In [2]:

```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:

```
#importing the dataset
data = pd.read_csv('C:/Users/SamDutse/Desktop/Curent Work/Diabetes/diabetes.csv')
```

In [5]:

```
#head of the data
data.head()
```

Out[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

In [6]:

```
#checking information about data and data types
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
```

In [7]:

```
#checking the dimension of the data
data.shape
```

Out[7]:

```
(768, 9)
```

In [8]:

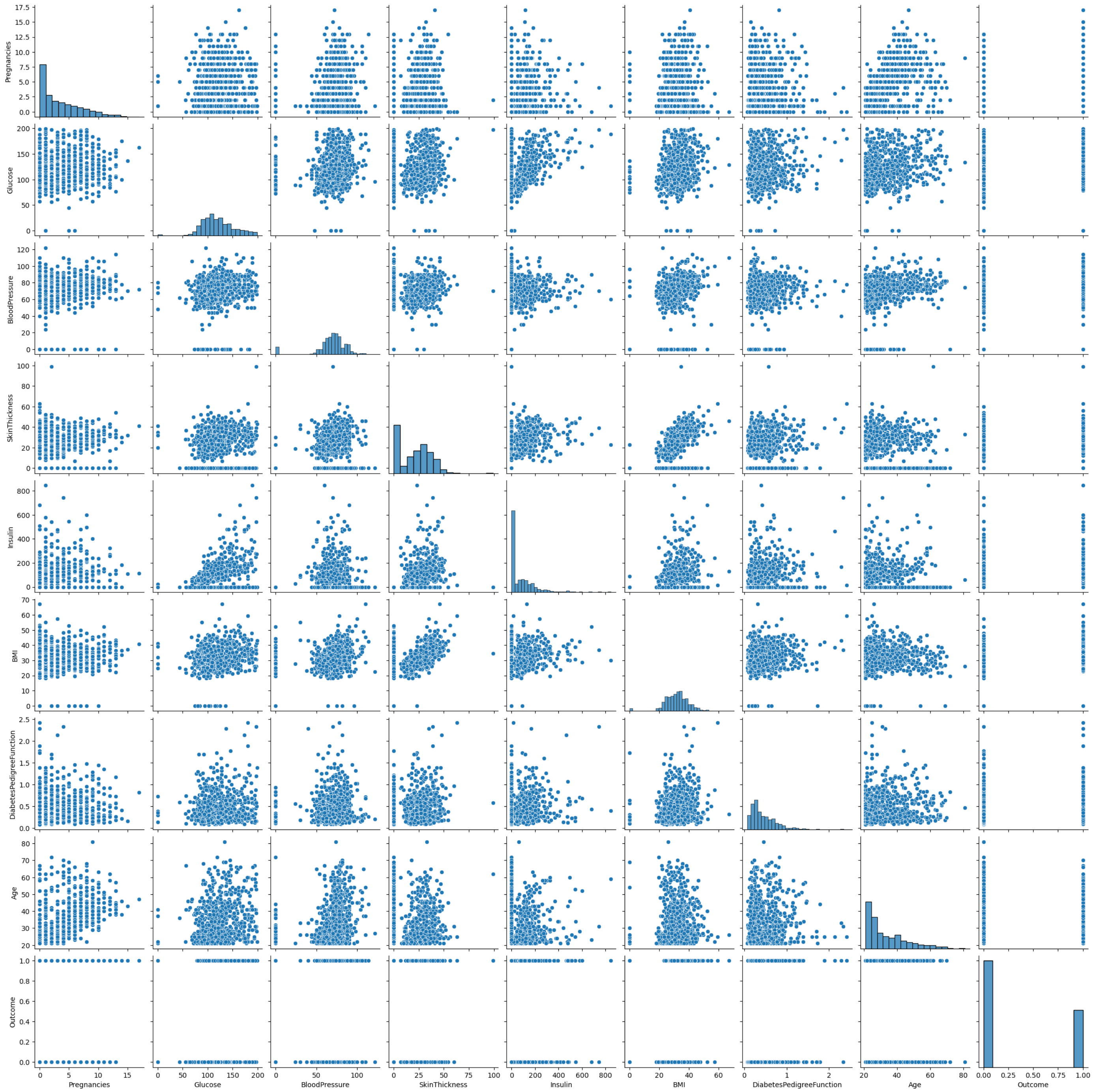
```
#checking for null values in the data
data.isnull().sum()
```

Out[8]:

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

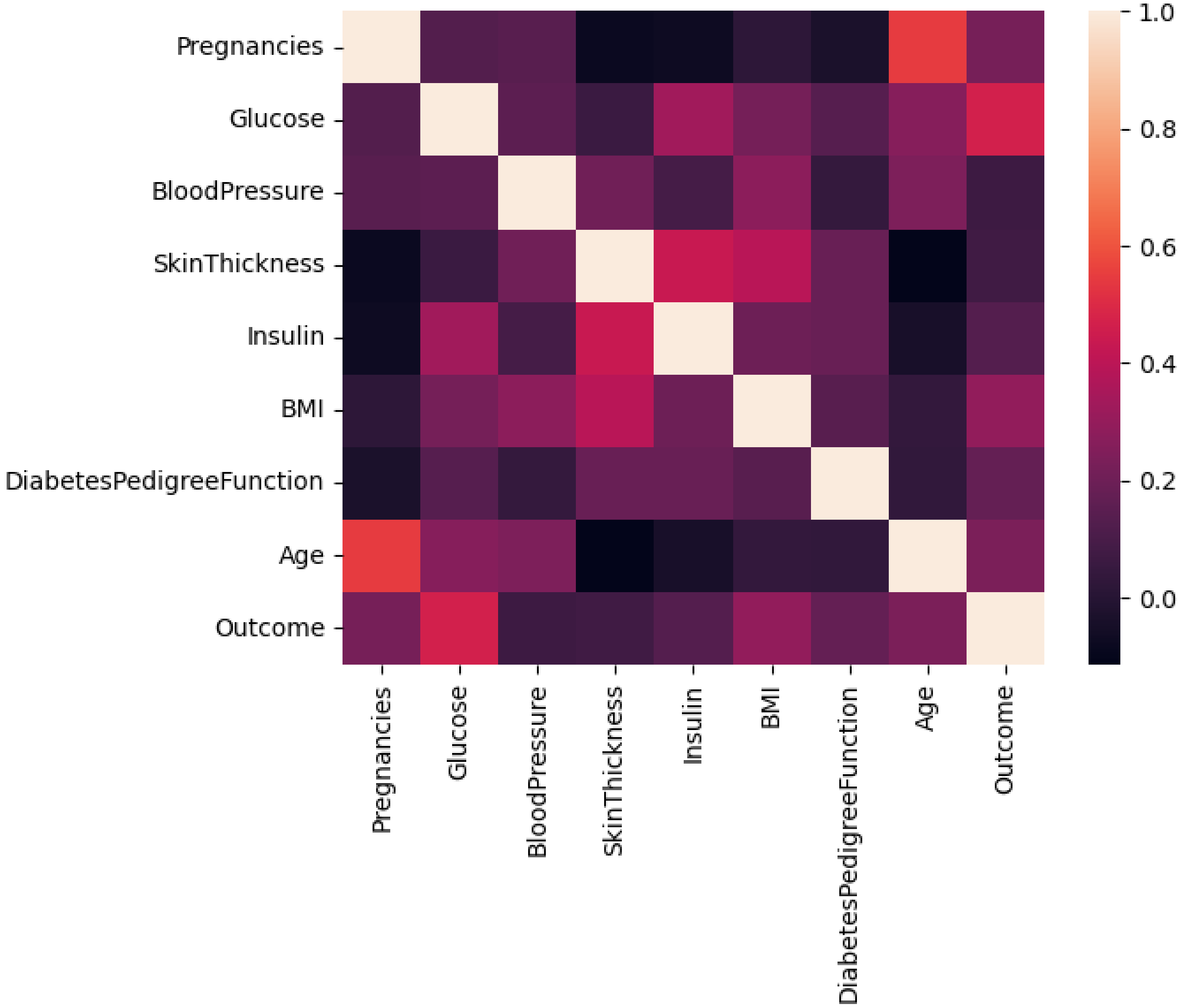
```
In [17]: #data visualization
         sns.pairplot(data = data)

Out[17]: <seaborn.axisgrid.PairGrid at 0x2795fec0790>
```



```
In [7]: #Building the correlation matrix
        sns.heatmap(data.corr())

Out[7]: <AxesSubplot:>
```



In [18]:

```
#importing model and model selection
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [19]:

```
#splitting the data into training and testing dataset
X = data.drop(["Outcome"], axis = 1) #training data
y = data["Outcome"] #testing data
```

In [21]:

```
#data splicing
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.3, random_state
```

In [22]:

```
#feature scaling
from sklearn.preprocessing import StandardScaler
Sc_X = StandardScaler()
X_train = Sc_X.fit_transform(X_train)
X_test = Sc_X.transform(X_test)
```

In [23]:

```
Classifier = LogisticRegression(random_state = 0)
Classifier.fit(X_train, y_train)
```

Out[23]:

```
LogisticRegression(random_state=0)
```

In [24]:

```
#predicting the test set result
y_pred = Classifier.predict(X_test)
y_pred
```

Out[24]:

```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0], dtype=int64)
```

In [25]:

```
#finding model accuracy  we use the confucion matrix
from sklearn.metrics import confusion_matrix
Cm = confusion_matrix(y_test, y_pred)
Cm
```

Out[25]:

```
array([[141,  16],
       [ 35,  39]], dtype=int64)
```

In [27]:

```
#predicting accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred) * 100
```

Out[27]:

```
77.92207792207793
```

In [ ]:

```

```