In [17]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df=pd.read_csv("diabetes.csv")
df.head()
```

Out[2]:

	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 [3]:

```
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]:

```
df.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.00000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.47187
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.33132
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.07800
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.24375
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.37250
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.62625
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.42000

CHECKING FOR MISSING VALUES

In [6]:

```
df.isnull().sum()
```

Out[6]:

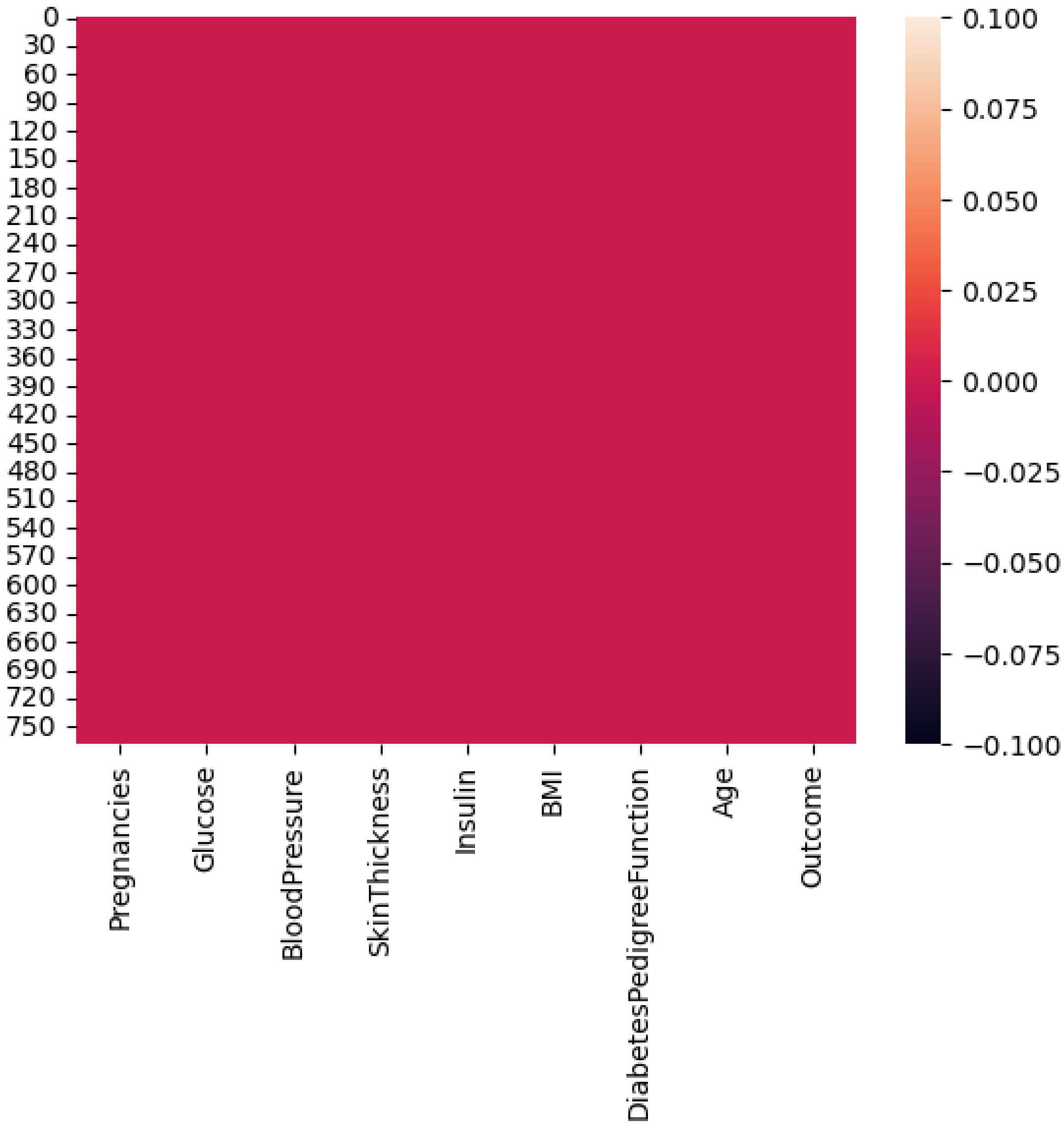
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [12]:

```
sns.heatmap(df.isnull())
```

Out[12]:

```
<AxesSubplot:>
```



CO RELATION MATRIX

In [14]:

```
df.corr()
```

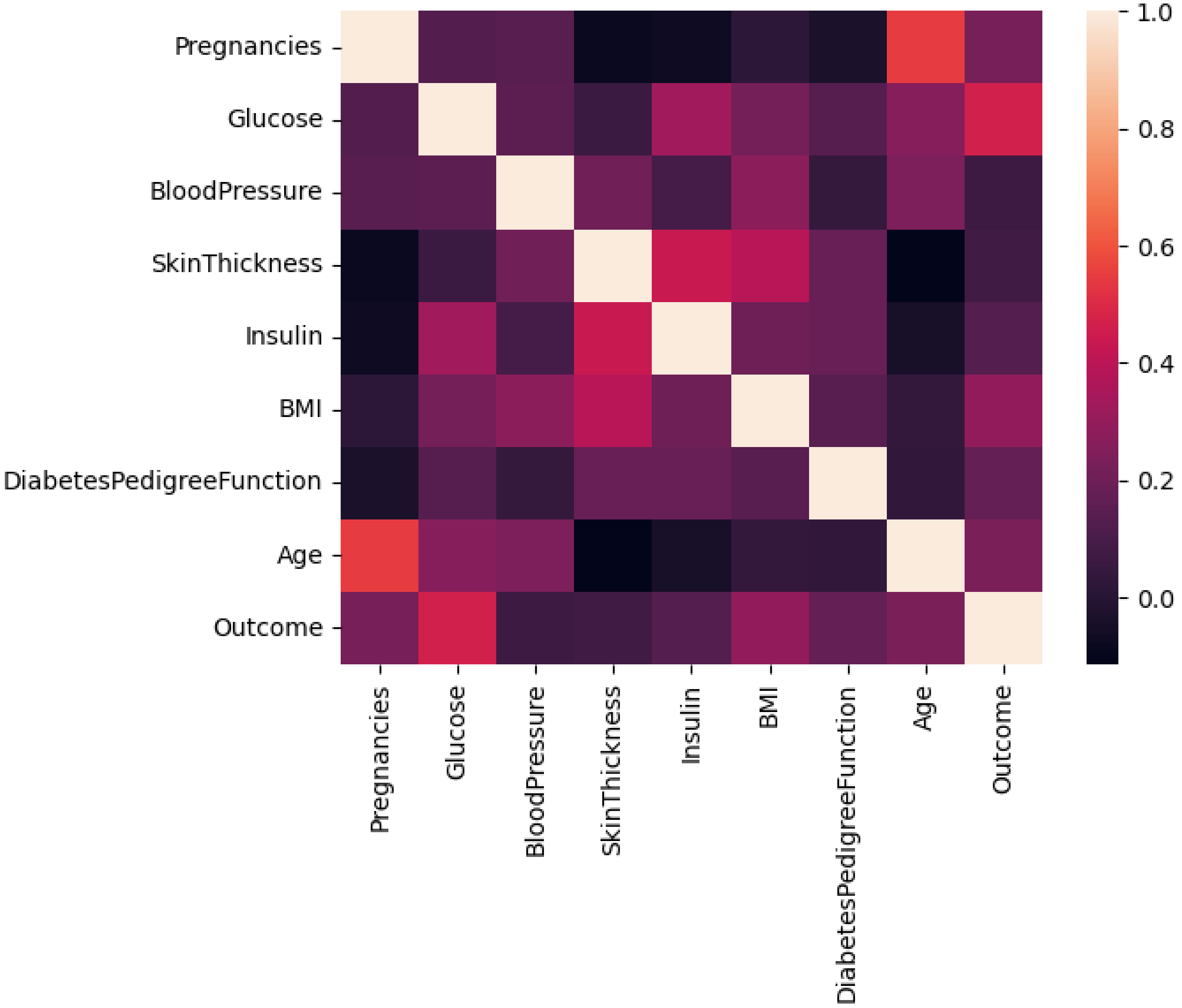
Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesP
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	

Visualizing the correlation

In [16]:

```
sns.heatmap(df.corr()).cmap="pink"
```



TRAINING THE MODEL WITH THE HELP OF

TRAIN TEST SPLIT

In [20]:
```
x=df.drop('Outcome',axis=1)
y=df['Outcome']
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2)
```

In X all the independent variables are stored In Y the predictor variable(“OUTCOME”) is stored. Train-test split is a technique used in machine learning to assess model performance. It divides the dataset into a training set and a testing set, with a 0.2 test size indicating that 20% of the data is used for testing and 80% for training.

Training the model

In [22]:
```
model=LogisticRegression()
model.fit(x_train,y_train)
```

Out[22]:
```
LogisticRegression()
```

Fitting the X train and y train data into the variable called model.

Making Prediction

In [23]:
```
prediction =model.predict(x_test)
```

In [24]:
```
print(prediction)
```

```
[1 0 1 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 0
 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 0 0 1 0 1 0]
```

The accuracy of the model is then calculated and determined.

In [25]:
```
accuracy=accuracy_score(prediction,y_test)
```

In [26]:
```
print(accuracy)
```

```
0.7987012987012987
```

The accuracy of the model is then calculated and determined.

In [ ]: