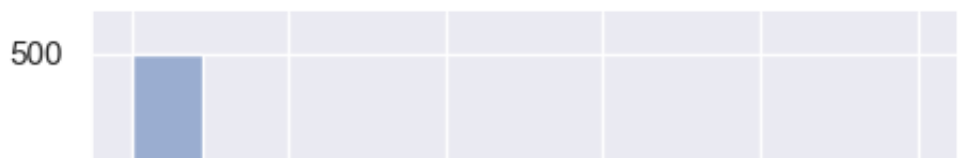


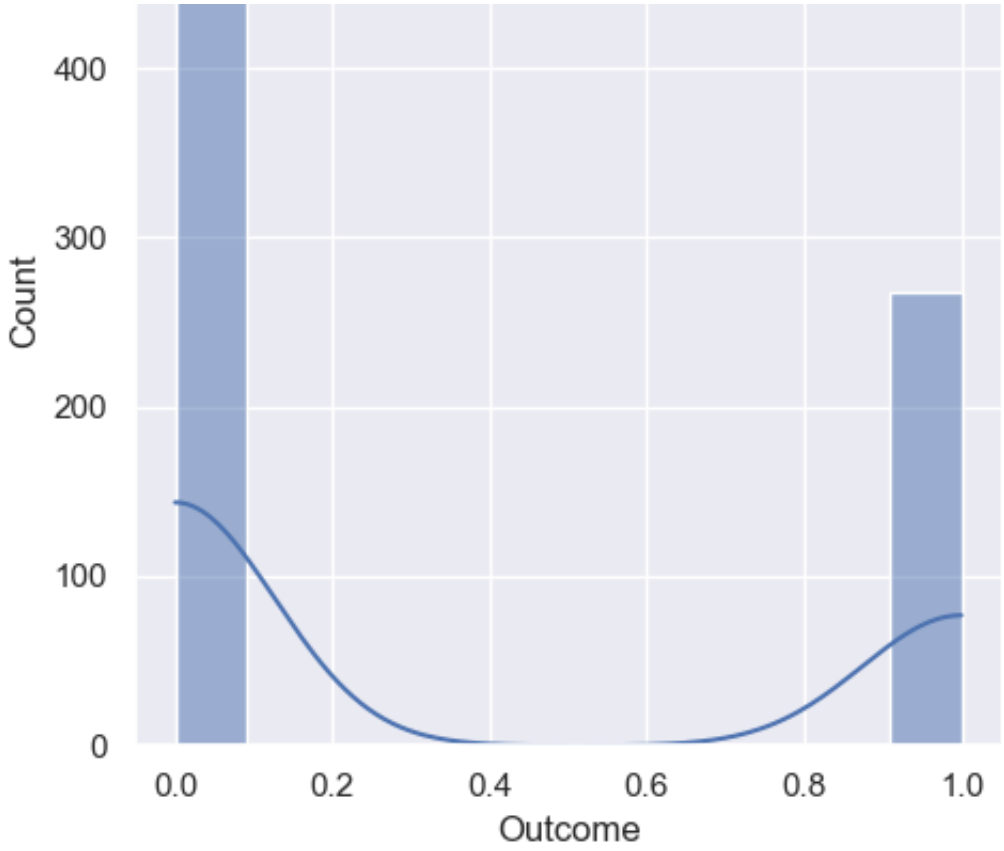



```
<Figure size 600x400 with 0 Axes>
```



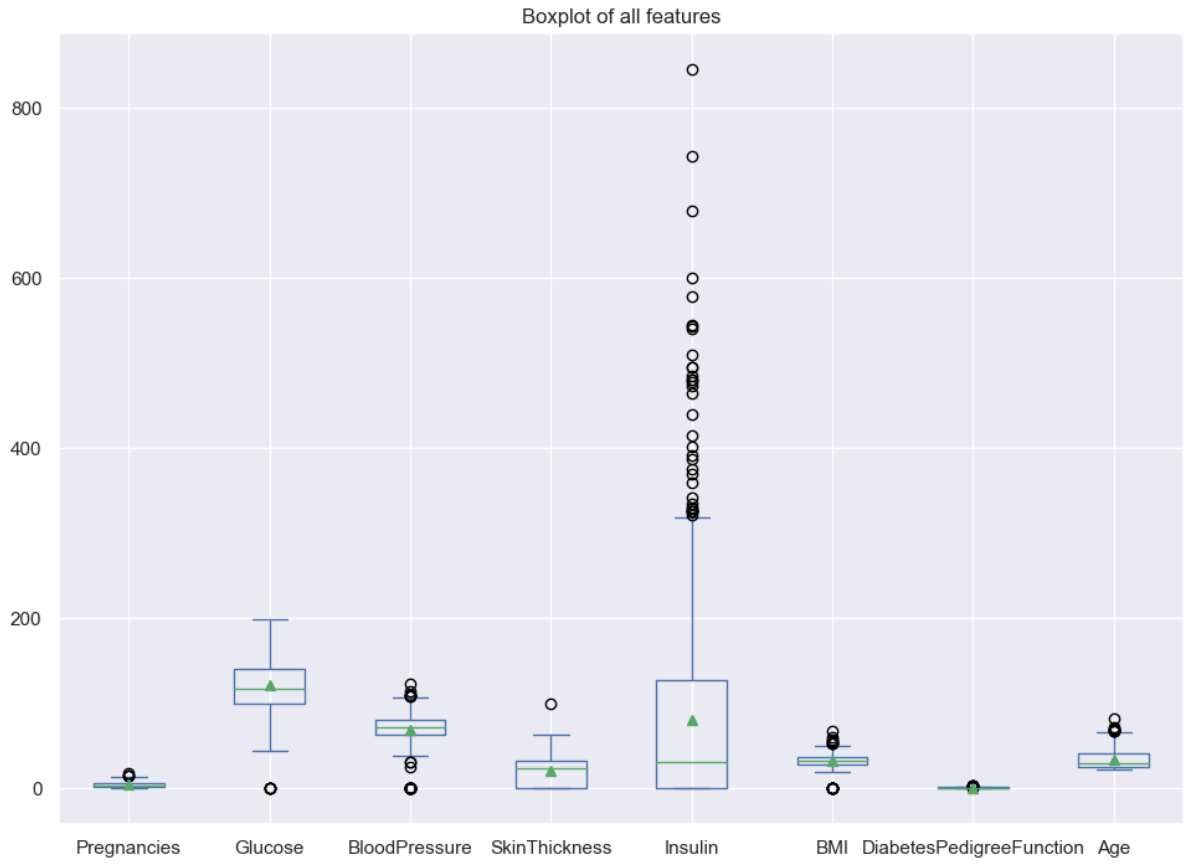

```
<Figure size 600x400 with 0 Axes>
```





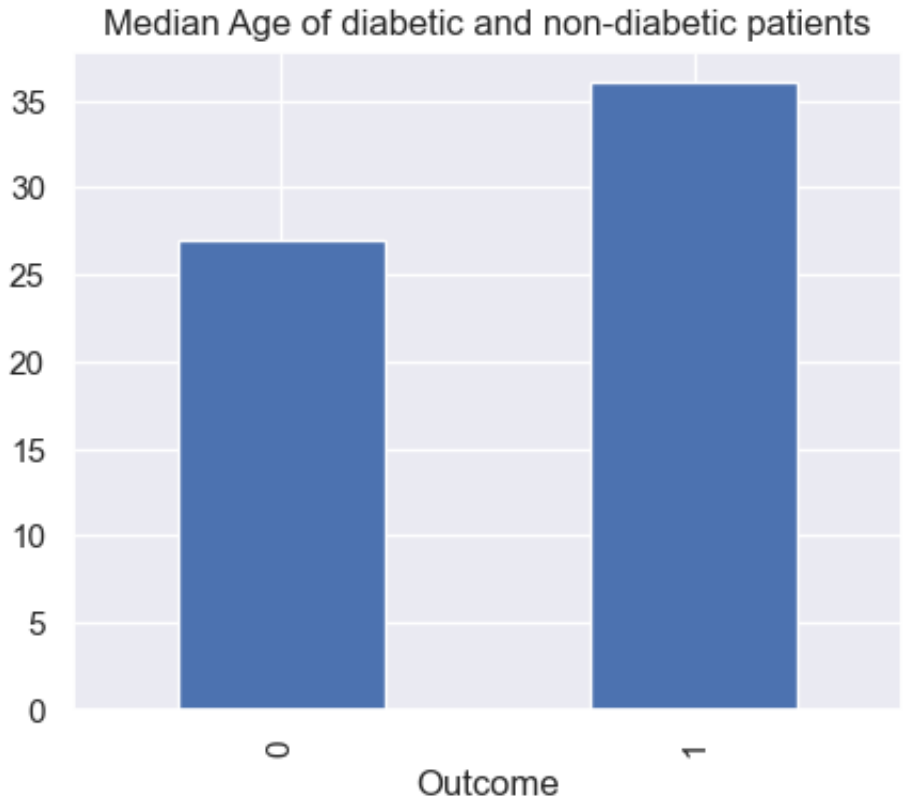
The presence of zeroes has skewed the distributions highly. Only the BMI seems to be normally distributed. Also, it is an imbalanced dataset as non-diabetic patients are almost twice in number.

In [100]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
ax = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickn
plt.show()
```

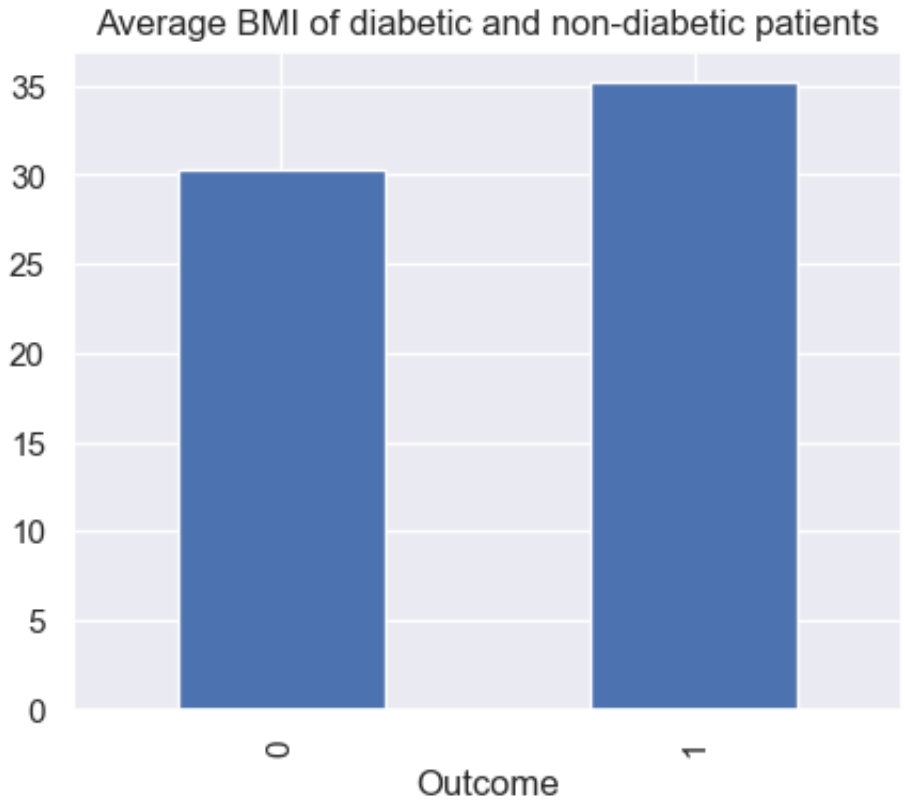


There are many outliers in the feature.

In [101]:
```
plt.figure(figsize=(5,4))
df.groupby('Outcome')['Age'].median().plot(kind='bar')
plt.title('Median Age of diabetic and non-diabetic patients')
plt.show()
```



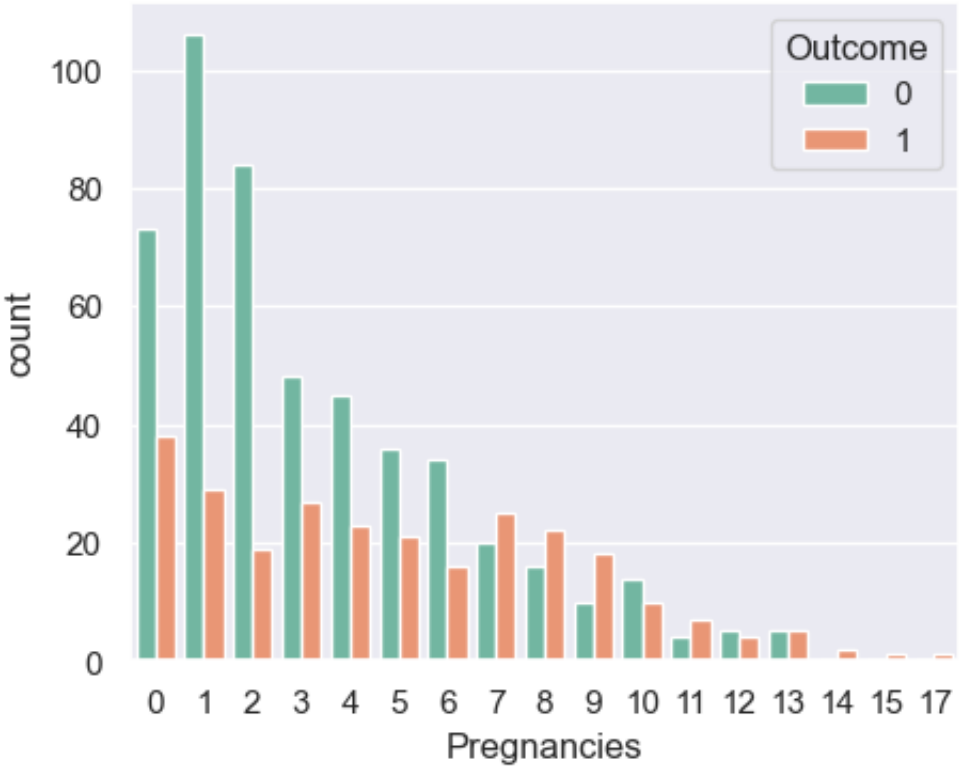
In [102]:
```
plt.figure(figsize=(5,4))
df.groupby('Outcome')['BMI'].mean().plot(kind='bar')
plt.title('Average BMI of diabetic and non-diabetic patients')
plt.show()
```



From the above plots, we can infer that:

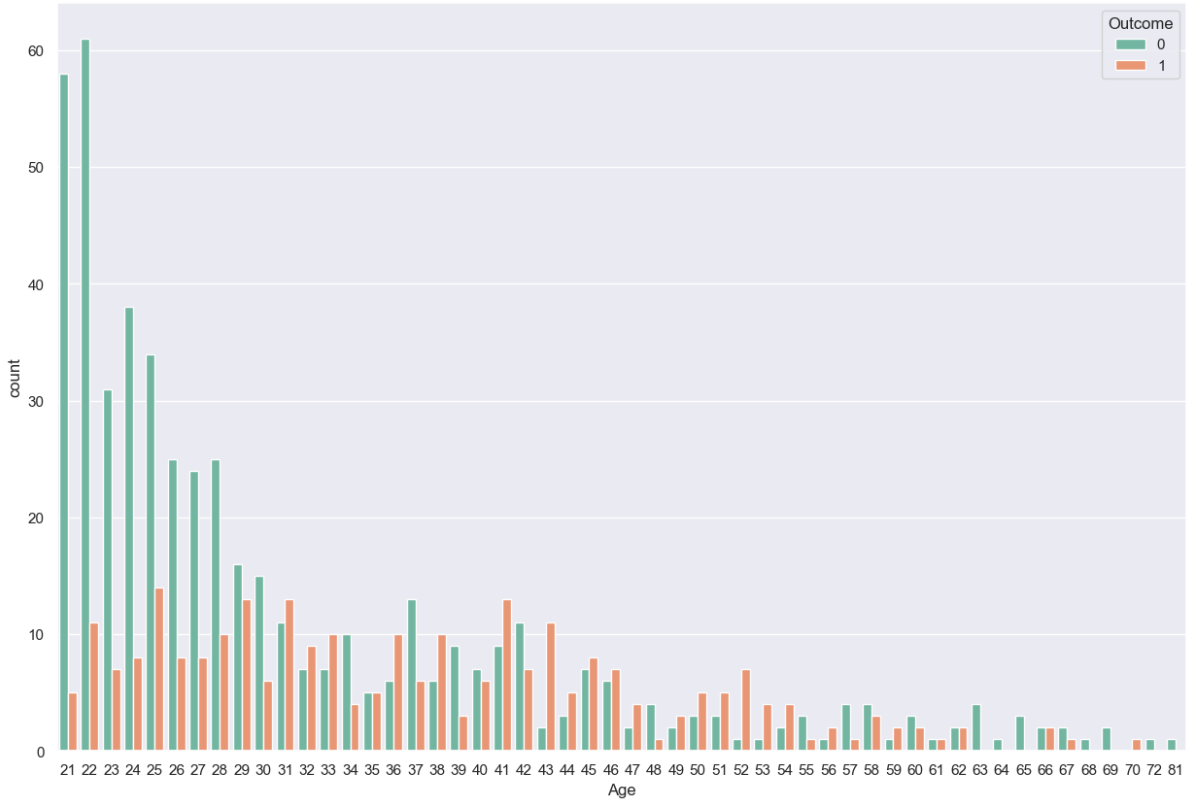
1. Diabetes has affected older people(more than 10 years older than non-diabetic).
2. BMI and insulin level of non-diabetic people are higher. This may attribute to patients undergoing insulin therapy.

In [103]:
```
plt.figure(figsize=(5,4))
sns.countplot(x='Pregnancies', hue='Outcome', data=df, palette=
plt.show()
```



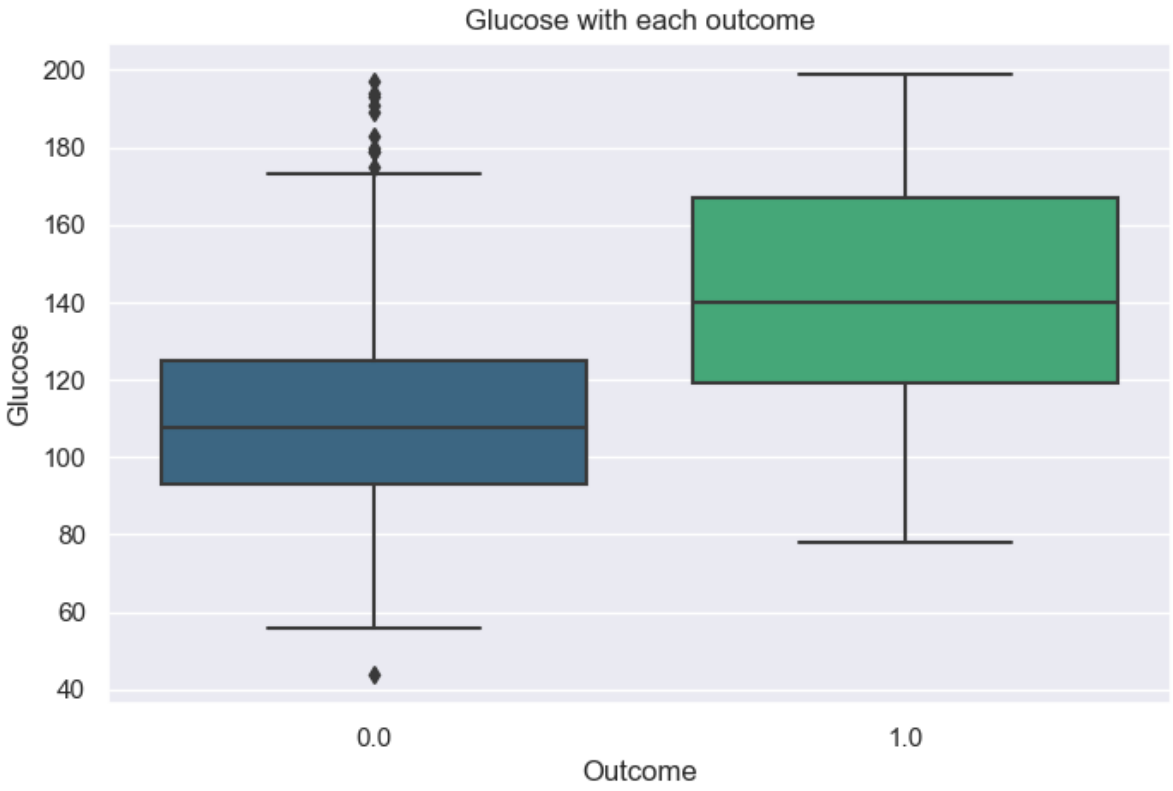
Pregnancies of more than 6 represent a higher chance of being affected with diabetes

In [104]:
```
plt.figure(figsize=(15,10))
sns.countplot(x='Age', hue='Outcome', data=df, palette='Set2')
plt.show()
```



Ages greater than 30 have higher chance of being affected by Diabetes

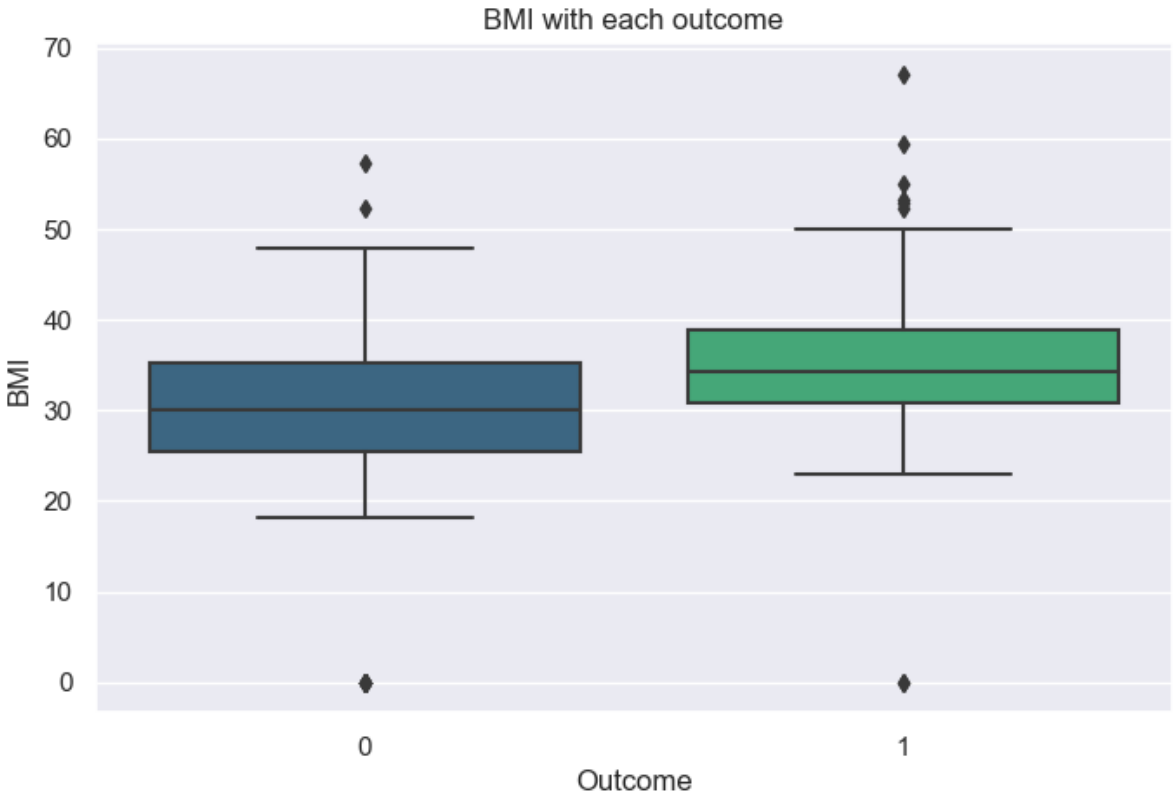
In [134]:
```
# checking outcome based on glucose
plt.figure(figsize=(8,5),dpi=100)
sns.boxplot(data=df,x='Outcome',y='Glucose',palette='viridis')
plt.title('Glucose with each outcome');
```



Individuals with diabetes have median glucose level around 140, whereas individuals without have around 110.

In [105]:
```
# check outcome based on BMI
plt.figure(figsize=(8,5),dpi=100)
sns.boxplot(data=df,x='Outcome',y='BMI',palette='viridis')
plt.title('BMI with each outcome')
```

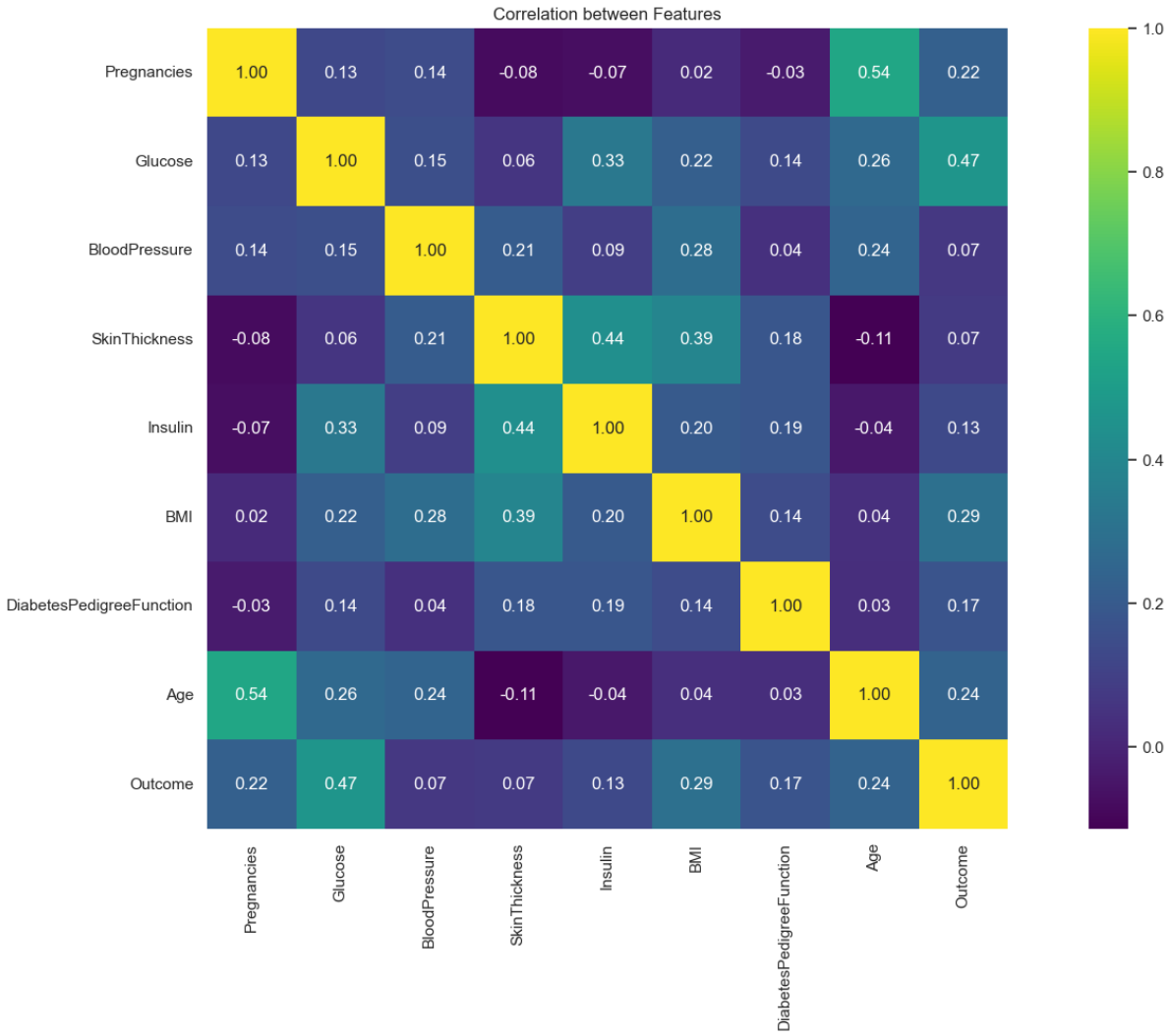
Out[105]: Text(0.5, 1.0, 'BMI with each outcome')



People with diabetes have higher median BMI than people without diabetes

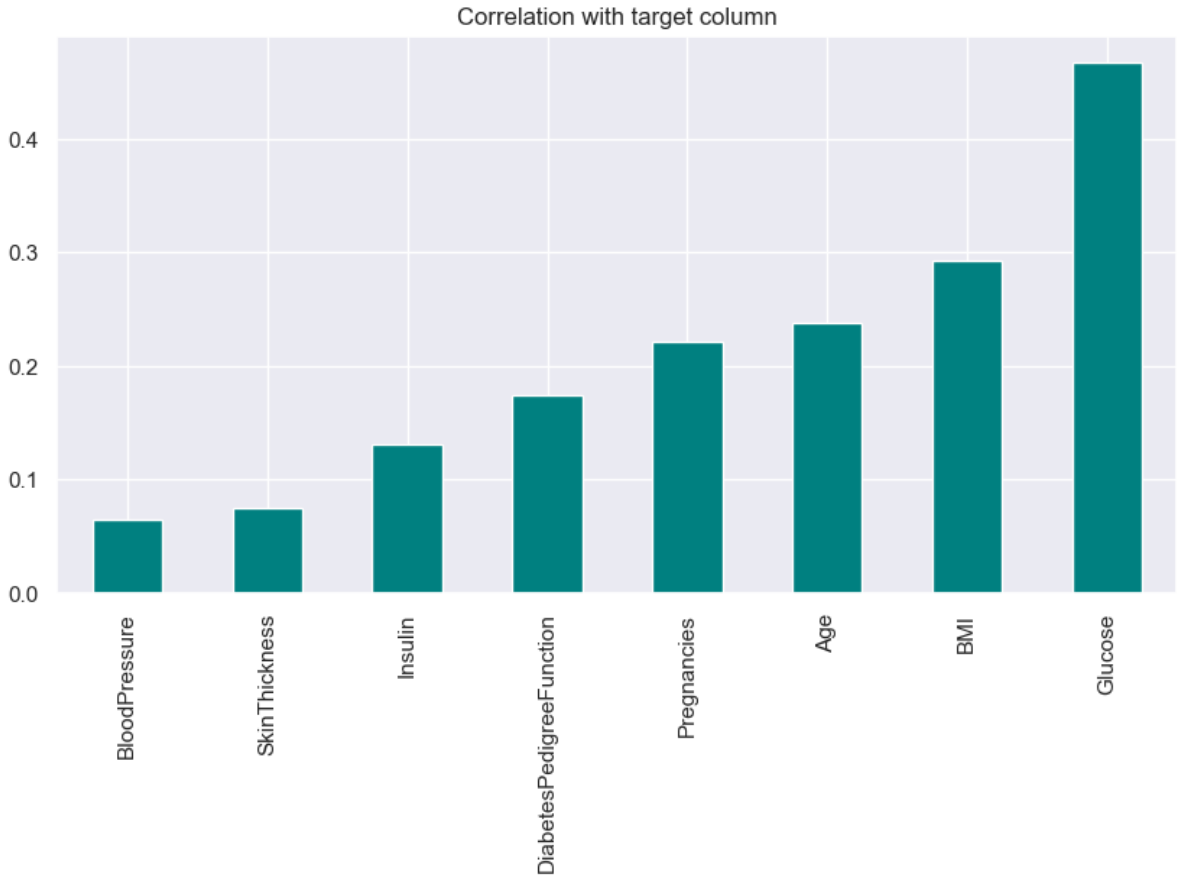
In [106]:
```
sns.pairplot(df,hue='Outcome',palette='viridis')
plt.show()
```

In [107]:
```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(), cmap='viridis', annot=True, fmt=".2f", s
plt.title('Correlation between Features')
plt.show()
```



In [108]:
```
# lets look at the of other features with target column

plt.figure(figsize=(10,5), dpi=100)
df.corr()['Outcome'].sort_values(ascending=True)[:-1].plot(kind
plt.title('Correlation with target column')
```

Out[108]: Text(0.5, 1.0, 'Correlation with target column')



The only feature that are correlated is the age and pregnancy and its okay cause the younger you are the less pregnant you should be.

Glucose, BMI, Age and Pregnancies have the niggest influence on the outcome (in this order), but, the blood pressure and skinthickness have the smallest influence

In [109]:
```
missing_values = ['Glucose','BloodPressure','SkinThickness','In
df[missing_values] = np.where(df[missing_values]==0, np.nan, df
```

In [110]:
```
from sklearn.impute import SimpleImputer
replace_ = SimpleImputer(missing_values=np.nan, strategy='mean'
cols = df.columns
df = pd.DataFrame(replace_.fit_transform(df))
df.columns = cols
```

In [111]:
```
1 df.describe()
```

Out[111]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Dia
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	121.686763	72.405184	29.153420	155.548223	32.457464	
std	3.369578	30.435949	12.096346	8.790942	85.021108	6.875151	
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	
25%	1.000000	99.750000	64.000000	25.000000	121.500000	27.500000	
50%	3.000000	117.000000	72.202592	29.153420	155.548223	32.400000	
75%	6.000000	140.250000	80.000000	32.000000	155.548223	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

IQR Technique

Quartiles: Imagine you have list of numbers, and you want to divide in four equal parts. The 3 points that divide the list into these four parts are called **Quartiles**. The first quartile (Q1) represents the lower 25% of the data, the second quartile(Q2) is the median, and the third quartile(Q3) represents the upper 25% of the data.

Interquartile Range(IQR): The IQR is the range of values between the first quartile(Q1), and the third quartile(Q3). It tells us how spread out the middle 50% of the data is. The larger the IQR, the more spread out the data in the middle.

Outer Detection: Outliers are data points that are very different from the majority of the data. In this approach, we use the IQR to identify potiential outliers. Any data point that falls below Q1 -1.5*IQR or above Q3 + 1.5* is considered an outlier.

Low Limit and Upper Limit: The low limit is calculated as Q1-1.5*IQR, and the upper limit is calculated as Q3+1.5*IQR. Data points below the low limit or above the upper limit are flagged as potential outliers.

Checking for Outliers: For each column in the dataset, we calcualte the IQR, the low limit, and the upper limit. Then, we check if any data points in that column fall outside these limits. If we find any such data points, it means there are outliers in that column.

In [112]:
```
## Outlier Analysis using IQR

def outlier_thresholds(dataframe, col_name, q1=0.20, q3=0.80):
    quartile1 = dataframe[col_name].quantile(q1)
    quartile3 = dataframe[col_name].quantile(q3)
    interquantile_range = quartile3 - quartile1
    up_limit = quartile3 + 1.5 * interquantile_range
    low_limit = quartile1 - 1.5 * interquantile_range
    return low_limit, up_limit

def check_outlier(dataframe, col_name):
    low_limit, up_limit = outlier_thresholds(dataframe, col_nam
    if dataframe[(dataframe[col_name] > up_limit) | (dataframe[
        return True
    else:
        return False

for col in df.columns:
    print(col, check_outlier(df, col))
```

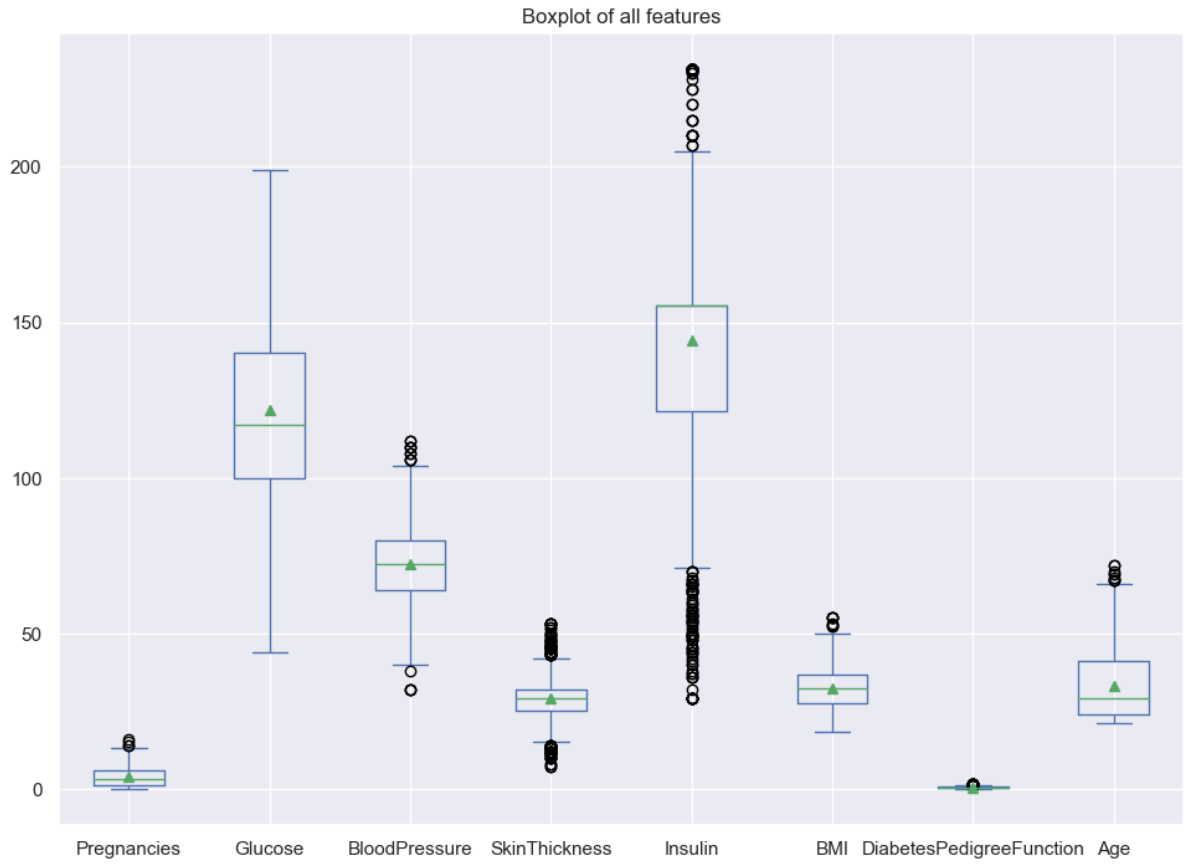
```
Pregnancies True
Glucose False
BloodPressure True
SkinThickness True
Insulin True
BMI True
DiabetesPedigreeFunction True
Age True
Outcome False
```

In [113]:
```
## Replacing Outliers according to the IQR method.

def replace_with_thresholds(dataframe, variable, q1=0.20, q3=0.
    low_limit, up_limit = outlier_thresholds(dataframe, variabl
    dataframe.loc[(dataframe[variable] < low_limit), variable] 
    dataframe.loc[(dataframe[variable] > up_limit), variable] =

for col in df.columns:
    replace_with_thresholds(df, col)

```

In [114]:
```
for col in df.columns:
    print(col, check_outlier(df, col))
```

```
Pregnancies False
Glucose False
BloodPressure False
SkinThickness False
Insulin False
BMI False
DiabetesPedigreeFunction False
Age False
Outcome False
```

In [115]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
ax = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickn
plt.show()
```



In [116]:
```
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [117]:
```
x = df.drop('Outcome', axis=1)
y=df['Outcome']
```

In [118]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_si
```

In [119]:
```
# Verify splitting of the data
print("{}% data is in the training set".format ((len(x_train)/l
print("{}% data is in the testing set".format ((len(x_test)/len
```

```
69.921875% data is in the training set
30.078125% data is in the testing set
```

In [120]:
```
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
/Users/abuadam/anaconda3/lib/python3.10/site-packages/sklearn/line
ar_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to con
verge (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 (ht
tps://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver opti
ons:
    https://scikit-learn.org/stable/modules/linear_model.html#logi
stic-regression (https://scikit-learn.org/stable/modules/linear_mo
del.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[120]:
```
▾ LogisticRegression

LogisticRegression()
```

In [121]:
```
y_predict = model.predict(x_test)
```

In [122]:
```
model_score = model.score(x_test,y_test)
model_score
```

Out[122]:
```
0.7878787878787878
```

In [123]:
```
model_score1 = model.score(x_train,y_train)
model_score1
```

Out[123]:
```
0.7653631284916201
```

Testing the prediction

In [124]:
```
x_test
```

Out[124]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
285	7.0	136.0	74.0	26.00000	135.000000	26.0	
101	1.0	151.0	60.0	29.15342	155.548223	26.1	
581	6.0	109.0	60.0	27.00000	155.548223	25.0	
352	3.0	61.0	82.0	28.00000	155.548223	34.4	
726	1.0	116.0	78.0	29.00000	180.000000	36.1	
...	...	...	...	...	...	...	
241	4.0	91.0	70.0	32.00000	88.000000	33.1	
599	1.0	109.0	38.0	18.00000	120.000000	23.1	
650	1.0	91.0	54.0	25.00000	100.000000	25.2	
11	10.0	168.0	74.0	29.15342	155.548223	38.0	
214	9.0	112.0	82.0	32.00000	175.000000	34.2	

231 rows × 8 columns

In [125]:
```
prediction=model.predict([[2,144,55,36,136,35,0.2,26]]) # Testi
# Pregnancy,Glucose,BloodPressure,SkinThickness,Insulin,BMI,Dia
print(prediction)

if prediction[0]==1:
    print("diabetic")
else:
    print("Non diabetic")
```

```
[0.]
Non diabetic

/Users/abuadam/anaconda3/lib/python3.10/site-packages/sklearn/bas
e.py:439: UserWarning: X does not have valid feature names, but Lo
gisticRegression was fitted with feature names
  warnings.warn(
```

In [126]:
```
prediction=model.predict([[2,144,55,36,136,35,0.2,46]]) # Testi
# Pregnancy,Glucose,BloodPressure,SkinThickness,Insulin,BMI,Dia
print(prediction)

if prediction[0]==1:
    print("diabetic")
else:
    print("Non diabetic")
```

```
[1.]
diabetic

/Users/abuadam/anaconda3/lib/python3.10/site-packages/sklearn/bas
e.py:439: UserWarning: X does not have valid feature names, but Lo
gisticRegression was fitted with feature names
  warnings.warn(
```

Exporting the model using Joblib and Pickle

In [127]:
```
import joblib
```

In [128]:
```
joblib.dump(model,"diabetescheck.pkl")
```

Out[128]:
```
['diabetescheck.pkl']
```

In [130]:
```
import pickle
```

In [133]:
```
pickle.dump(model,open("diabeticchecking.pkl",'wb'))
```

In [ ]:
```

```