In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv('diabetes.csv')
df
```

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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | 0.171 | 63 | 0 |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | 0.340 | 27 | 0 |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | 0.245 | 30 | 0 |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | 0.349 | 47 | 1 |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | 0.315 | 23 | 0 |

768 rows × 9 columns

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 [4]:
```
df.head()
```

Out[4]:

| | 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 [5]:
```
df.Outcome.value_counts()
```

Out[5]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [6]:

```
df1 = df[df['Outcome'] == 0].iloc[:300,:]
df2 = df[df['Outcome'] == 1]
```

In [7]:

```
df3 = pd.concat([df1,df2],axis=0)
```

In [8]:

```
df3.Outcome.value_counts()
```

Out[8]:

```
0    300
1    268
Name: Outcome, dtype: int64
```

In [9]:

```
x,y = df.iloc[:,:-1],df.Outcome
```

In [10]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [11]:

```
mn = MinMaxScaler()
x = mn.fit_transform(x)
x
```

Out[11]:

```
array([[0.35294118, 0.74371859, 0.59016393, ..., 0.50074516, 0.23441503,
        0.48333333],
       [0.05882353, 0.42713568, 0.54098361, ..., 0.39642325, 0.11656704,
        0.16666667],
       [0.47058824, 0.91959799, 0.52459016, ..., 0.34724292, 0.25362938,
        0.18333333],
       ...,
       [0.29411765, 0.6080402 , 0.59016393, ..., 0.390462  , 0.07130658,
        0.15      ],
       [0.05882353, 0.63316583, 0.49180328, ..., 0.4485842 , 0.11571307,
        0.43333333],
       [0.05882353, 0.46733668, 0.57377049, ..., 0.45305514, 0.10119556,
        0.03333333]])
```

In [12]:
```
from sklearn.model_selection import train_test_split
```

In [13]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=2)
```

In [14]:
```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassif
from xgboost import XGBClassifier
```

In [15]:
```
from sklearn.metrics import classification_report,f1_score
```

In [16]:
```
abc = AdaBoostClassifier()
abc.fit(xtrain,ytrain)
ypred = abc.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.78      0.83      0.81       109
           1       0.53      0.44      0.48        45

    accuracy                           0.72       154
   macro avg       0.66      0.64      0.65       154
weighted avg       0.71      0.72      0.71       154

[0.80888889 0.48192771]
```

In [17]:

```
gbc = GradientBoostingClassifier()
gbc.fit(xtrain,ytrain)
ypred = gbc.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.81      0.81      0.81       109
           1       0.54      0.56      0.55        45

    accuracy                           0.73       154
   macro avg       0.68      0.68      0.68       154
weighted avg       0.74      0.73      0.73       154

[0.81105991 0.54945055]
```

In [18]:

```
xgb = XGBClassifier()
xgb.fit(xtrain,ytrain)
ypred = xgb.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.82      0.79      0.80       109
           1       0.53      0.58      0.55        45

    accuracy                           0.73       154
   macro avg       0.67      0.68      0.68       154
weighted avg       0.73      0.73      0.73       154

[0.80373832 0.55319149]
```

In [19]:

```
rf = RandomForestClassifier()
rf.fit(xtrain, ytrain)
ypred = rf.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84       109
           1       0.61      0.56      0.58        45

    accuracy                           0.77       154
   macro avg       0.72      0.70      0.71       154
weighted avg       0.76      0.77      0.76       154

[0.83783784 0.58139535]
```

## Balancing target column

In [20]:

```
df1 = df[df['Outcome'] == 0].iloc[:300,:]
df2 = df[df['Outcome'] == 1]

df3 = pd.concat([df1,df2],axis=0)

df3.Outcome.value_counts()
```

Out[20]:

```
0    300
1    268
Name: Outcome, dtype: int64
```

In [21]:

```
x,y = df3.iloc[:,:-1],df3.Outcome
```

In [22]:

```
mn = MinMaxScaler()
x = mn.fit_transform(x)
x
```

Out[22]:

```
array([[0.05882353, 0.42713568, 0.54098361, ..., 0.39642325, 0.11656704,
        0.16666667],
       [0.05882353, 0.44723618, 0.54098361, ..., 0.41877794, 0.03800171,
        0.        ],
       [0.29411765, 0.58291457, 0.60655738, ..., 0.38152012, 0.05251921,
        0.15      ],
       ...,
       [0.35294118, 0.95477387, 0.75409836, ..., 0.5290611 , 0.0853971 ,
        0.75      ],
       [0.52941176, 0.85427136, 0.60655738, ..., 0.6557377 , 0.13877028,
        0.36666667],
       [0.05882353, 0.63316583, 0.49180328, ..., 0.4485842 , 0.11571307,
        0.43333333]])
```

In [23]:

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.20,random_state=2)
```

In [24]:

```
abc = AdaBoostClassifier()
abc.fit(xtrain,ytrain)
ypred = abc.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.80      0.70      0.75        61
           1       0.70      0.79      0.74        53

    accuracy                           0.75       114
   macro avg       0.75      0.75      0.75       114
weighted avg       0.75      0.75      0.75       114

[0.74782609 0.74336283]
```

In [25]:

```
gbc = GradientBoostingClassifier()
gbc.fit(xtrain,ytrain)
ypred = gbc.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.77      0.67      0.72        61
           1       0.67      0.77      0.72        53

    accuracy                           0.72       114
   macro avg       0.72      0.72      0.72       114
weighted avg       0.73      0.72      0.72       114

[0.71929825 0.71929825]
```

In [26]:
```
xgb = XGBClassifier()
xgb.fit(xtrain,ytrain)
ypred = xgb.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.71      0.66      0.68        61
           1       0.64      0.70      0.67        53

    accuracy                           0.68       114
   macro avg       0.68      0.68      0.68       114
weighted avg       0.68      0.68      0.68       114

[0.68376068 0.66666667]
```

In [27]:
```
rf = RandomForestClassifier()
rf.fit(xtrain, ytrain)
ypred = rf.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.75      0.69      0.72        61
           1       0.67      0.74      0.70        53

    accuracy                           0.71       114
   macro avg       0.71      0.71      0.71       114
weighted avg       0.71      0.71      0.71       114

[0.71794872 0.7027027 ]
```

# Adaboost is performing better

In [28]:

```
abc = AdaBoostClassifier()
abc.fit(xtrain,ytrain)
ypred = abc.predict(xtest)

print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.80      0.70      0.75        61
           1       0.70      0.79      0.74        53

    accuracy                           0.75       114
   macro avg       0.75      0.75      0.75       114
weighted avg       0.75      0.75      0.75       114

[0.74782609 0.74336283]
```

In [29]:

```
print(abc.score(xtrain, ytrain))
print(abc.score(xtest, ytest))
```

```
0.8480176211453745
0.7456140350877193
```

In [30]:

```
# for i in range(5000,10000):
#     xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.25,random_state=i)
#     abc = AdaBoostClassifier()
#     abc.fit(xtrain,ytrain)
#     ypred = abc.predict(xtest)
#     if abc.score(xtrain, ytrain) > 0.82 and abc.score(xtest, ytest) > 0.81:
#         print('Random State:', i)
#         print(abc.score(xtrain, ytrain))
#         print(abc.score(xtest, ytest))
#         print()
```

In [31]:

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.25,random_state=7998)
abc = AdaBoostClassifier()
abc.fit(xtrain,ytrain)
ypred = abc.predict(xtest)

print(abc.score(xtrain, ytrain))
print(abc.score(xtest, ytest))


print(classification_report(ytest,ypred))
print(f1_score(ytest,ypred,average=None,labels=[0,1]))
```

```
0.823943661971831
0.823943661971831
              precision    recall  f1-score   support

           0       0.83      0.84      0.83        75
           1       0.82      0.81      0.81        67

    accuracy                           0.82       142
   macro avg       0.82      0.82      0.82       142
weighted avg       0.82      0.82      0.82       142

[0.83443709 0.81203008]
```

In [ ]:

In [ ]:

In [ ]: