Diabetes Prediction using Machine Learning

---Maitreya Sameer Ganu , IISER Thiruvananthapuram


```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

data =
pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.csv
')
data.head()
```

```
   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


   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
```

Let's visualize our data

```
data
```

```
     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]
```

```
data.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
```

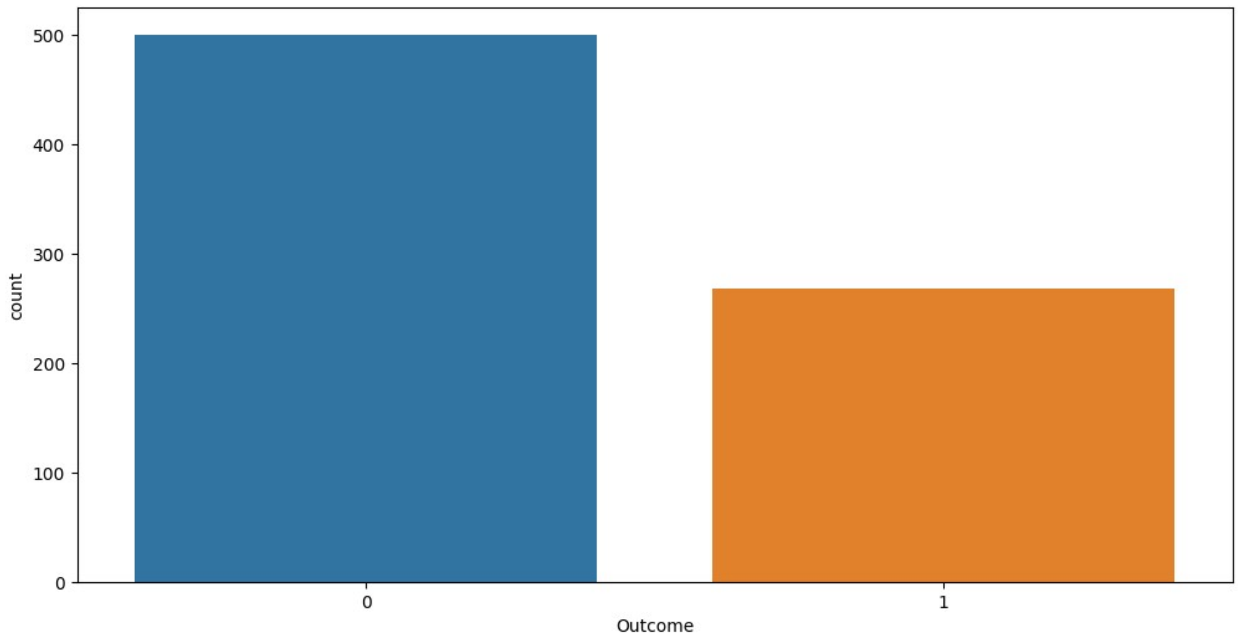
```
data.info()
```

```
<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
```

Clearly , there are no missing values and values which are null

```
plt.figure(figsize = (12,6))
sns.countplot(x = 'Outcome' , data = data)
```

```
<Axes: xlabel='Outcome', ylabel='count'>
```

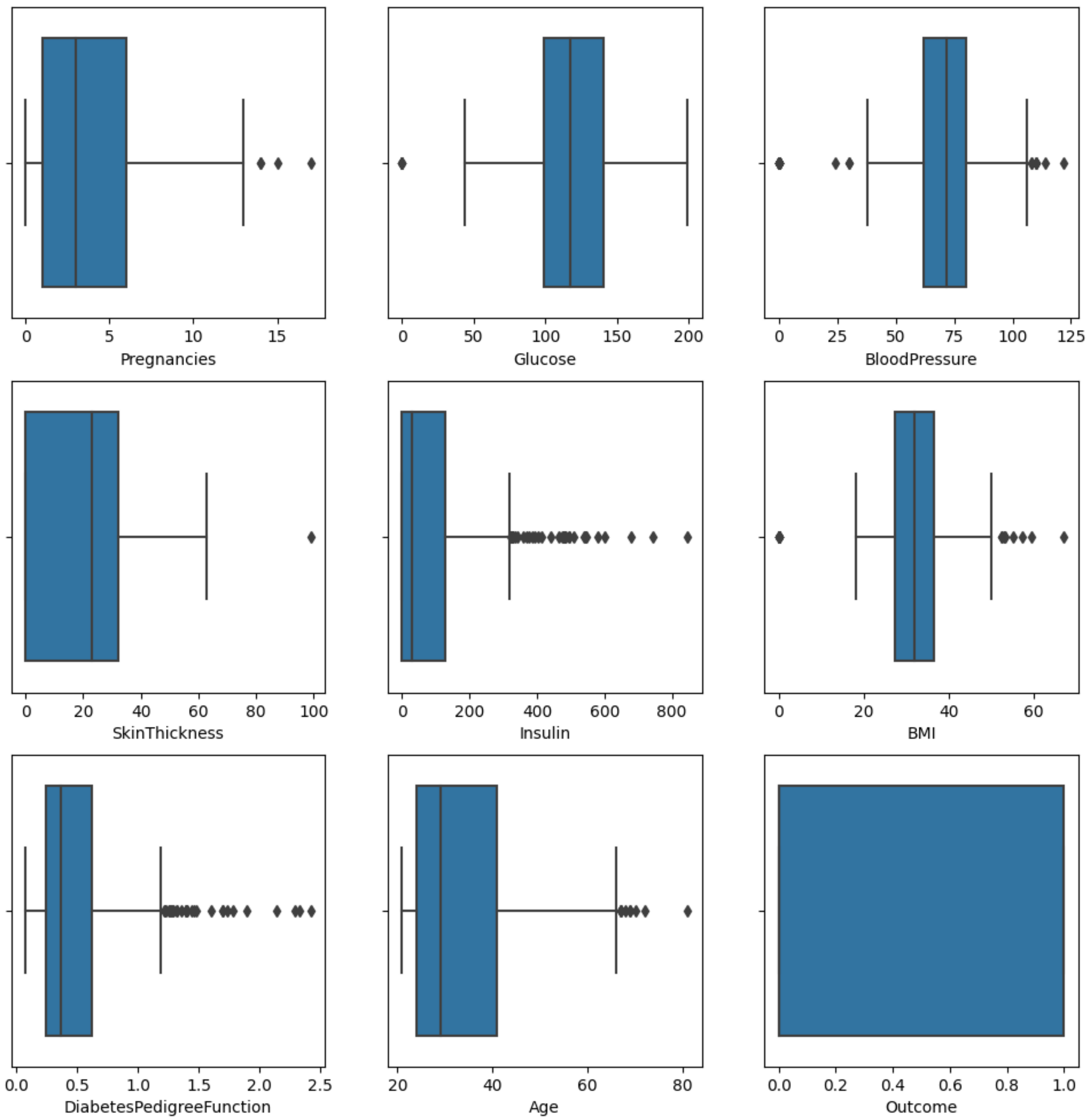


Let's observe any outliers

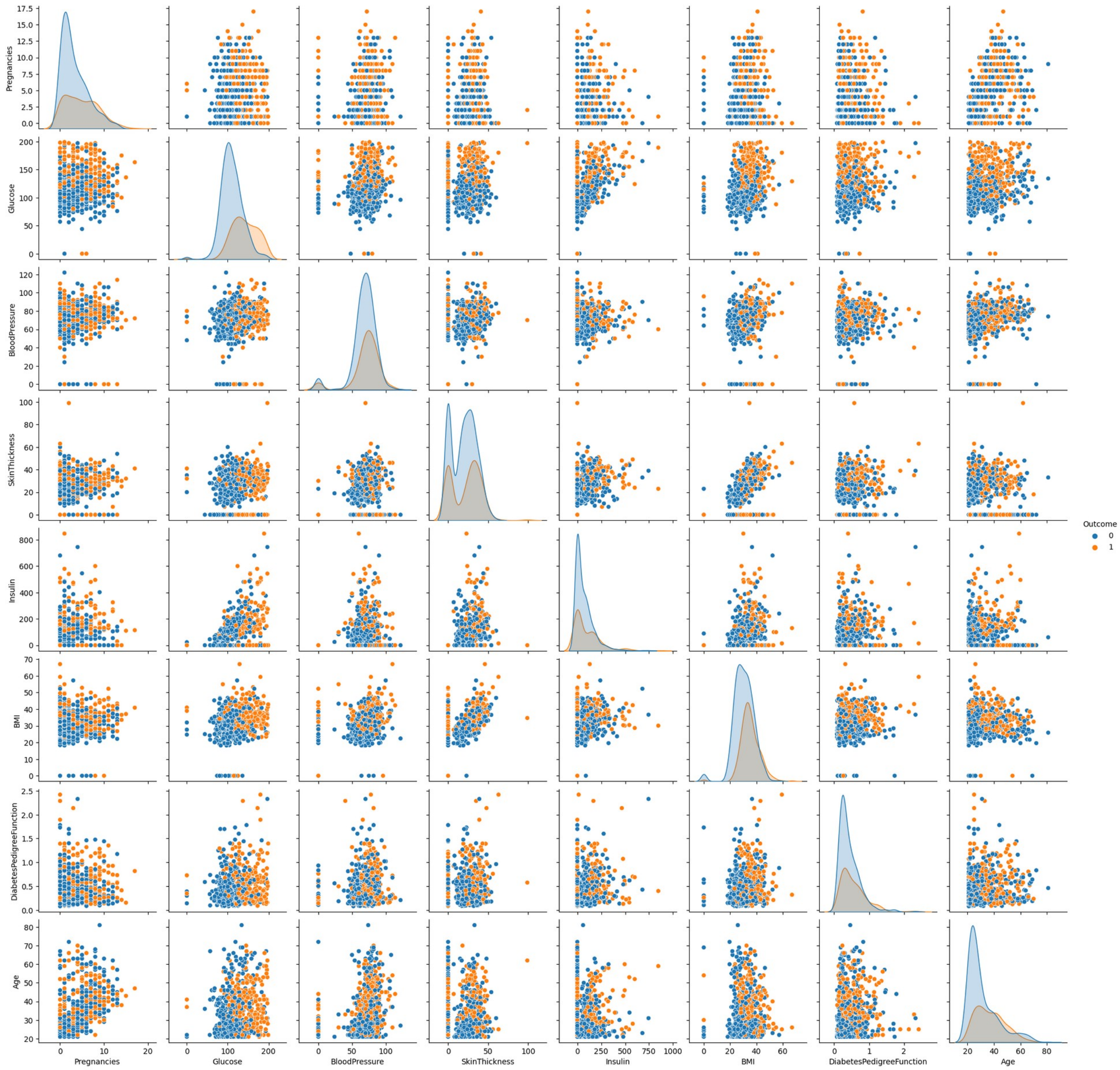
```
data.columns
```

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

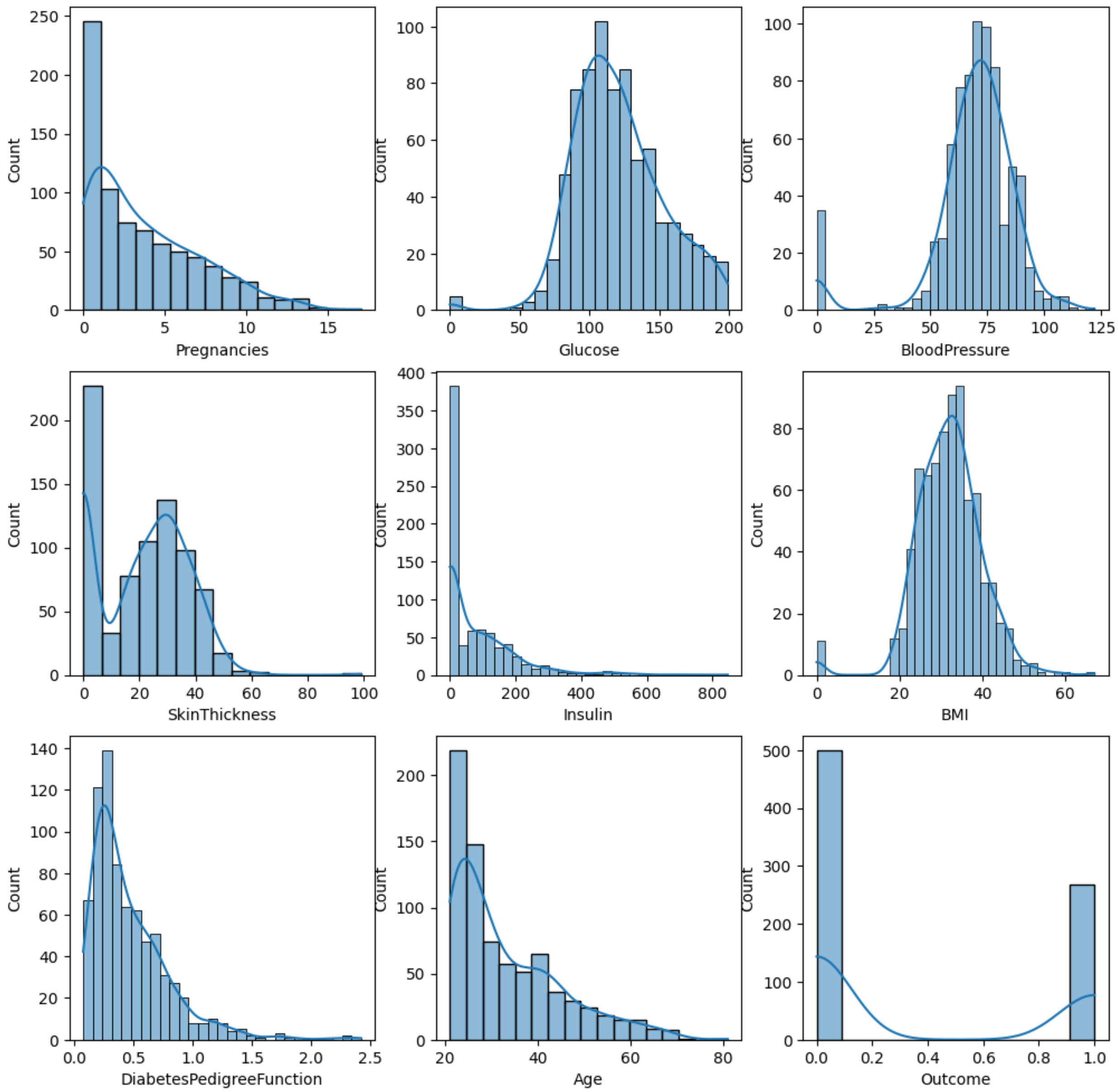
```
plt.figure(figsize = (12,12))
for i,col in enumerate(['Pregnancies', 'Glucose', 'BloodPressure',
'SkinThickness', 'Insulin','BMI', 'DiabetesPedigreeFunction', 'Age',
'Outcome']):
    plt.subplot(3,3 , i+1)
    sns.boxplot(x = col , data = data)
plt.show()
```



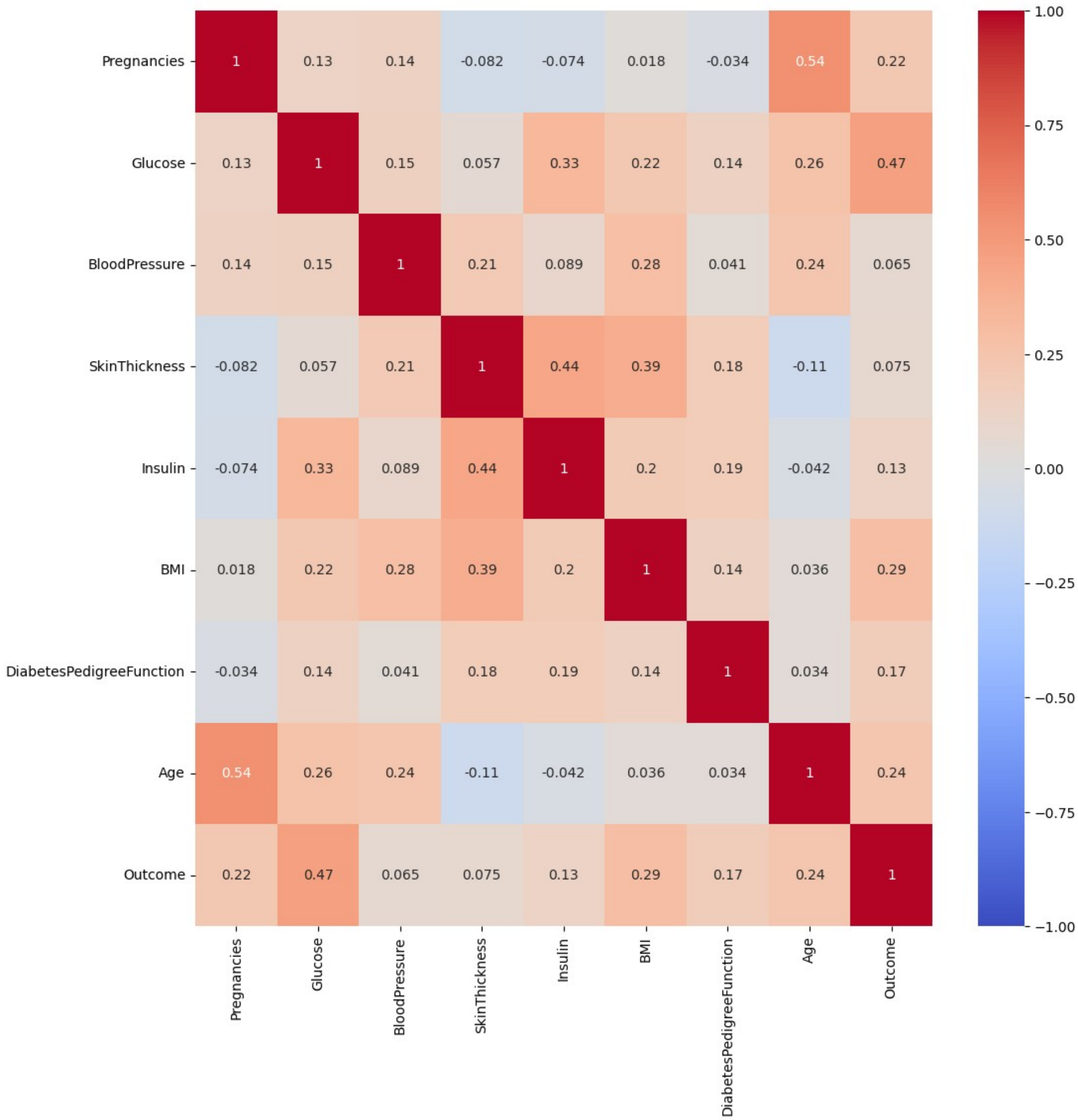

```
import warnings

warnings.filterwarnings('ignore')
sns.pairplot(data , hue = 'Outcome')
plt.show()
```




```
plt.figure(figsize = (12,12))
for i,col in enumerate(['Pregnancies', 'Glucose', 'BloodPressure',
'SkinThickness', 'Insulin','BMI', 'DiabetesPedigreeFunction', 'Age',
'Outcome']):
    plt.subplot(3,3 , i+1)
    sns.histplot(x = col , data = data , kde = True)
plt.show()
```




```
plt.figure(figsize = (12,12))
sns.heatmap(data.corr() , vmin = -1 , center = 0 , cmap = 'coolwarm' ,
annot = True)
plt.show()
```



Standard Scaling and Label Encodings

```
from sklearn.preprocessing import StandardScaler
import warnings

warnings.filterwarnings('ignore')
sc_X = StandardScaler()
X = pd.DataFrame(sc_X.fit_transform(data.drop(['Outcome'],axis =
1),),columns = ['Pregnancies', 'Glucose', 'BloodPressure',
'SkinThickness', 'Insulin','BMI', 'DiabetesPedigreeFunction', 'Age'])
```

```
X.head()
```

```
   Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin
BMI  \
0     0.639947  0.848324       0.149641       0.907270 -0.692891
0.204013
1    -0.844885 -1.123396      -0.160546       0.530902 -0.692891 -
0.684422
2     1.233880  1.943724      -0.263941      -1.288212 -0.692891 -
1.103255
3    -0.844885 -0.998208      -0.160546       0.154533  0.123302 -
0.494043
4    -1.141852  0.504055      -1.504687       0.907270  0.765836
1.409746

   DiabetesPedigreeFunction       Age
0                  0.468492  1.425995
1                 -0.365061 -0.190672
2                  0.604397 -0.105584
3                 -0.920763 -1.041549
4                  5.484909 -0.020496
```

```
y = data['Outcome']
```

```
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 = 0)
```

```
from sklearn.neighbors import KNeighborsClassifier
test_scores = []
train_scores = []
for i in range(1,15):
    knn = KNeighborsClassifier(i)
    knn.fit(X_train , y_train)
    train_scores.append(knn.score(X_train,y_train))
    test_scores.append(knn.score(X_test,y_test))
```

```
max_train_score = max(train_scores)
train_scores_index = [i for i,v in enumerate(train_scores) if v ==
max_train_score]
print("Maximum Train Score {} % and k = {}".format(max_train_score*100
, list(map(lambda x: x+1 , train_scores_index))))
```

```
Maximum Train Score 100.0 % and k = [1]
```

```
max_test_score = max(test_scores)
test_scores_index = [i for i,v in enumerate(test_scores) if v ==
max_test_score]
print("Maximum Test Score {} % and k = {}".format(max_test_score*100 ,
list(map(lambda x: x+1 , test_scores_index))))
```

```
Maximum Test Score 80.51948051948052 % and k = [5]
```

```python
import warnings

warnings.filterwarnings('ignore')
plt.figure(figsize=(12, 5))
sns.lineplot(x=range(1, 15), y=train_scores, marker='o', label='Train scores')
sns.lineplot(x=range(1, 15), y=test_scores, marker='o', label='Test scores')
plt.title('Train vs Test Scores', fontsize=14)
plt.xlabel('Model Complexity', fontsize=12)
plt.ylabel('Score', fontsize=12)

plt.grid(True)
plt.legend()
plt.show()
```

Here , for k = 1 we are getting the highest train score and for k = 5 ,we are getting the highest test score

```python
knn  = KNeighborsClassifier(5)
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

```
0.8051948051948052
```

```python
from sklearn.metrics import confusion_matrix,classification_report
y_pred = knn.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[94 13]
 [17 30]]
              precision    recall  f1-score   support

           0       0.85      0.88      0.86       107
           1       0.70      0.64      0.67        47

    accuracy                           0.81       154
   macro avg       0.77      0.76      0.76       154
weighted avg       0.80      0.81      0.80       154
```

Clearly , we got a pretty accurate output.

Here , 20% of the data was used for testing purposes taking into consideration the Pareto's Principle.