diabetes-analysis

September 27, 2023

1 IMPORTING LIBRARIES

[5]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import scipy
from  matplotlib.colors import ListedColormap
import warnings
warnings.filterwarnings('ignore')
```

2 Loading Data Set for Performing EDA (exploraitory data analysis)

[9]:

```python
df= pd.read_csv("D:\Downloads\diabetes 01.csv")
print(df)
```

```
     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
..           …      …            …            …      …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     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
..                         …  …        …
763                      0.171   63         0
764                      0.340   27         0
765                      0.245   30         0
766                      0.349   47         1
767                      0.315   23         0

[768 rows x 9 columns]
```

3 EDA starts from here

[13]:

```
df.head(11)
## df.head () returns the toprows of the DataFrame
```

[13]:

```
    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
5             5      116             74              0        0  25.6
6             3       78             50             32       88  31.0
7            10      115              0              0        0  35.3
8             2      197             70             45      543  30.5
9             8      125             96              0        0   0.0
10            4      110             92              0        0  37.6

    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
5                      0.201   30        0
6                      0.248   26        1
7                      0.134   29        0
8                      0.158   53        1
9                      0.232   54        1
10                     0.191   30        0
```

[14]:

```
df.tail(11)
##  df.tail () returns the lastrows of the DataFrame
```

[14]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
757            0      123             72              0        0  36.3
758            1      106             76              0        0  37.5
```

```
759            6      190             92              0        0  35.5
760            2       88             58             26       16  28.4
761            9      170             74             31        0  44.0
762            9       89             62              0        0  22.5
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
757                     0.258   52        1
758                     0.197   26        0
759                     0.278   66        1
760                     0.766   22        0
761                     0.403   43        1
762                     0.142   33        0
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0
```

[16]:

```
df.shape
#    The first value represents the number of rows in the DataFrame.
 #   The second value represents the number of columns in the DataFrame.
```

[16]:

```
(768, 9)
```

[19]:

```
df.describe()
#The df.describe() method provides a quick overview of the central tendency and␣
 ↪dispersion of numerical data in your DataFrame, which is helpful for␣
 ↪understanding the distribution of your data.
```

[19]:

```
       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
```

[20]:
```
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
```

[23]:
```
df.columns
```

[23]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

4 Checking and Cleaning Null Values if any

[24]:
```
df.isnull()
df.isnull().sum()
```

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

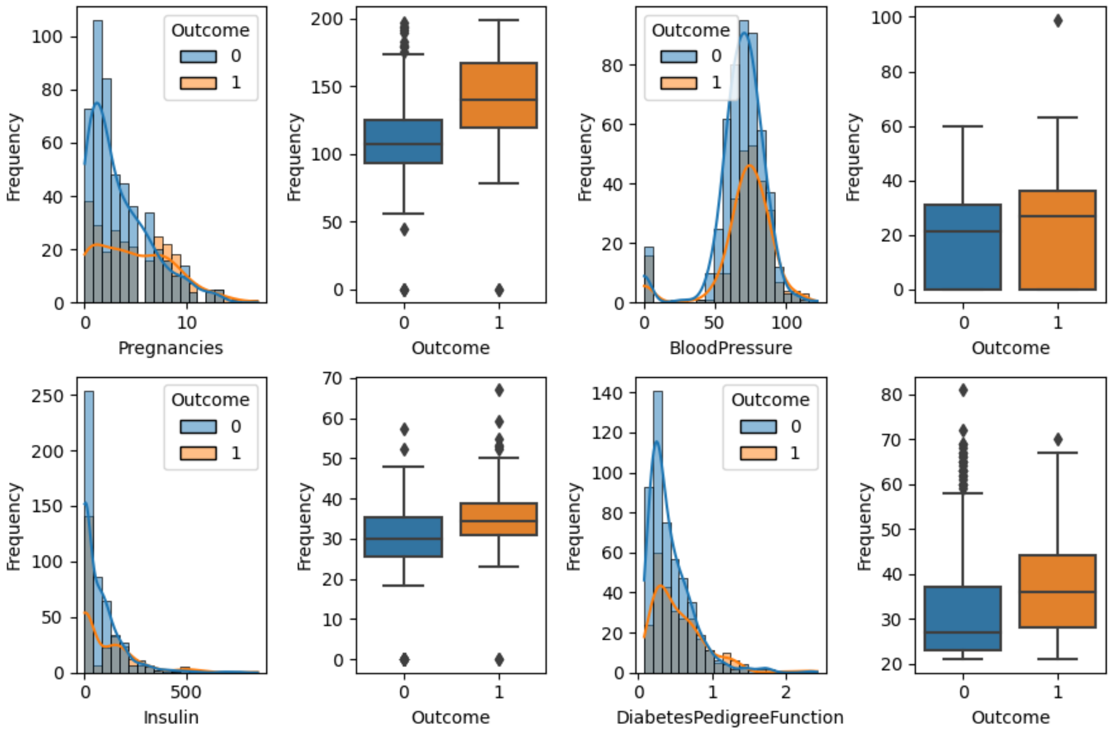
```
dtype: int64
```

5 Analysis starts by Plotting Graphs on different parameters.

6 Frequency Plot

[35]:

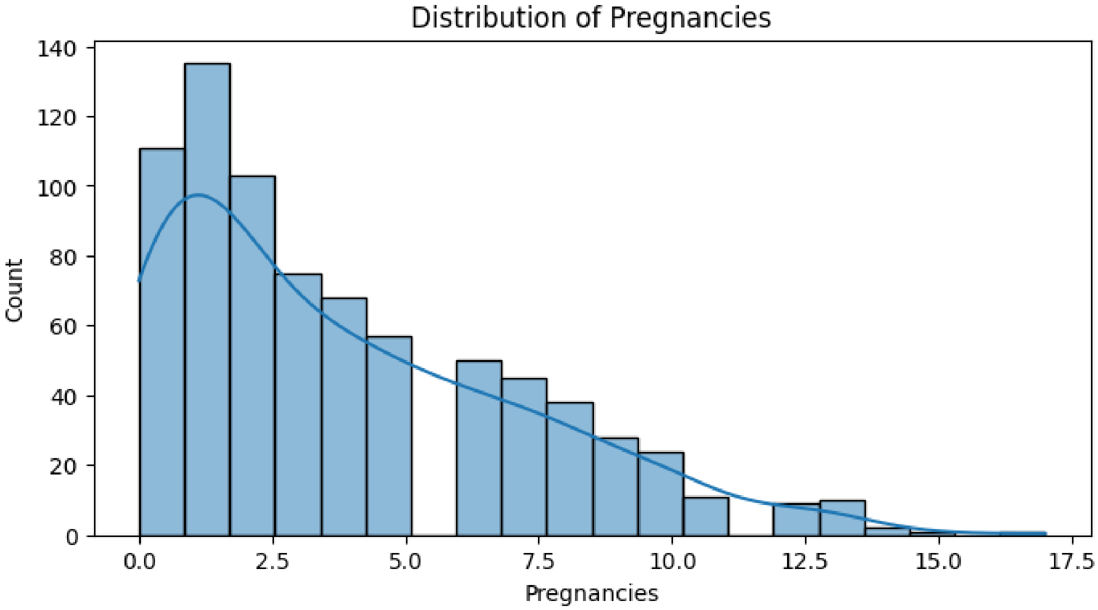
```
plt.figure(figsize=(9,6))
columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',␣
 ↪'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']
for i, col in enumerate(columns,1):
    plt.subplot(2, 4, i)


    if i% 2==0:
        sns.boxplot(x='Outcome',  y= col , data =df)
        plt.xlabel('Outcome')
    else:
        sns.histplot(data=df , x= col, hue ='Outcome', kde = True, bins = 20,␣
 ↪edgecolor ='k')
        plt.xlabel(col)

    plt.ylabel('Frequency')

plt.tight_layout()
```

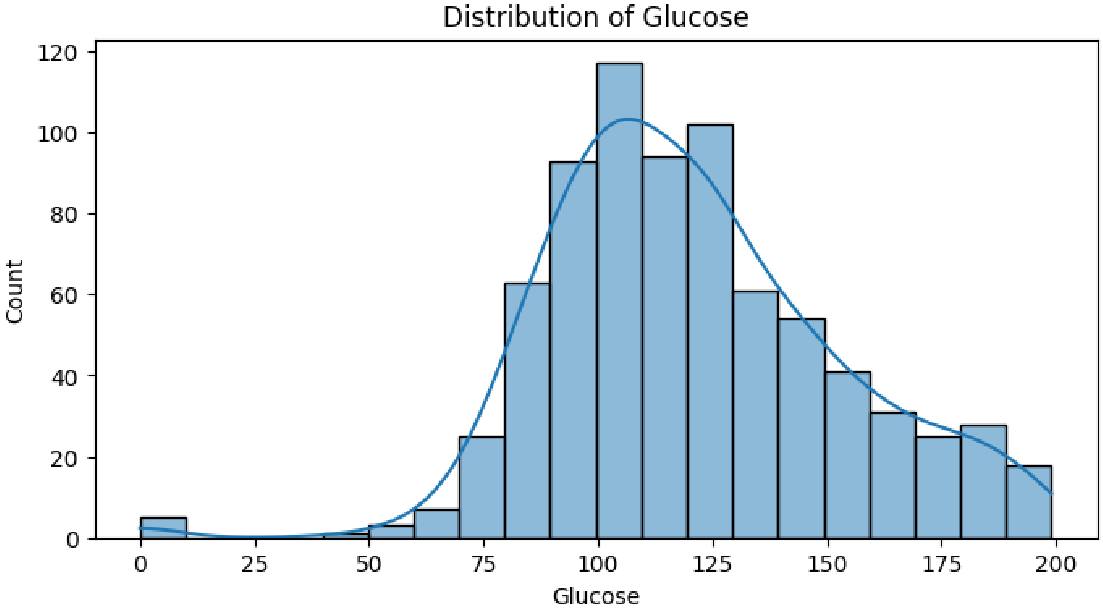


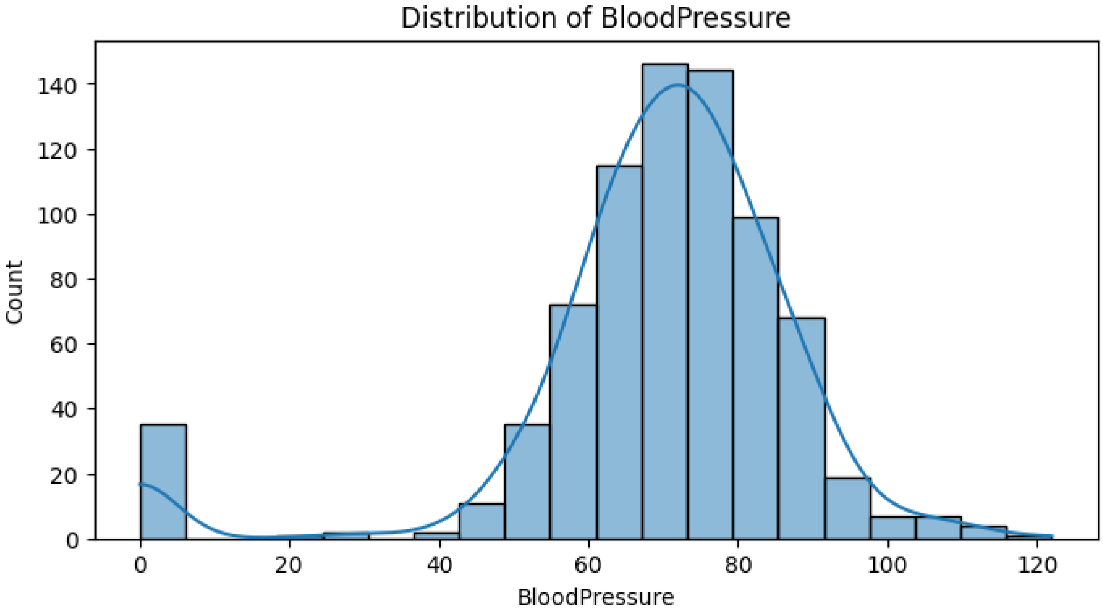
7 Distribution of Pregnancies

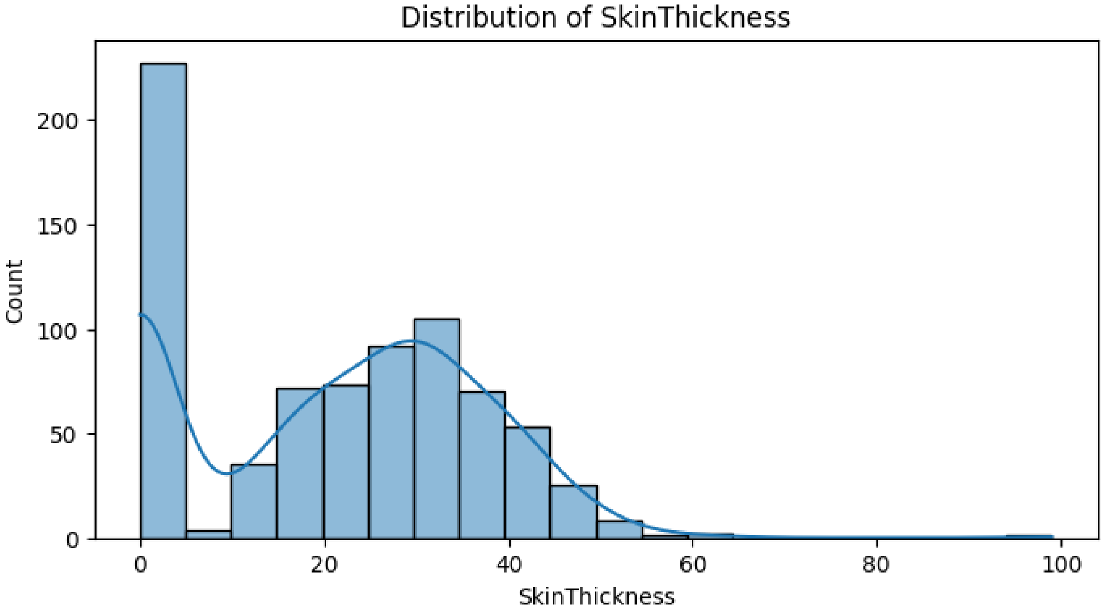
[36]:

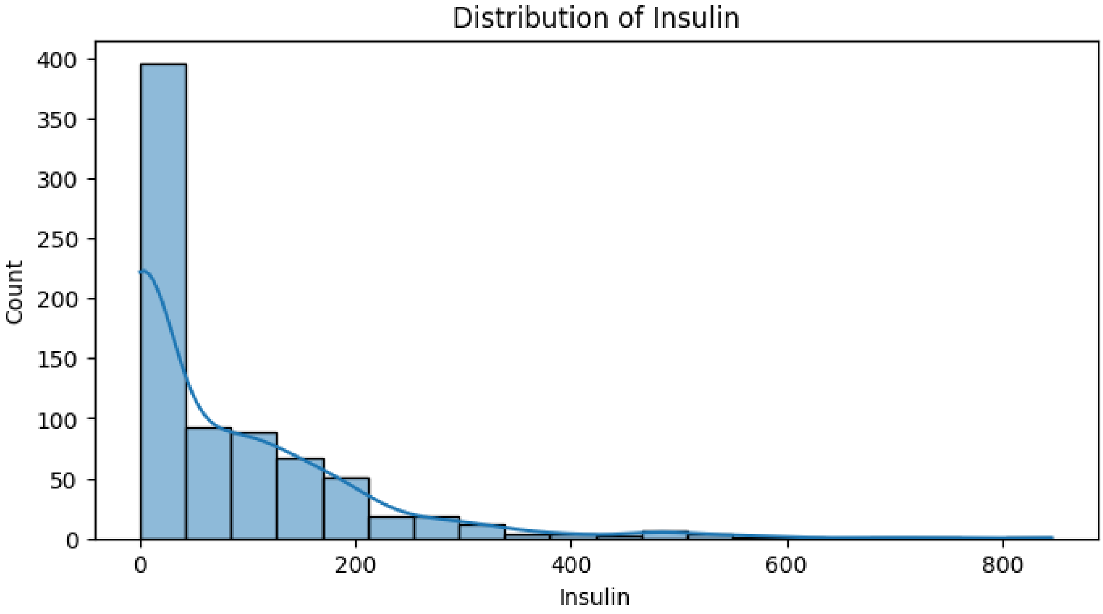
```
for column in df.select_dtypes(include=['int64', 'float64']):
    plt.figure(figsize=(8,4))
    sns.histplot(df[column], bins =20, kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()
```

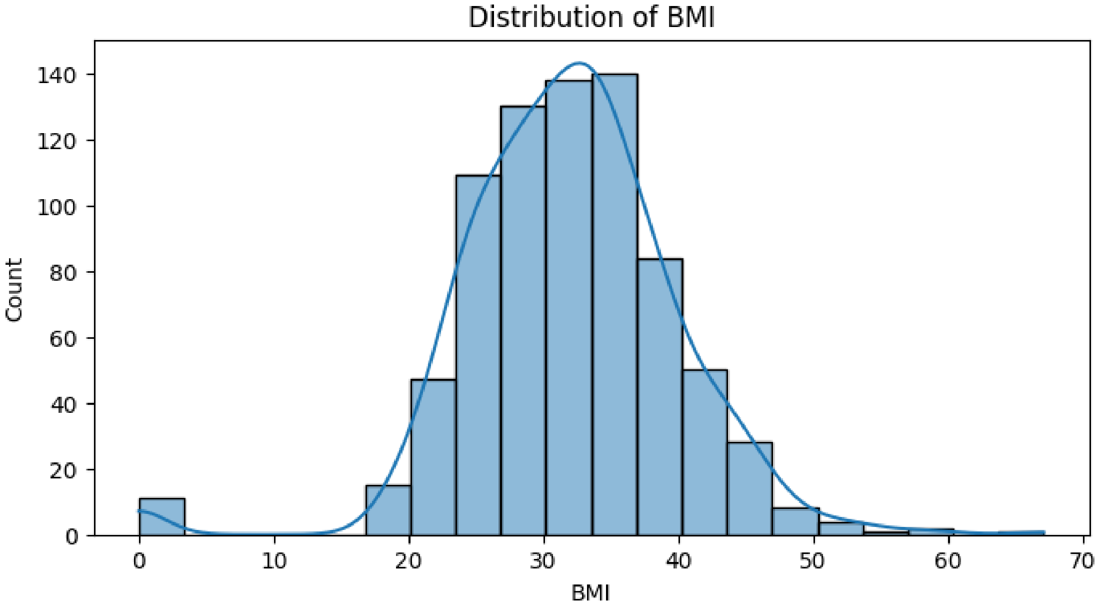

Distribution of Pregnancies
Count
0
20
40
60
80
100
120
140
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Pregnancies

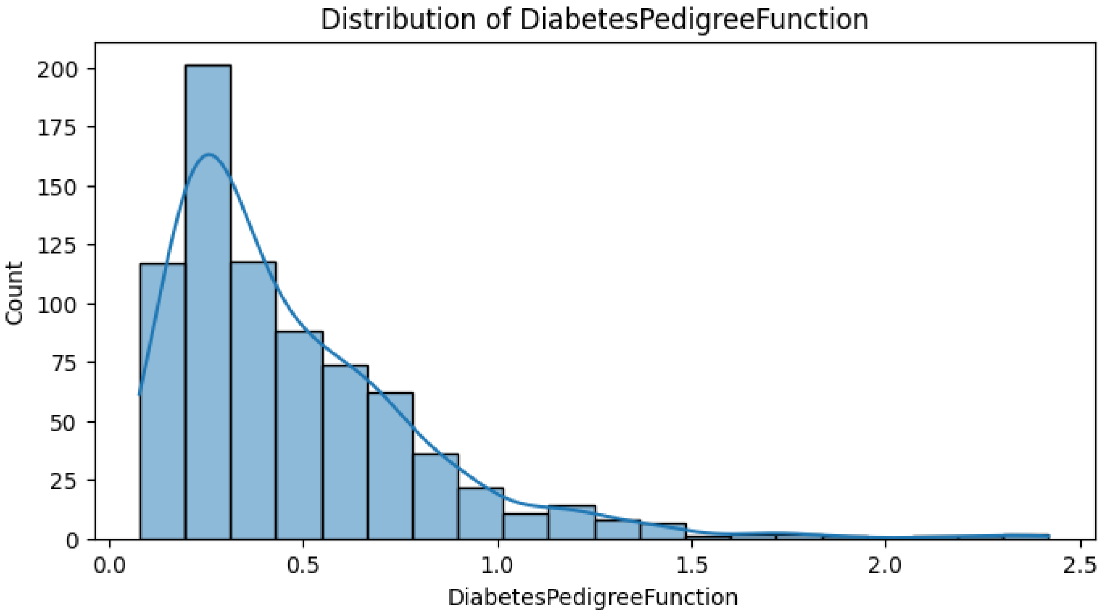

Distribution of Glucose
Count
0
20
40
60
80
100
120
0
25
50
75
100
125
150
175
200
Glucose

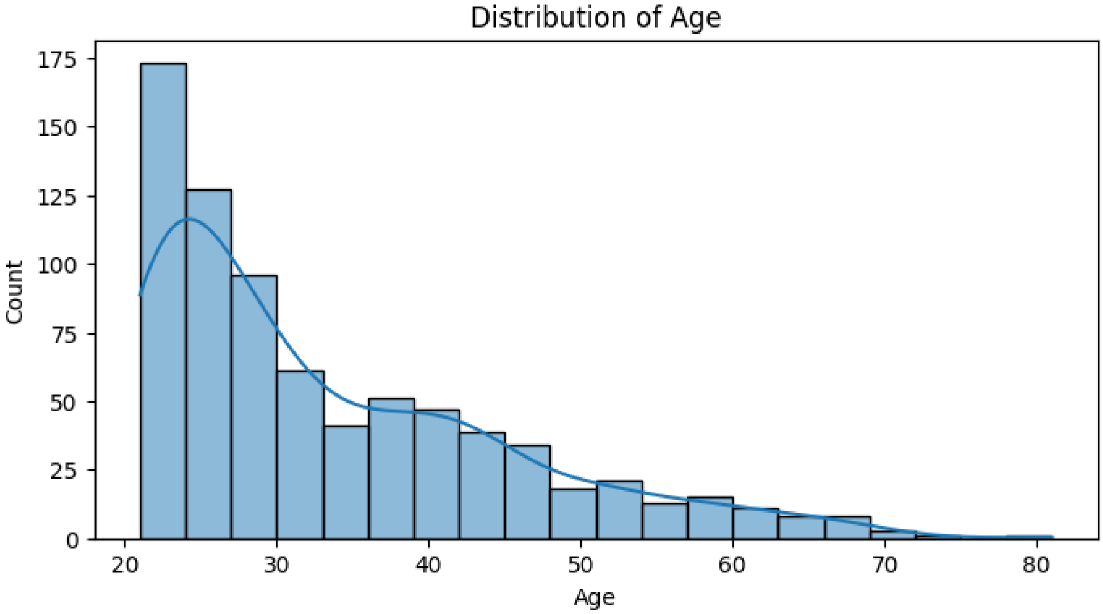

Distribution of BloodPressure
Count
0
20
40
60
80
100
120
140
BloodPressure
0
20
40
60
80
100
120

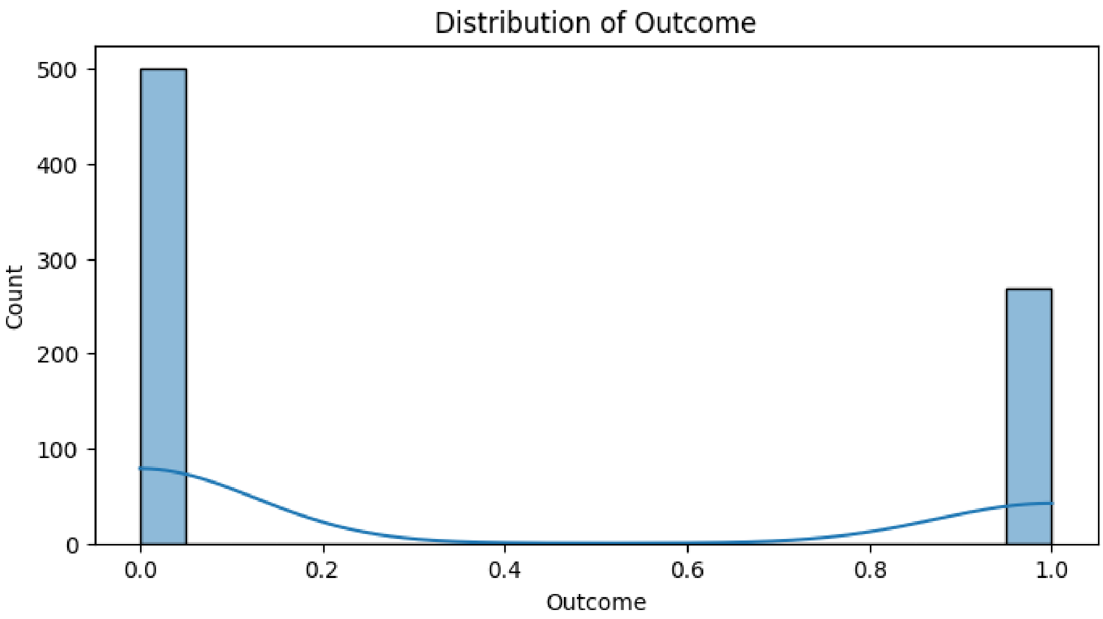

Distribution of SkinThickness
Count
0
50
100
150
200
SkinThickness
0
20
40
60
80
100


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


Distribution of BMI
Count
140
120
100
80
60
40
20
0
0
10
20
30
40
50
60
70
BMI


Distribution of DiabetesPedigreeFunction
Count
0
25
50
75
100
125
150
175
200
0.0
0.5
1.0
1.5
2.0
2.5
DiabetesPedigreeFunction

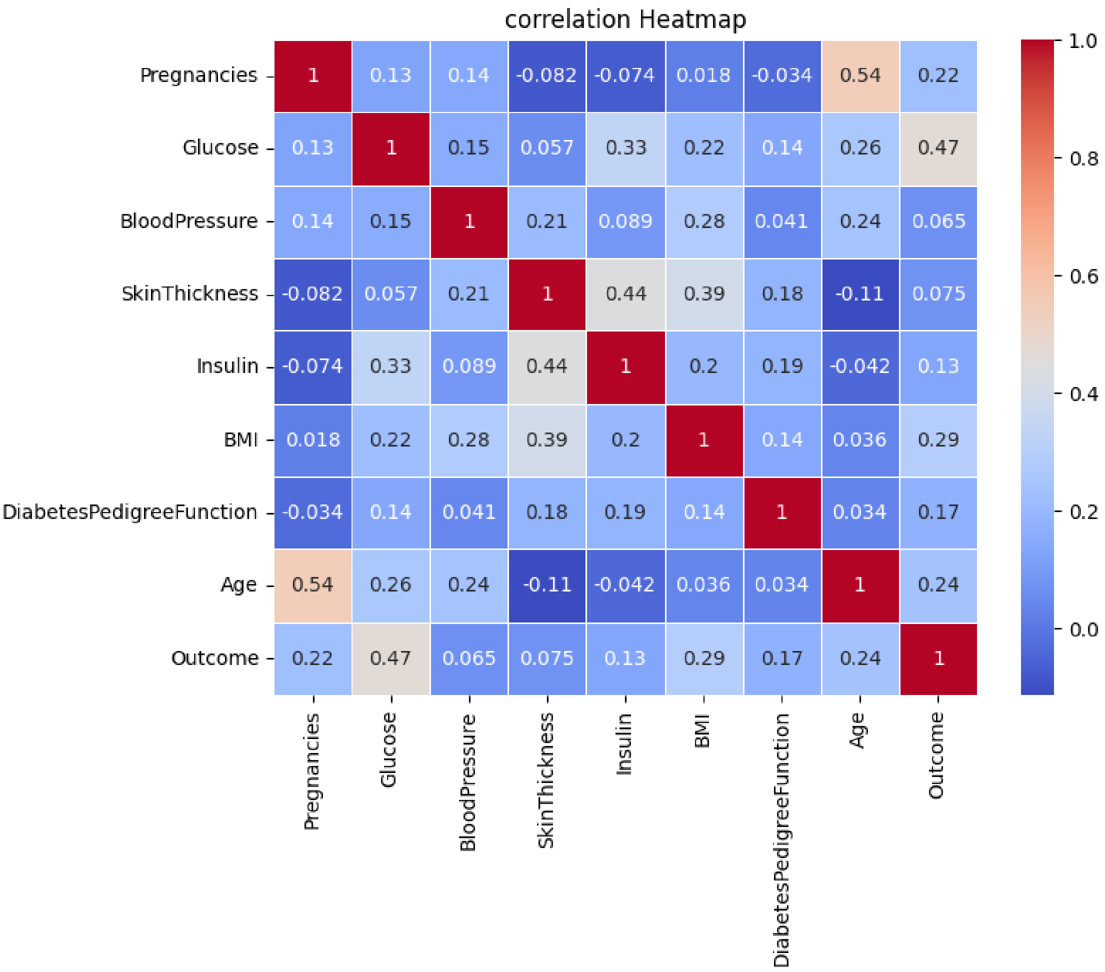

Distribution of Age
Count
0
25
50
75
100
125
150
175
20
30
40
50
60
70
80
Age



8 Correlation heatmap to visualize relationships between numerical variables

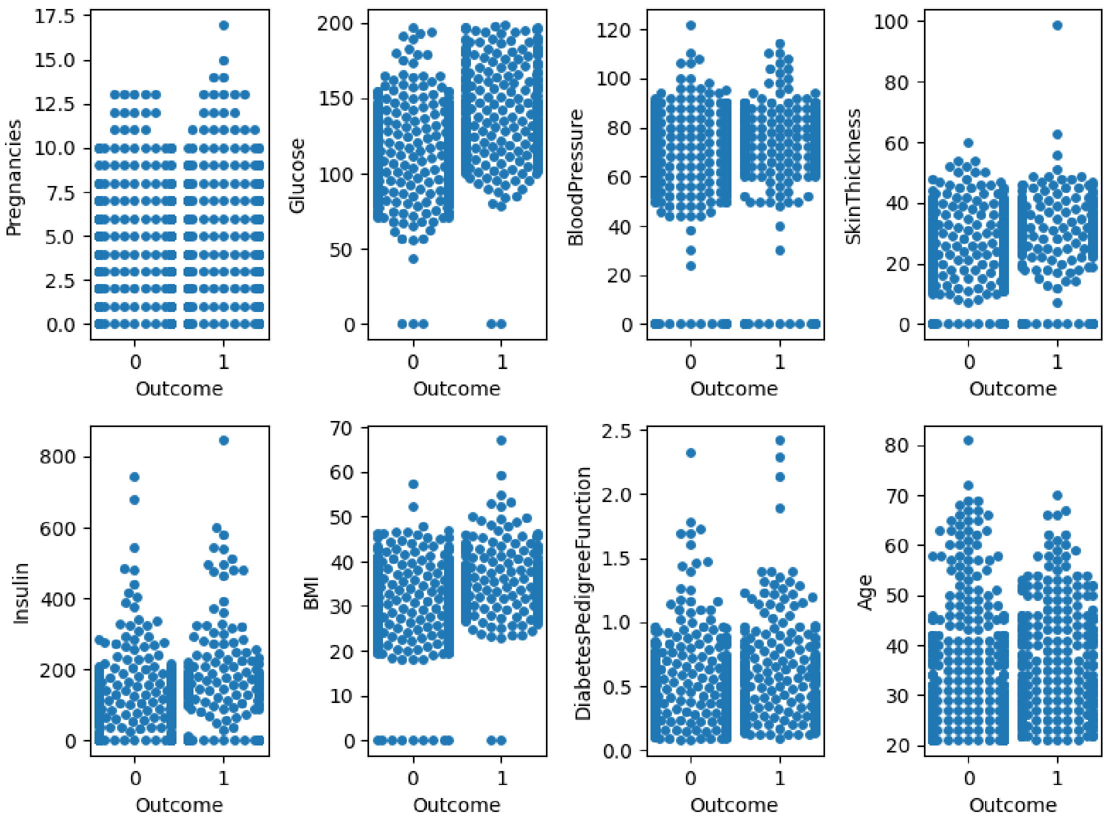
[41]:

```
correlation_matrix =df.corr()
plt.figure(figsize=(8,6))
sns.heatmap(correlation_matrix, annot =True , cmap='coolwarm' , linewidths =0.5)
plt.title('correlation Heatmap')
plt.show()
```



9 feature distribution by diabetes outcome

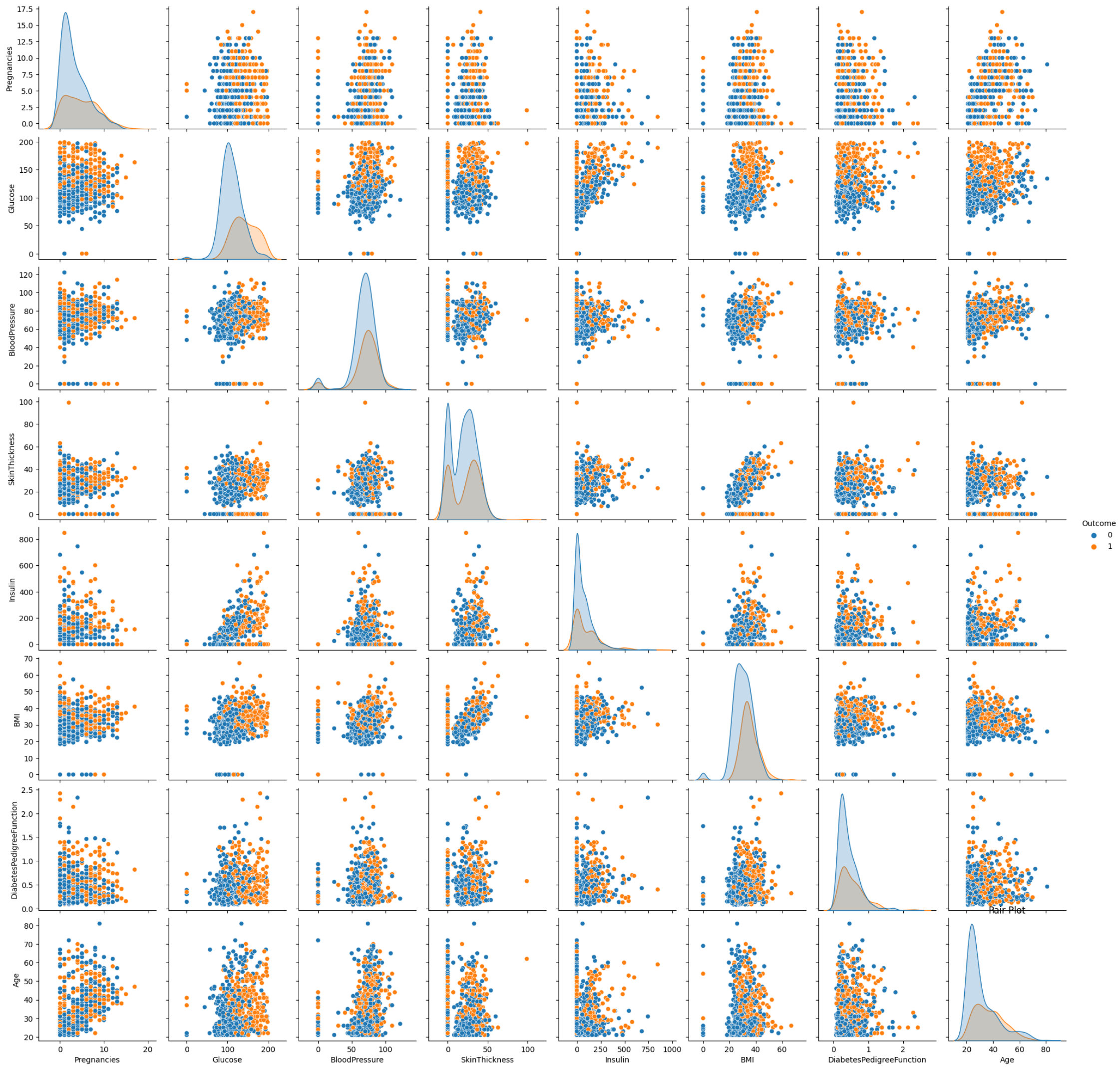
[64]:
```
plt.figure(figsize=(8,6))
for i, col in enumerate(['Pregnancies', 'Glucose', 'BloodPressure',␣
 ↪'SkinThickness', 'Insulin','BMI', 'DiabetesPedigreeFunction', 'Age']):
    plt.subplot(2,4, i + 1)
    sns.swarmplot(x='Outcome', y=col , data=df)
    plt.xlabel('Outcome')
    plt.ylabel(col)
plt.tight_layout()
plt.show()
```



10 Pair Plot of Features with Outcome Comparison

[61]:

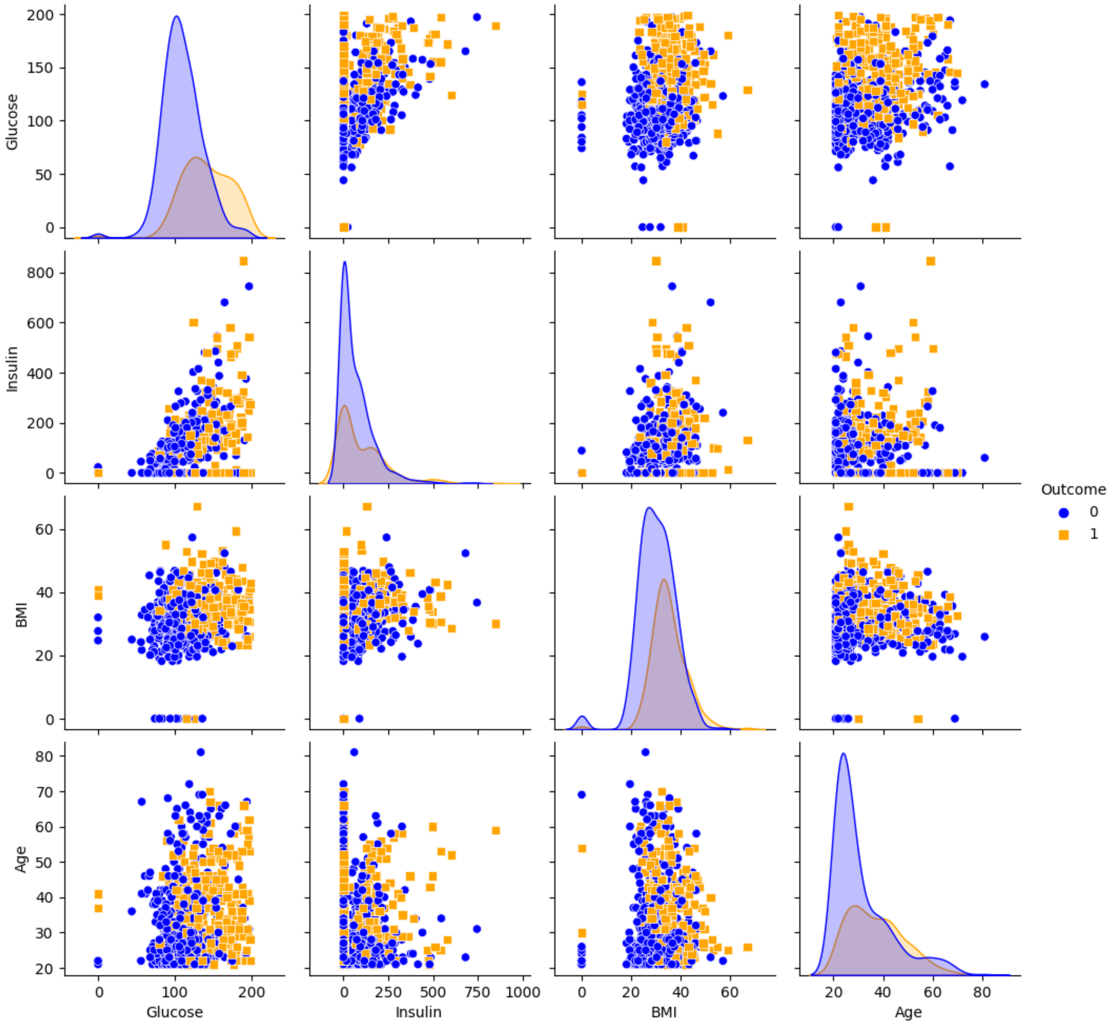
```
sns.pairplot(df, hue='Outcome', diag_kind = 'kde')
plt.title('Pair Plot')
plt.show()
```



11 Pair Plot of Glucose , Insulin , BMI , Age, and Outcome”

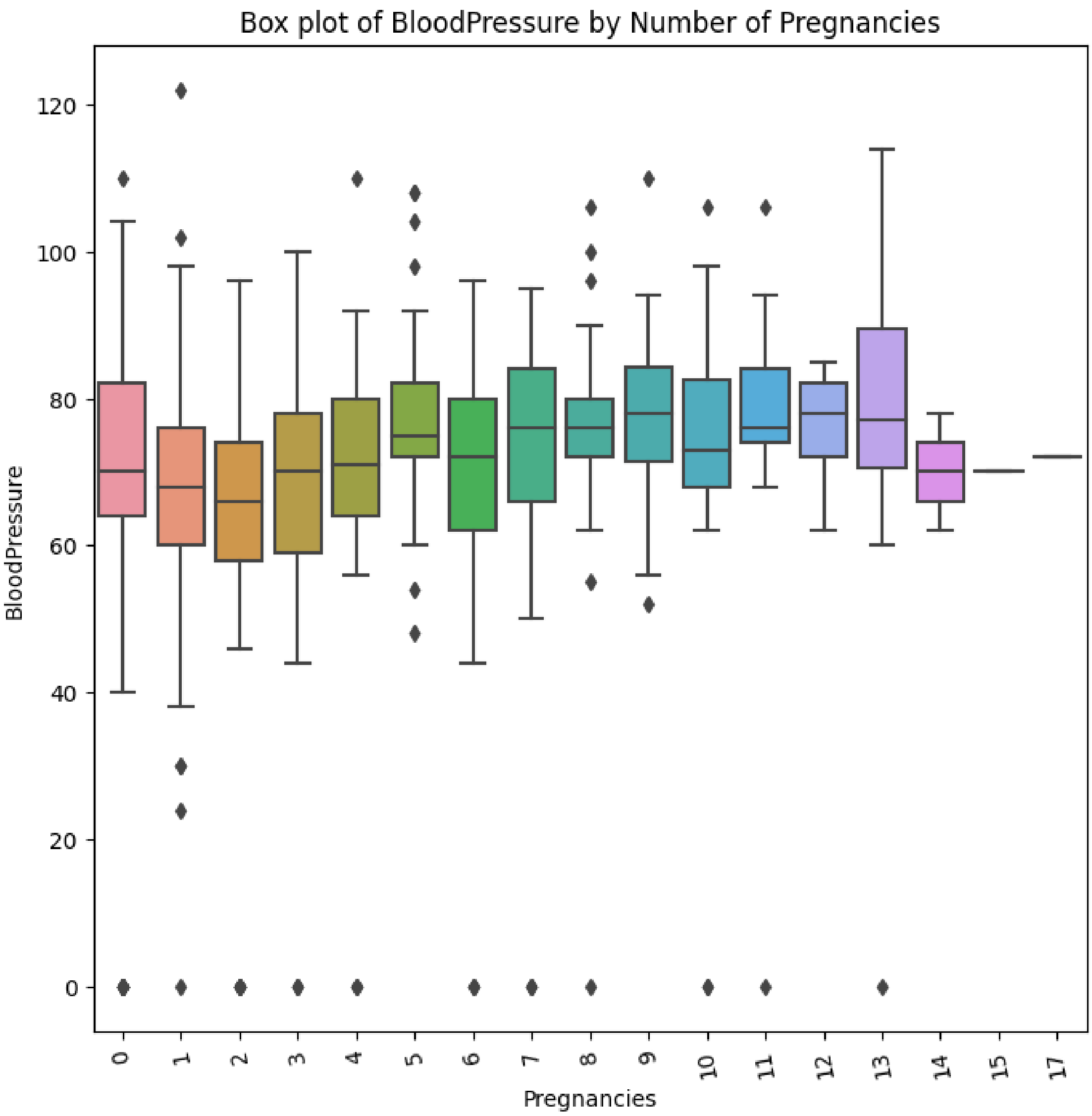
[67]:

```
columns_to_include = ['Glucose', 'Insulin', 'BMI', 'Age', 'Outcome']



sns.pairplot(df[columns_to_include], hue='Outcome', markers=["o", "s"],␣
 ↪palette={0: 'blue', 1: 'orange'})
plt.show()
```



12 Box Plot of BloodPressure by Number of Pregnancies

[69]:
```
plt.figure(figsize=(8,8))
sns.boxplot(x='Pregnancies', y='BloodPressure', data=df)
plt.xlabel('Pregnancies')
plt.ylabel('BloodPressure')
plt.title('Box plot of BloodPressure by Number of Pregnancies')
plt.xticks(rotation = 820)
plt.show()
```



13 Conclusion

[ ]:

```
I performed a comprehensive exploratory data analysis (EDA) on a dataset␣
  ↪related to diabetes patients. First,
I imported essential libraries such as Pandas for data manipulation and Seaborn␣
  ↪for visualization.
After loading the dataset from a CSV file, I examined its dimensions, finding␣
  ↪768 rows and 9 columns.
Utilizing df.describe(), I obtained summary statistics for numerical columns,␣
  ↪offering insights into data
```

```
characteristics. Moreover, I confirmed the absence of missing values with df.info(). The data cleaning step
revealed no null values. To understand data distribution and relationships, I created various visualizations,
including histograms, density plots, a correlation heatmap, swarm plots, and a pair plot.
I customized a specific pair plot for key columns and generated a box plot to explore blood pressure variations
based on the number of pregnancies. This comprehensive analysis provides valuable insights into the dataset, aiding
in data-driven decisions and potential modeling for diabetes prediction.
```