In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

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

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
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.shape
```

Out[5]:

```
(768, 9)
```

In [6]:

```
df.describe()
```

Out[6]:

	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

In [7]:

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

Out[7]:

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

In [8]:

```
from sklearn.preprocessing import MinMaxScaler
scalar = MinMaxScaler()
```

In [9]:

```
df.drop(['SkinThickness'],inplace = True, axis = 1)
```

In [10]:

```
df
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	0	33.6	0.627	50	1
1	1	85	66	0	26.6	0.351	31	0
2	8	183	64	0	23.3	0.672	32	1
3	1	89	66	94	28.1	0.167	21	0
4	0	137	40	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...
763	10	101	76	180	32.9	0.171	63	0
764	2	122	70	0	36.8	0.340	27	0
765	5	121	72	112	26.2	0.245	30	0
766	1	126	60	0	30.1	0.349	47	1
767	1	93	70	0	30.4	0.315	23	0

768 rows × 8 columns

In [11]:

```
def remove_outliers_iqr(df):
    # Calculate the first quartile (Q1) and third quartile (Q3)
    q1 = np.percentile(df, 25)
    q3 = np.percentile(df, 75)

    # Calculate the interquartile range (IQR)
    iqr = q3 - q1

    # Define the lower and upper bounds for outliers
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    # Remove outliers
    df = [x for x in df if lower_bound <= x <= upper_bound]

    return df
```

In [12]:

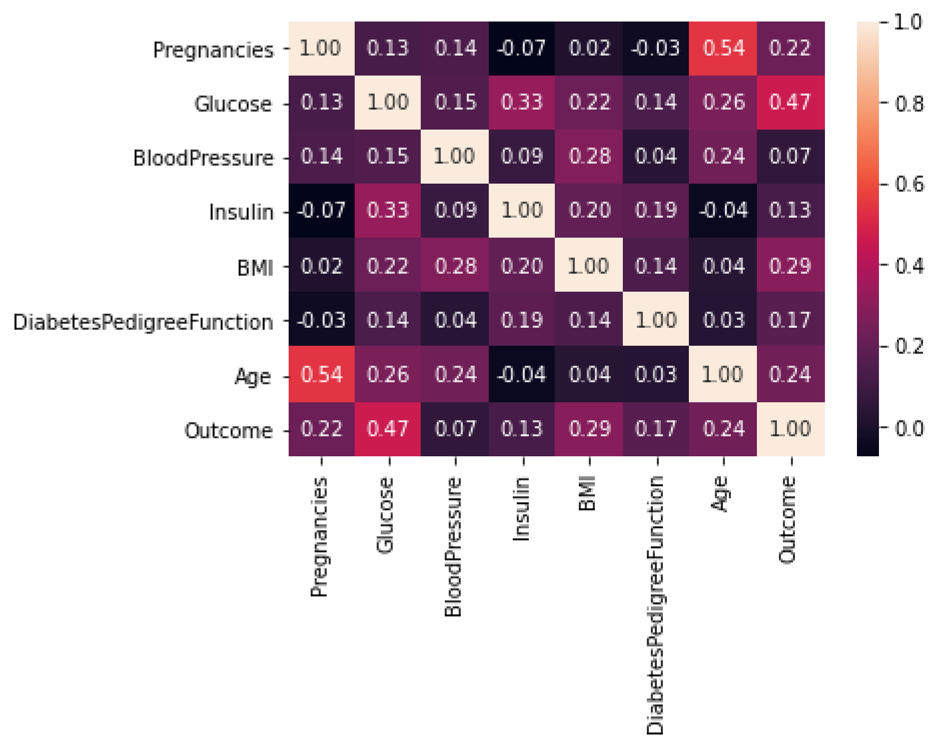
```
import seaborn as sns
```

In [13]:

```
sns.heatmap(df.corr(), annot=True, fmt='.2f')
```

Out[13]:

```
<AxesSubplot:>
```



In [14]:

```
from sklearn.model_selection import train_test_split
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

In [15]:

```
scalar.fit(X)
standardised_data = scalar.transform(X)
```

In [16]:

```
X = standardised_data
y = df['Outcome']
```

In [17]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=0
```

In [18]:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=20, criterion='entropy', random_state=42)
```

In [19]:

```
rf.fit(X_train, y_train)
```

Out[19]:

```
RandomForestClassifier(criterion='entropy', n_estimators=20, random_state=42)
```

In [20]:

```
y_pred = rf.predict(X_test)
```

In [21]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[21]:

```
0.8441558441558441
```

In [22]:

```
'''RANDOM FOREST.'''
```

Out[22]:

```
'RANDOM FOREST.'
```