In [2]:

```
#Installation of required libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score, mean_squared_error, r2_score, roc_auc_score, roc_curve, classificati
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold
import warnings
warnings.simplefilter(action = "ignore")
```

In [3]:

```
#Reading the dataset
df = pd.read_csv("C:\Program Files\Python36\Data\diabetes.csv")
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1	103	30	38	83	43.3	0.183	33	0
1	1	115	70	30	96	34.6	0.529	32	1
2	3	126	88	41	235	39.3	0.704	27	0
3	1	122	64	32	156	35.1	0.692	30	1
4	10	179	70	0	0	35.1	0.200	37	0

In [4]:

```
#Feature information
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
```

In [5]:
```
# Descriptive statistics of the data set accessed.
df.describe([0.10,0.25,0.50,0.75,0.90,0.95,0.99]).T
```

Out[5]:

	count	mean	std	min	10%	25%	50%	75%	90%	95%	99%	max
Pregnancies	768.0	3.845052	3.369578	0.000	0.000	1.00000	3.0000	6.00000	9.0000	10.00000	13.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	85.000	99.00000	117.0000	140.25000	167.0000	181.00000	196.00000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	54.000	62.00000	72.0000	80.00000	88.0000	90.00000	106.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.000	0.00000	23.0000	32.00000	40.0000	44.00000	51.33000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.000	0.00000	30.5000	127.25000	210.0000	293.00000	519.90000	846.00
BMI	768.0	31.992578	7.884160	0.000	23.600	27.30000	32.0000	36.60000	41.5000	44.39500	50.75900	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.165	0.24375	0.3725	0.62625	0.8786	1.13285	1.69833	2.42
Age	768.0	33.240885	11.760232	21.000	22.000	24.00000	29.0000	41.00000	51.0000	58.00000	67.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.000	0.00000	0.0000	1.00000	1.0000	1.00000	1.00000	1.00

In [6]:
```
# The distribution of the Outcome variable was examined.
df["Outcome"].value_counts()*100/len(df)
```

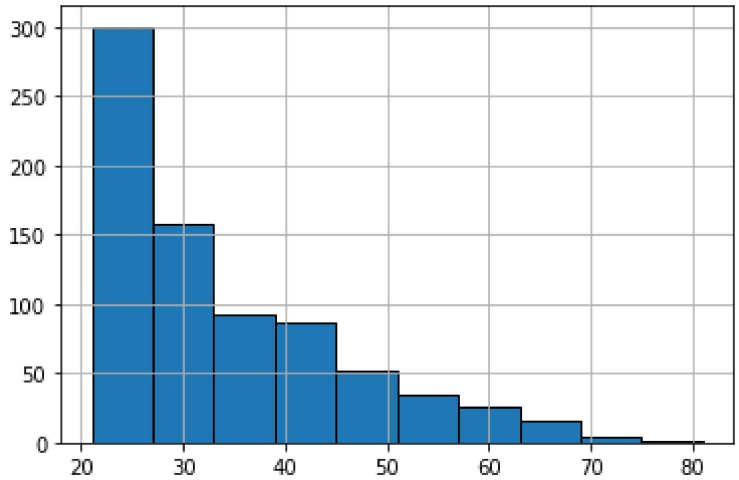
Out[6]:
```
0    65.104167
1    34.895833
```

```
Name: Outcome, dtype: float64
```

In [7]:
```
# The classes of the outcome variable were examined.
df.Outcome.value_counts()
```

Out[7]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [8]:
```
# The histagram of the Age variable was reached.
df["Age"].hist(edgecolor = "black");
```



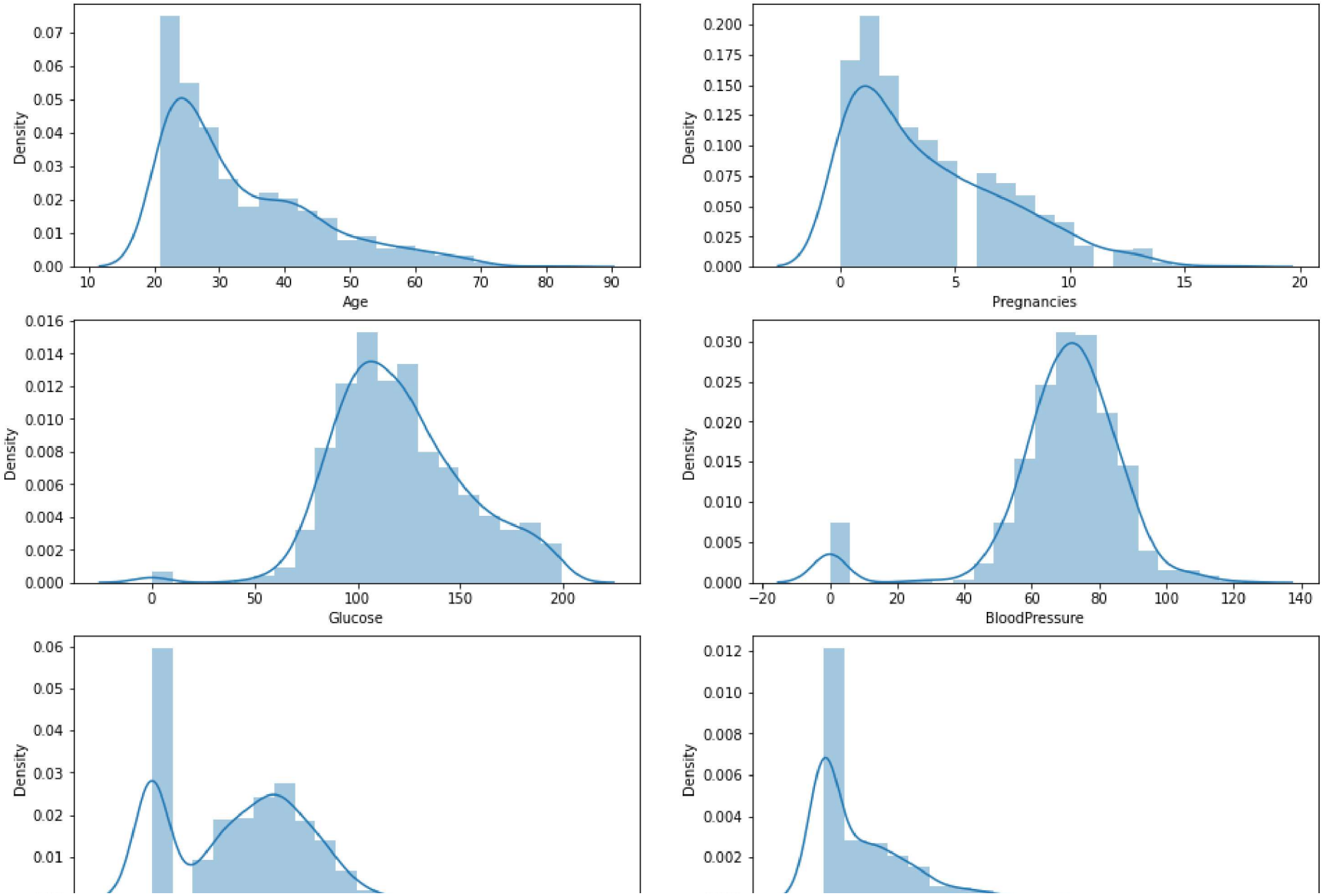
In [9]:
```
print("Max Age: " + str(df["Age"].max()) + " Min Age: " + str(df["Age"].min()))
```

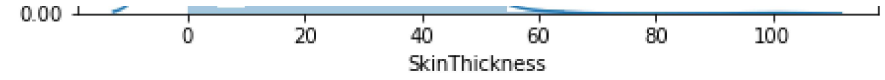
```
Max Age: 81 Min Age: 21
```

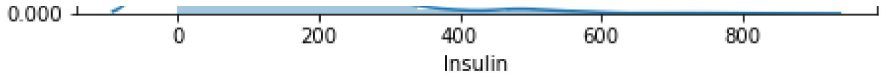
In [10]:
```
# Histogram and density graphs of all variables were accessed.
fig, ax = plt.subplots(4,2, figsize=(16,16))
sns.distplot(df.Age, bins = 20, ax=ax[0,0])
sns.distplot(df.Pregnancies, bins = 20, ax=ax[0,1])
sns.distplot(df.Glucose, bins = 20, ax=ax[1,0])
sns.distplot(df.BloodPressure, bins = 20, ax=ax[1,1])
sns.distplot(df.SkinThickness, bins = 20, ax=ax[2,0])
sns.distplot(df.Insulin, bins = 20, ax=ax[2,1])
```

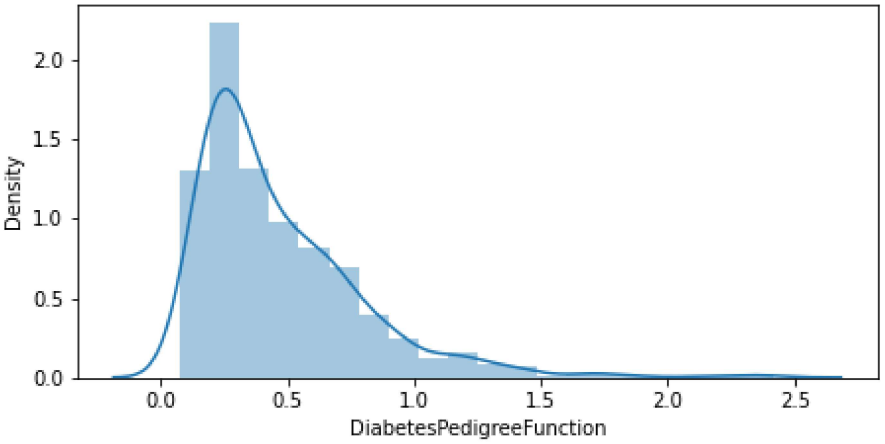
```
sns.distplot(df.DiabetesPedigreeFunction, bins = 20, ax=ax[3,0])
sns.distplot(df.BMI, bins = 20, ax=ax[3,1])
```

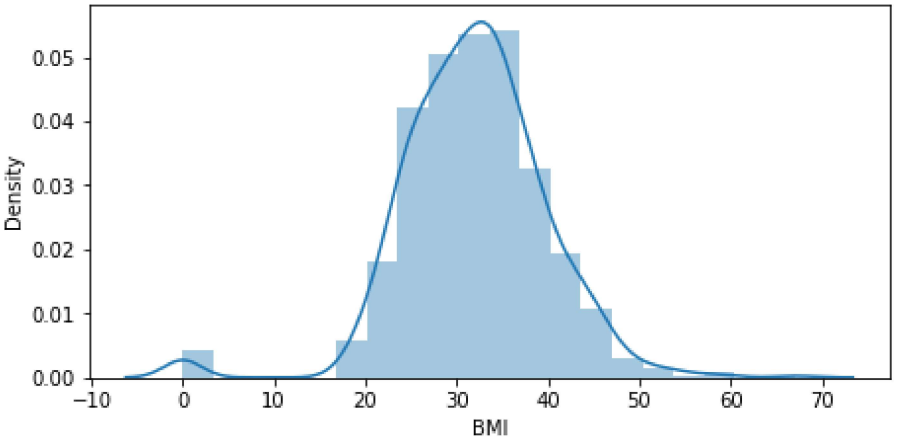
Out[10]: <AxesSubplot:xlabel='BMI', ylabel='Density'>











In [11]:
```
df.groupby("Outcome").agg({"Pregnancies":"mean"})
```

Out[11]:

Outcome	Pregnancies
0	3.298000
1	4.865672

In [12]:
```
df.groupby("Outcome").agg({"Age":"mean"})
```

Out[12]:

Outcome	Age
0	31.190000
1	37.067164

In [13]:
```
df.groupby("Outcome").agg({"Age":"max"})
```

Out[13]:

Outcome	Age
0	81
1	70

In [14]:
```
df.groupby("Outcome").agg({"Insulin": "mean"})
```

Out[14]:

Outcome	Insulin
0	68.792000
1	100.335821

In [15]:
```
df.groupby("Outcome").agg({"Insulin": "max"})
```

Out[15]:

Outcome	Insulin
0	744
1	846

In [16]:
```
df.groupby("Outcome").agg({"Glucose": "mean"})
```

Out[16]:

Outcome	Glucose
0	109.980000
1	141.257463

In [17]:

```
df.groupby("Outcome").agg({"Glucose": "max"})
```

Out[17]:

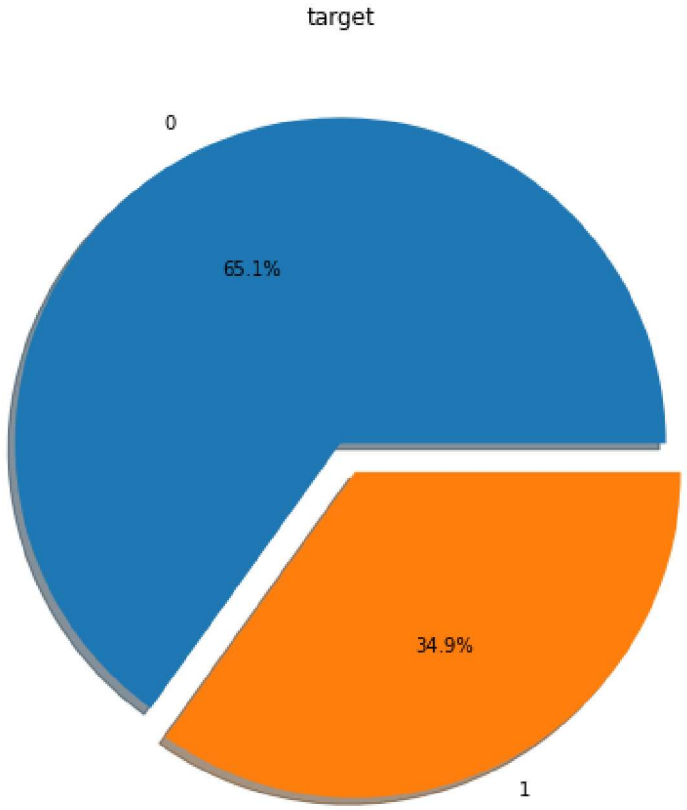
Outcome	Glucose
0	197
1	199

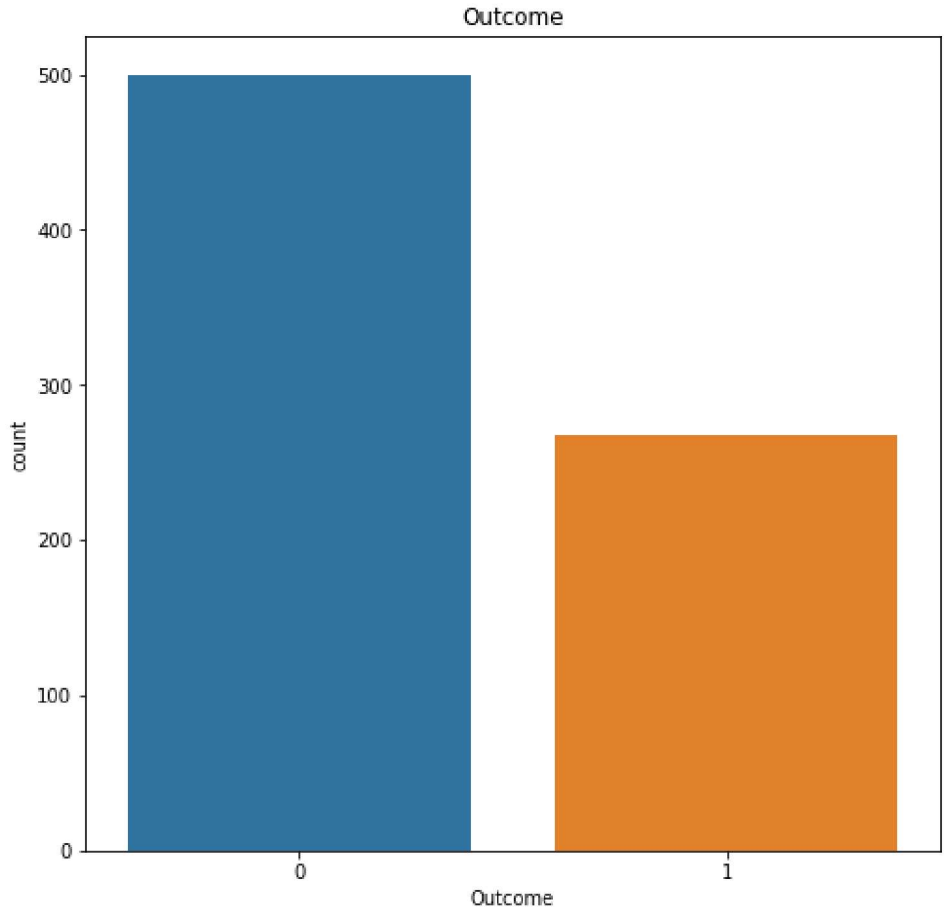
In [18]:
```
df.groupby("Outcome").agg({"BMI": "mean"})
```

Out[18]:

Outcome	BMI
0	30.304200
1	35.142537

In [19]:
```
# The distribution of the outcome variable in the data was examined and visualized.
f,ax=plt.subplots(1,2,figsize=(18,8))
df['Outcome'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],shadow=True)
ax[0].set_title('target')
ax[0].set_ylabel('')
sns.countplot('Outcome',data=df,ax=ax[1])
ax[1].set_title('Outcome')
plt.show()
```





In [20]:

```
# Access to the correlation of the data set was provided. What kind of relationship is examined between the variables.
# If the correlation value is> 0, there is a positive correlation. While the value of one variable increases, the value of the oth
# Correlation = 0 means no correlation.
# If the correlation is <0, there is a negative correlation. While one variable increases, the other variable decreases.
# When the correlations are examined, there are 2 variables that act as a positive correlation to the Salary dependent variable.
# These variables are Glucose. As these increase, Outcome variable increases.
df.corr()
```

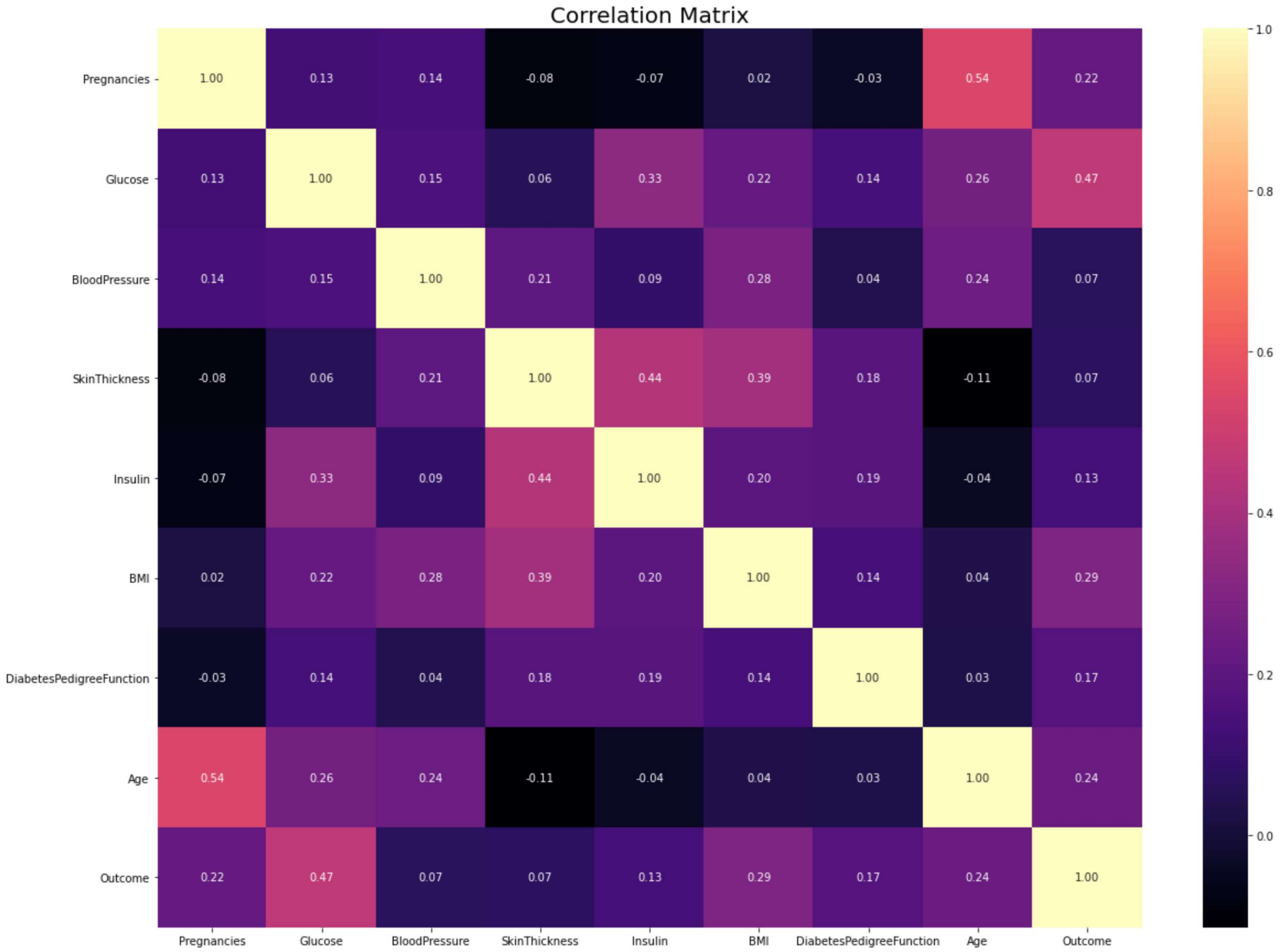
Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

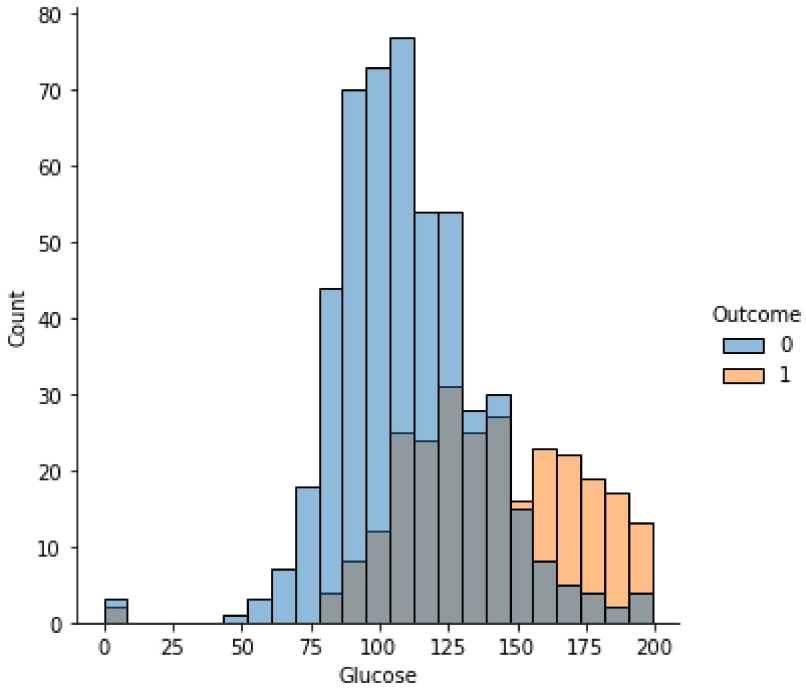
In [21]:

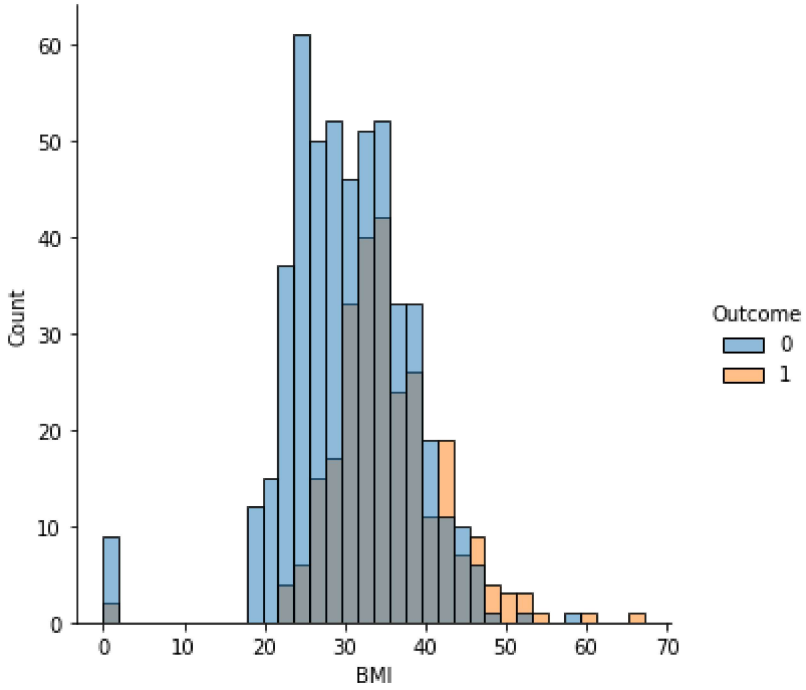
```
# Correlation matrix graph of the data set
f, ax = plt.subplots(figsize= [20,15])
sns.heatmap(df.corr(), annot=True, fmt=".2f", ax=ax, cmap = "magma" )
ax.set_title("Correlation Matrix", fontsize=20)
plt.show()
```


Correlation Matrix
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
1.00 0.13 0.14 -0.08 -0.07 0.02 -0.03 0.54 0.22
0.13 1.00 0.15 0.06 0.33 0.22 0.14 0.26 0.47
0.14 0.15 1.00 0.21 0.09 0.28 0.04 0.24 0.07
-0.08 0.06 0.21 1.00 0.44 0.39 0.18 -0.11 0.07
-0.07 0.33 0.09 0.44 1.00 0.20 0.19 -0.04 0.13
0.02 0.22 0.28 0.39 0.20 1.00 0.14 0.04 0.29
-0.03 0.14 0.04 0.18 0.19 0.14 1.00 0.03 0.17
0.54 0.26 0.24 -0.11 -0.04 0.04 0.03 1.00 0.24
0.22 0.47 0.07 0.07 0.13 0.29 0.17 0.24 1.00
1.0
0.8
0.6
0.4
0.2
0.0

In [22]:
```
sns.displot(df, x="Glucose", hue='Outcome')
sns.displot(df, x="BMI", hue='Outcome')
```

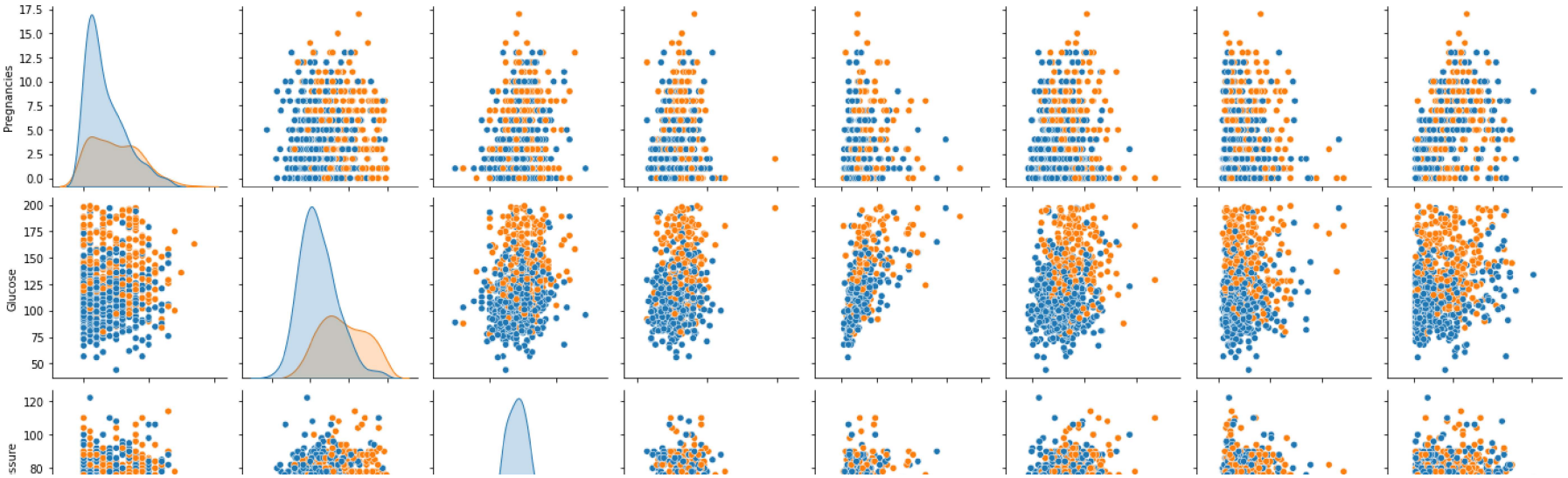
Out[22]: <seaborn.axisgrid.FacetGrid at 0x1d7d36b0250>

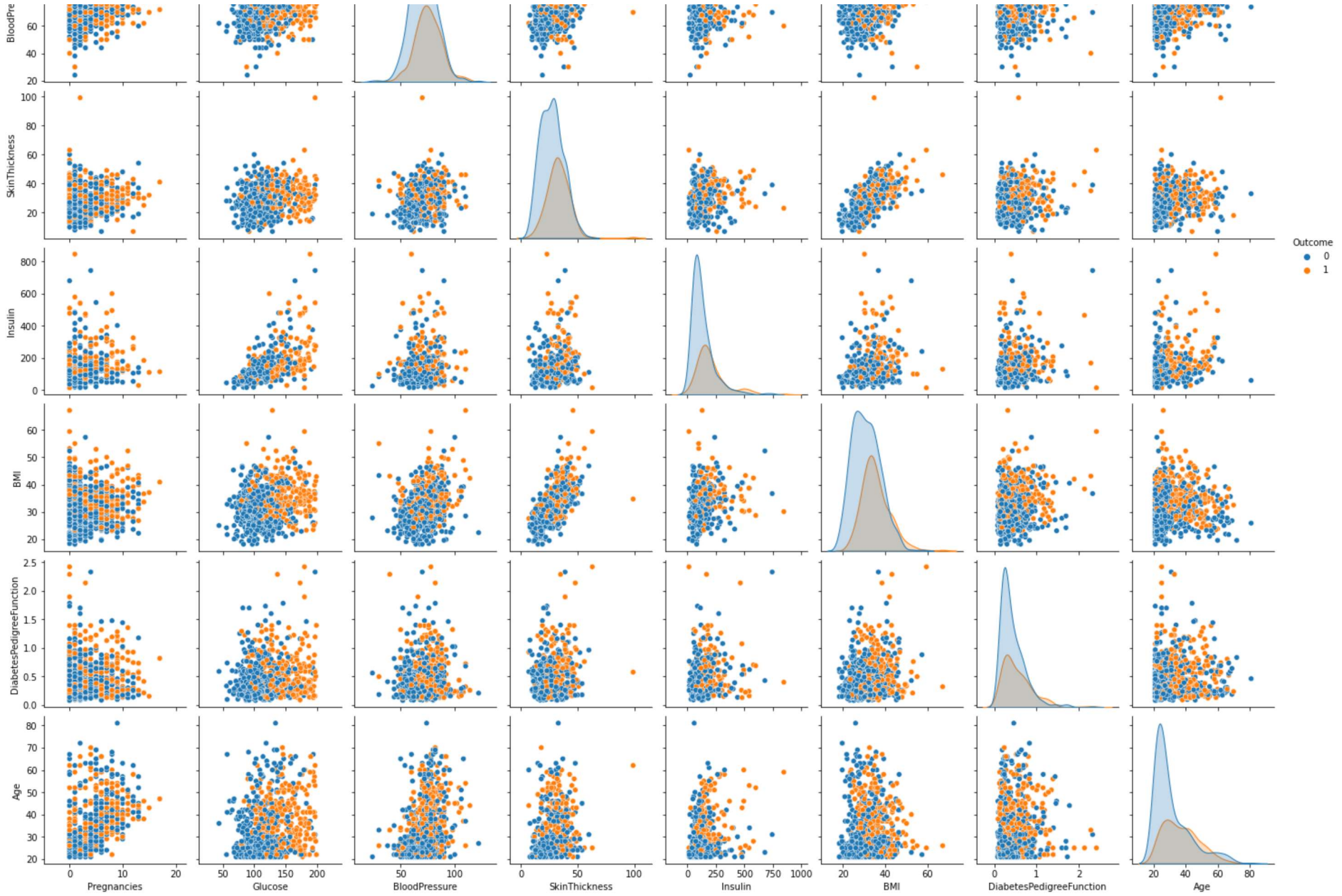




In [37]:
```
sns.pairplot(df,hue="Outcome")
```

Out[37]: <seaborn.axisgrid.PairGrid at 0x1d7ccc9a1c0>




BloodPre
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
0
1

In [ ]:

In [31]:

```
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].
```

In [32]:
```
df.head()
```

Out[32]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1	103.0	30.0	38.0	83.0	43.3	0.183	33	0
1	1	115.0	70.0	30.0	96.0	34.6	0.529	32	1
2	3	126.0	88.0	41.0	235.0	39.3	0.704	27	0
3	1	122.0	64.0	32.0	156.0	35.1	0.692	30	1
4	10	179.0	70.0	NaN	NaN	35.1	0.200	37	0

In [33]:
```
# Now, we can look at where are missing values
df.isnull().sum()
```

Out[33]:
```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [ ]:
```

```