Diabetes Prediction

Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Blood glucose is your main source of energy and comes from the food you eat. Insulin, a hormone made by the pancreas, helps glucose from food get into your cells to be used for energy.

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

Content

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

Let's Begin

Import necessary libraries

In [98]:

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

Read 'diabetes.csv' dataset and store it in a DataFrame

In [99]:

```
df=pd.read_csv('diabetes.csv')
df
```

Out[99]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 9 columns

View the top 5 rows

In [100]:

```
df.head()
```

Out[100]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288

View the bottom 5 rows

In [101]:

```
df.tail()
```

Out[101]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

View info about the dataset

In [102]:

```
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
```

View basic statistical information about the dataset

In [103]:
```
df.describe()
```

Out[103]:

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

Check for any null values in the dataset

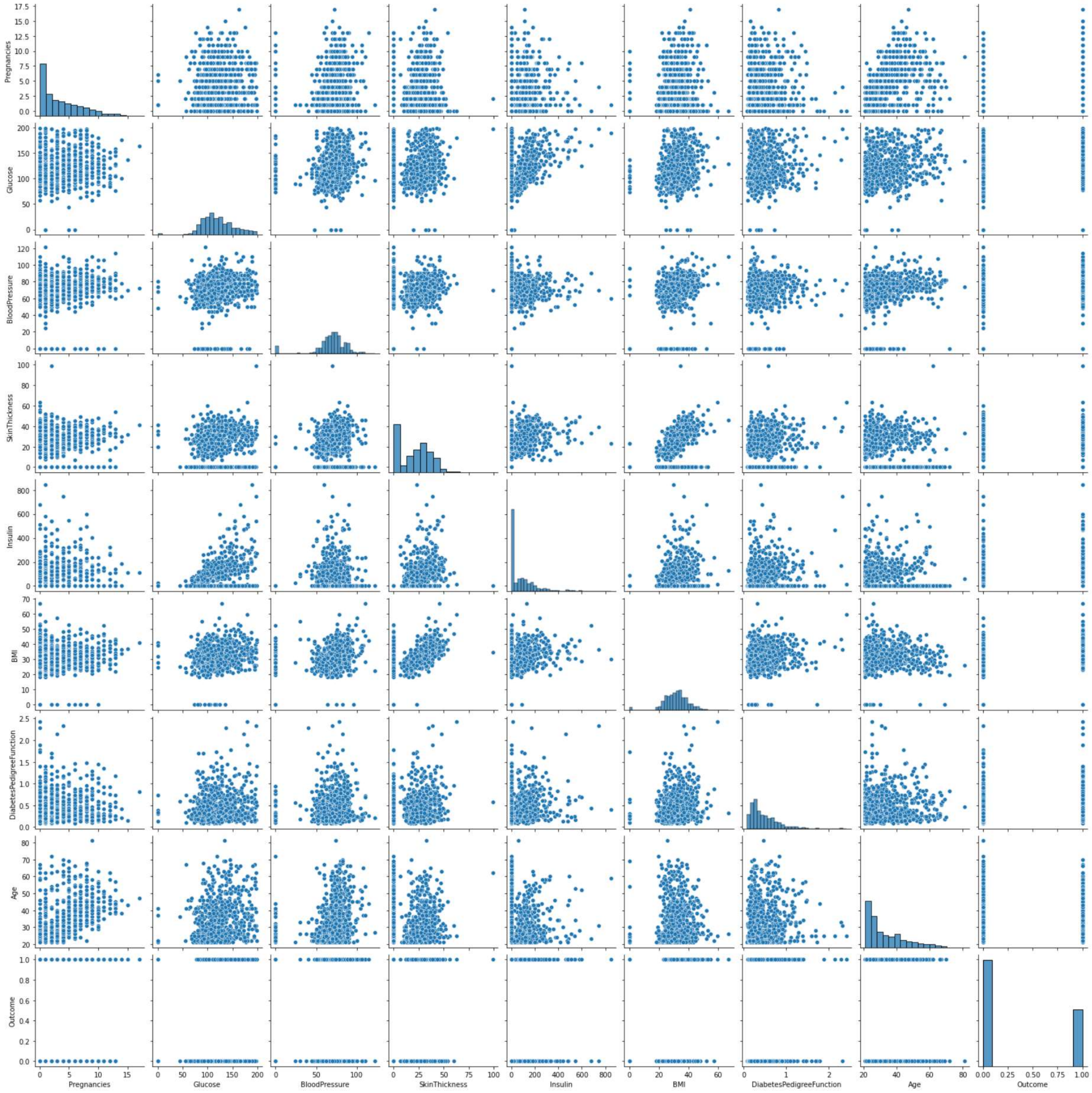
In [104]:
```
df.isna().sum()
```

Out[104]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Visualization

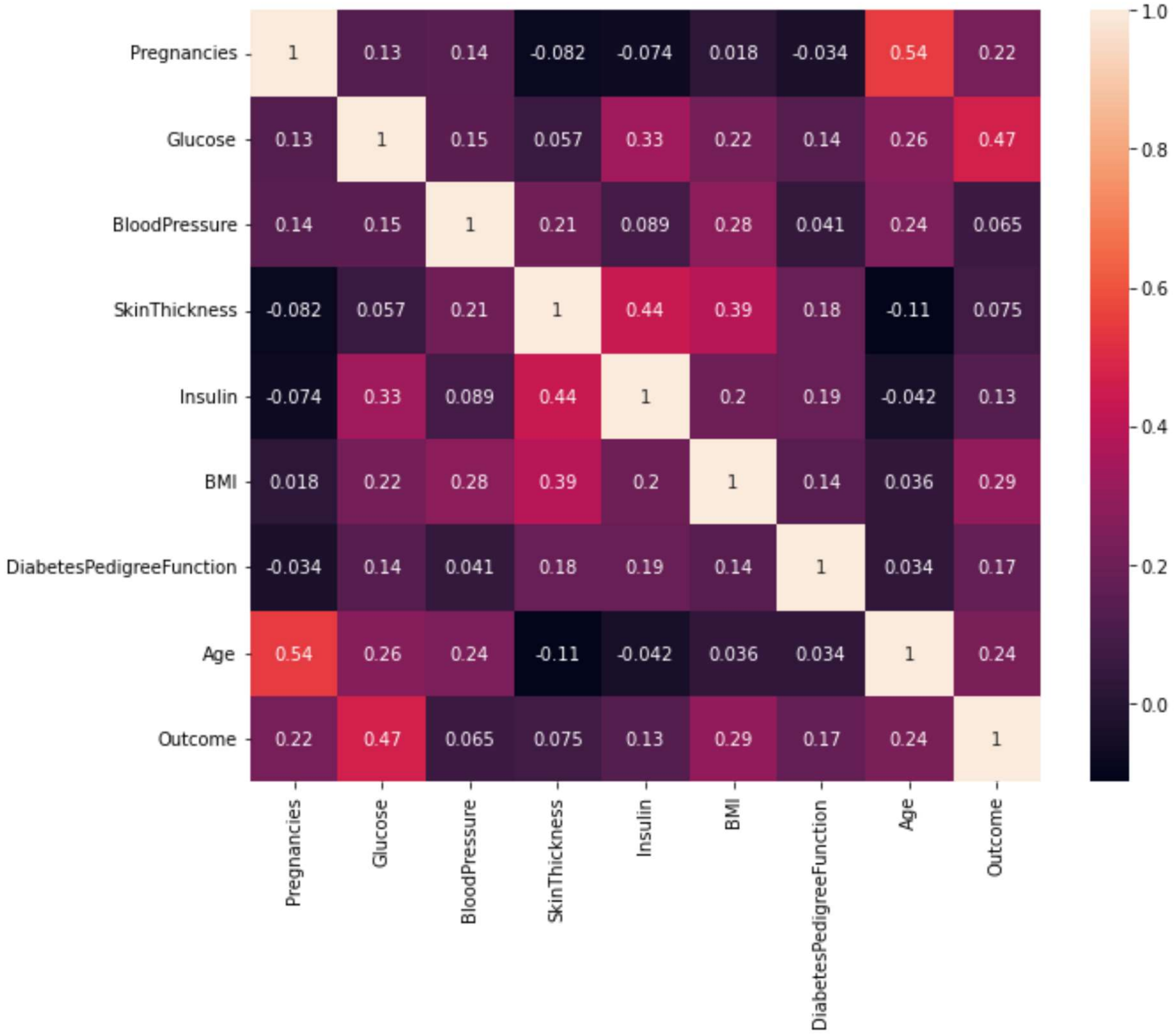
Plot a pairplot of the dataset

```
In [105]: sns.pairplot(df)
          plt.show()
```



Plot a heatmap to view the correlation between the input and target variables

In [106]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



Split the dataset into input and target variables

In [151]:
```
X=df.drop(columns=["Outcome"])
y=df['Outcome']
```

In [152]:
```
X.shape
```

Out[152]:
```
(768, 8)
```

In [153]:

```
y.shape
```

Out[153]:

```
(768,)
```

Standardize the data with StandardScaler

In [154]:

```
from sklearn.preprocessing import StandardScaler
```

In [159]:

```
st=StandardScaler()
xcolumns=X.columns
c=st.fit_transform(X)
pd.DataFrame(df,columns=xcolumns)
```

Out[159]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 8 columns

In [158]:

```
X.head()
```

Out[158]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288

Split the dataset into Training and Testing set

```
In [125]: from sklearn.model_selection import train_test_split
```

```
In [127]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=
```

Check the shape of X_train and X_test

```
In [128]: X_train.shape
```

```
Out[128]: (537, 8)
```

```
In [129]: X_test.shape
```

```
Out[129]: (231, 8)
```

Create a Support Vector Machine model and train it

```
In [131]: from sklearn.svm import SVC
```

```
In [132]: model=SVC()
```

```
In [133]: #Train the model
          model.fit(X_train,y_train)
```

```
Out[133]: SVC()
```

Check the score of the model

```
In [134]: model.score(X_train,y_train)
```

```
Out[134]: 0.7690875232774674
```

Make prediction with X_test

```
In [135]: y_pred=model.predict(X_test)
```

Check the accuracy of our model

```
In [138]: from sklearn import metrics
```

```
In [140]: metrics.accuracy_score(y_test,y_pred)
```

```
Out[140]: 0.7575757575757576
```

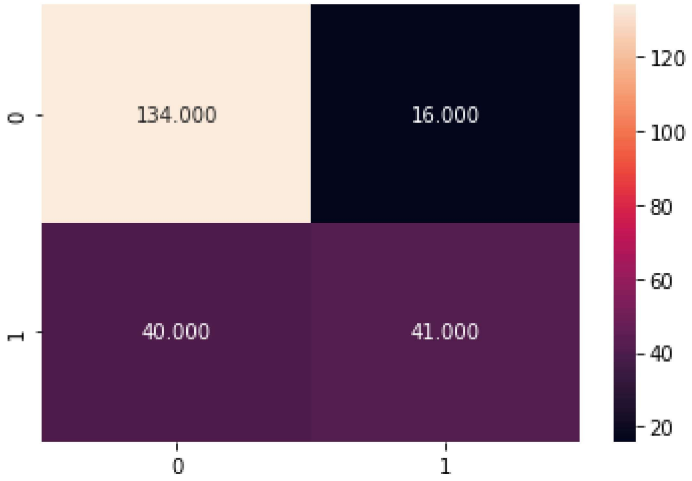
Create a confusion matrix

```
In [141]: metrics.confusion_matrix(y_test,y_pred)
```

```
Out[141]: array([[134,  16],
                 [ 40,  41]], dtype=int64)
```

Plot confusion matrix on heatmap

```
In [144]: sns.heatmap(metrics.confusion_matrix(y_test,y_pred),annot=True,fmt=".3f")
          plt.show()
```



Create a classification report

In [145]:

```
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.89      0.83       150
           1       0.72      0.51      0.59        81

    accuracy                           0.76       231
   macro avg       0.74      0.70      0.71       231
weighted avg       0.75      0.76      0.75       231
```