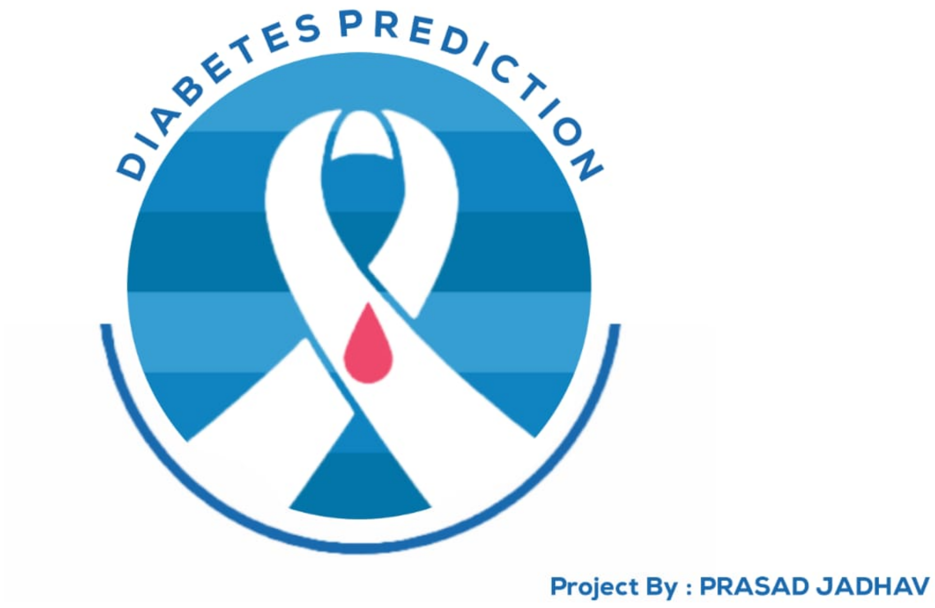


In [33]:
```python
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from imblearn.over_sampling import SMOTE

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import svm

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

In [2]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```python
dataset = pd.read_csv('diabetes.csv')
dataset.shape
```

Out[3]:
```
(768, 9)
```

In [4]:
```python
dataset.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [5]:

```
dataset.tail()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

In [6]:

```
dataset.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]:

```
dataset.describe()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedig
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	

In [8]:
```
dataset.corr()
```

Out[8]:

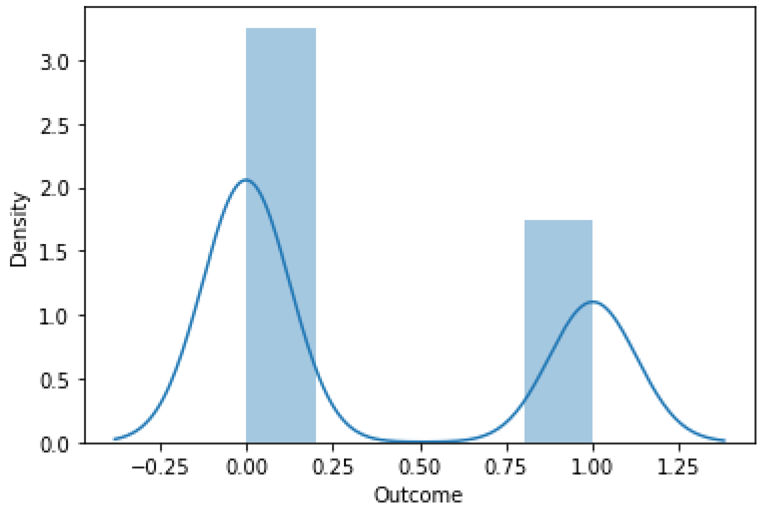
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	D
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	
SkinThickness	-0.081672	0.057328	0.207371	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	

In [9]:
```
dataset.cov()
```

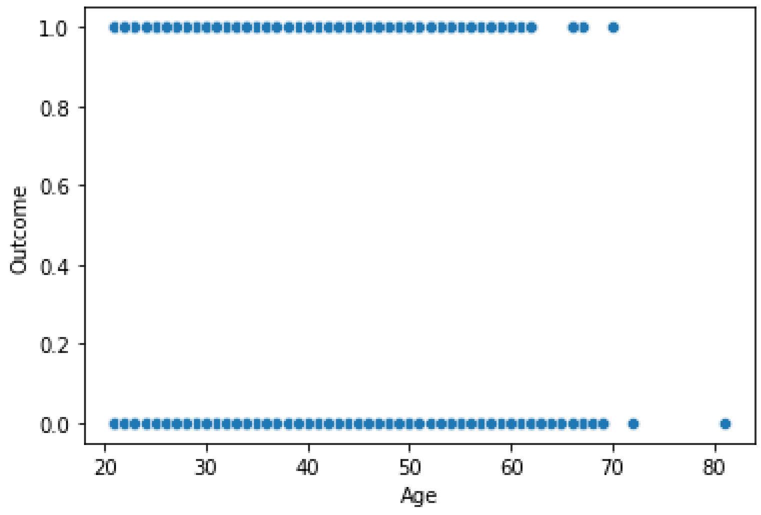
Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	11.354056	13.947131	9.214538	-4.390041	-28.555231	0.
Glucose	13.947131	1022.248314	94.430956	29.239183	1220.935799	55.
BloodPressure	9.214538	94.430956	374.647271	64.029396	198.378412	43.
SkinThickness	-4.390041	29.239183	64.029396	254.473245	802.979941	49.
Insulin	-28.555231	1220.935799	198.378412	802.979941	13281.180078	179.
BMI	0.469774	55.726987	43.004695	49.373869	179.775172	62.
DiabetesPedigreeFunction	-0.037426	1.454875	0.264638	0.972136	7.066681	0.
Age	21.570620	99.082805	54.523453	-21.381023	-57.143290	3.
Outcome	0.356618	7.115079	0.600697	0.568747	7.175671	1.

In [10]:
```
sns.distplot(dataset['Outcome'])
plt.show()
```



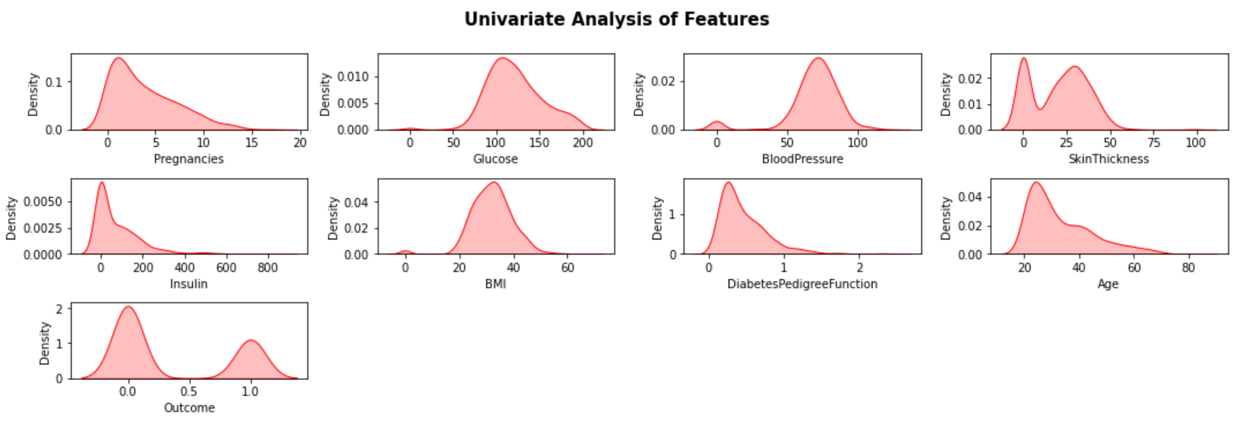
```
In [11]: sns.scatterplot(dataset['Age'],dataset['Outcome'])
         plt.show()
```



```
In [12]: features = [feature for feature in dataset.columns]
```

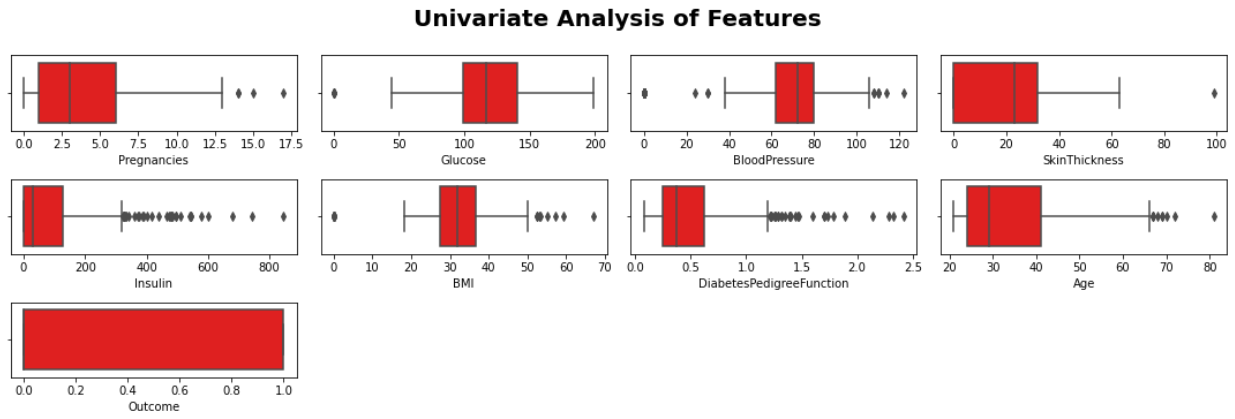
```
In [13]: plt.figure(figsize=(15,15))
         plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=15,y

         for i in range(0,len(features)):
             plt.subplot(10,4,i+1)
             sns.kdeplot(x=dataset[features[i]],shade=True,color='red')
             plt.tight_layout()
```



In [14]:
```
plt.figure(figsize = (15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=20,y=

for i in range(0,len(features)):
    plt.subplot(10,4,i+1)
    sns.boxplot(data=dataset,x=features[i],color='red')
    plt.xlabel(features[i])
    plt.tight_layout()
```



In [15]:
```
dataset['Outcome'].value_counts()
```

Out[15]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [16]:
```
X = dataset.drop('Outcome',axis=1)
y = dataset['Outcome']
```

In [19]:
```
X_res,y_res = SMOTE().fit_resample(X,y)
```

In [20]:
```
X_train,X_test,y_train,y_test = train_test_split(X_res,y_res,test_size=0.20,ra
```

In [21]:
```
st = StandardScaler()
X_train = st.fit_transform(X_train)
X_test = st.fit_transform(X_test)
```

In [22]:
```
model_df = {}

def model_val(model,X,y):
    X_train,X_test,y_train,y_test = train_test_split(X_res,y_res,test_size=0.2
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    print(f'{model} Accuracy is {accuracy_score(y_test,y_pred)}')

    score = cross_val_score(model,X,y,cv=5,n_jobs=-1)
    print(f'{model} Average cross val score is {np.mean(score)}')
    model_df[model] = round(np.mean(score)*100,2)
```

In [23]:
```
model = LogisticRegression()
model_val(model,X,y)
```

```
LogisticRegression() Accuracy is 0.71
LogisticRegression() Average cross val score is 0.7695696460402341
```

In [24]:
```
model = DecisionTreeClassifier()
model_val(model,X,y)
```

```
DecisionTreeClassifier() Accuracy is 0.7
DecisionTreeClassifier() Average cross val score is 0.7123673711909005
```

In [25]:
```
model = RandomForestClassifier()
model_val(model,X,y)
```

```
RandomForestClassifier() Accuracy is 0.78
RandomForestClassifier() Average cross val score is 0.7683133859604447
```

In [26]:
```
model = GradientBoostingClassifier()
model_val(model,X,y)
```

```
GradientBoostingClassifier() Accuracy is 0.77
GradientBoostingClassifier() Average cross val score is 0.7591715474068416
```

In [27]:
```
model_df
```

Out[27]:
```
{LogisticRegression(): 76.96,
 DecisionTreeClassifier(): 71.24,
 RandomForestClassifier(): 76.83,
 GradientBoostingClassifier(): 75.92}
```

In [28]:
```
C = 1.0
svm = svm.SVC(kernel='linear',C=C)
svm.fit(X_train,y_train)
```

Out[28]:
```
▾ SVC
SVC(kernel='linear')
```

In [29]:
```
y_pred = svm.predict(X_test)
```

In [30]:
```
accuracy_score(y_test,y_pred)
```

Out[30]:
```
0.705
```

In [31]:
```
parameters = [

    {'kernel':['linear'],'C': [1,10]},
    {'kernel':['poly'],'C':[1, 10]},
    {'kernel':['rbf'],'gamma': [1e-3,1e-4],'C': [1,10]}
]
```

In [34]:
```
gr  = GridSearchCV(svm.SVC(),parameters,n_jobs=-1)
gr.fit(X_train,y_train)
```

Out[34]:
```
▸ GridSearchCV
▸estimator: SVC
      ▸ SVC
```

In [36]:
```
y_pred_ = gr.predict(X_test)
```

In [37]:
```
accuracy_score(y_test,y_pred_)
```

Out[37]:
```
0.705
```

In [38]:
```
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
```

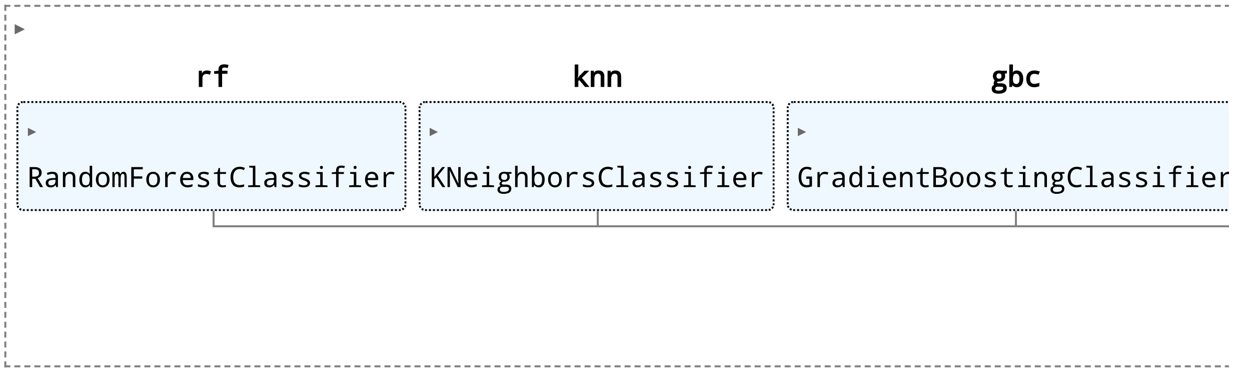
In [41]:
```
estimators = [
             ('rf', RandomForestClassifier(n_estimators = 10, random_state = 4
             ('knn', KNeighborsClassifier(n_neighbors = 10)),
             ('gbc', GradientBoostingClassifier()),
             ('lr', LogisticRegression()),
             ('ccv', CalibratedClassifierCV()),
             ('mlp', MLPClassifier()),
             ('dt', DecisionTreeClassifier()),
             ('lda', LinearDiscriminantAnalysis()),
             ('gnb', GaussianNB()),
             ('adb', AdaBoostClassifier()),
             ('etc', ExtraTreesClassifier()),
             ('sgd', SGDClassifier()),
             ('svm', SVC()),
             ('xgb', XGBClassifier(n_estimators= 10, random_state = 42))
             ]
```

```
In [42]: from sklearn.ensemble import StackingClassifier

         st = StackingClassifier(
                                  estimators = estimators,
                                  final_estimator = LogisticRegression(),
                                  cv = 10,
                                  stack_method='predict',
                                  n_jobs=-1
                                  )
```

```
In [43]: st.fit(X_train,y_train)
```

Out[43]:



```
In [44]: y_pred = st.predict(X_test)
```

```
In [45]: accuracy_score(y_test,y_pred)

Out[45]: 0.785
```

```
In [46]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import BaggingClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.ensemble import ExtraTreesClassifier
         from sklearn.model_selection import GridSearchCV
         from xgboost import XGBClassifier
```

```
In [49]: dt_parameters = {
          'criterion' : ['gini', 'entropy', 'log_loss'],
          'splitter' : ['best', 'random'],
          'max_depth' : range(1,10,1),
          'min_samples_split' : range(2,10,2),
          'min_samples_leaf' : range(1,5,1),
          'max_features' : ['auto', 'sqrt', 'log2']
         }

         bg_parameters = {
          'n_estimators' : [5, 10, 15],
          'max_samples' : range(2, 10, 1),
          'max_features' : range(2, 10, 3)
         }

         rf_parameters = {
          'criterion' : ['gini', 'entropy', 'log_loss'],
          'max_depth' : range(1, 10, 1),
          'min_samples_split' : range(2, 10, 2),
          'min_samples_leaf' : range(1, 10, 1),
         }

         et_parameters = {
          'n_estimators' : [10,20,30],
          'criterion' : ['gini', 'entropy', 'log_loss'],
          'max_depth' : range(2,10,1),
          'min_samples_split' : range(2,10,2),
          'min_samples_leaf' : range(1,5,1),
          'max_features' : ['sqrt', 'log2']
         }

         xgb_parameters = {
          'max_depth' : [2,4,6],
          'min_child_weight' : [2,4,6],
          'gamma' : [i/10 for i in range(4)]
         }
```

In [50]:
```
hyper_1 = GridSearchCV(estimator = DecisionTreeClassifier(), param_grid = dt_pa
hyper_2 = GridSearchCV(estimator = BaggingClassifier(), param_grid = bg_parame
hyper_3 = GridSearchCV(estimator = RandomForestClassifier(), param_grid = rf_pa
hyper_4 = GridSearchCV(estimator = ExtraTreesClassifier(), param_grid = et_para
hyper_5 = GridSearchCV(estimator = XGBClassifier(), param_grid = xgb_parameter
```

In [51]:
```
hyper_1.fit(X_train,y_train)
hyper_2.fit(X_train,y_train)
hyper_3.fit(X_train,y_train)
hyper_4.fit(X_train,y_train)
hyper_5.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 2592 candidates, totalling 7776 fits
Fitting 3 folds for each of 72 candidates, totalling 216 fits
Fitting 3 folds for each of 972 candidates, totalling 2916 fits
Fitting 3 folds for each of 2304 candidates, totalling 6912 fits
Fitting 3 folds for each of 36 candidates, totalling 108 fits
```

Out[51]:

▸ **GridSearchCV**

▸ **estimator: XGBClassifier**

▸ XGBClassifier

In [52]:
```
h_pred_1 = hyper_1.predict(X_test)
h_pred_2 = hyper_2.predict(X_test)
h_pred_3 = hyper_3.predict(X_test)
h_pred_4 = hyper_4.predict(X_test)
h_pred_5 = hyper_5.predict(X_test)
```

In [53]:
```
print(accuracy_score(y_test,h_pred_1))
print(accuracy_score(y_test,h_pred_2))
print(accuracy_score(y_test,h_pred_3))
print(accuracy_score(y_test,h_pred_4))
print(accuracy_score(y_test,h_pred_5))
```

```
0.705
0.67
0.795
0.795
0.78
```

In [54]:
```python
import pickle
import joblib
```

In [56]:
```python
pickle.dump(hyper_3,open('diabetes_prediction.pkl','wb'))
```

In [57]:
```python
model = pickle.load(open('diabetes_prediction.pkl','rb'))
```

In [58]:
```python
dataset.head()
```

Out[58]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [60]:
```python
new_df = pd.DataFrame({
    'Pregnancies':6,
    'Glucose':148,
    'BloodPressure':72,
    'SkinThickness':35,
    'Insulin':0,
    'BMI':33.6,
    'DiabetesPedigreeFunction':0.627,
    'Age':50,
},index=[0])
```

In [61]:
```python
model.predict(new_df)
```

Out[61]:
```
array([1], dtype=int64)
```

In [63]:
```python
p = hyper_3.predict(new_df)

prob = hyper_3.predict_proba(new_df)
if p == 1:
    print('Diabetes!')
    print(f'You will be Diabetes! with Probability of {prob[0][1]:.2f}')
else:
    print('Not-Diabetes!')
```

```
Diabetes!
You will be Diabetes! with Probability of 0.65
```