In [171]:
```
%cd C:\Users\manoj\Downloads
```

```
C:\Users\manoj\Downloads
```

In [172]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats
```

In [173]:
```
df=pd.read_csv('diabetes.csv')
```

In [212]:
```
df.head()
```

Out[212]:

	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 [174]:
```
X=np.array(df.iloc[:,:8])
```

In [175]:
```
y=np.array(df.iloc[:,8])
```

In [176]:
```
from sklearn.linear_model import LogisticRegression
from numpy import mean
from numpy import std
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

In [177]:
```
model=LogisticRegression(max_iter=500)
```

In [178]:
```
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
```

In [179]:
```
space=dict()
space['solver']=['newton-cg','lbfgs','liblinear']
space['penalty']=['none','l1','l2','elasticnet']
```

In [180]:
```
search=RandomizedSearchCV(model,space,n_iter=500,scoring='accuracy',n_jobs=-1,cv=cv,random_state=1)
```

In [181]:
```
result=search.fit(X,y)
```

```
C:\Users\manoj\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:278: UserWarning: The total space of parameters 12 is smaller than n_iter=500. Running 12 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

In [182]:
```
print('Best Score: %s' % result.best_score_)
print('Best Hyperparameters: %s' % result.best_params_)
```

```
Best Score: 0.7760025062656641
Best Hyperparameters: {'solver': 'newton-cg', 'penalty': 'l2'}
```

In [183]:
```
model=LogisticRegression(penalty='l2',solver='newton-cg',max_iter=500)
```

In [184]:
```
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
```

In [185]:
```
scores=cross_val_score(model,X,y,scoring='accuracy',cv=cv,n_jobs=-1)
```

In [186]:
```
print('Accuracy:%3f (%.3f)' %(mean(scores),std(scores)))
```

```
Accuracy:0.776003 (0.047)
```

In [187]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

In [188]:
```
from sklearn.linear_model import LogisticRegression
```

In [189]:
```
model= LogisticRegression(max_iter=500)
```

In [190]:
```
model.fit(X_train,y_train)
```

Out[190]:
```
LogisticRegression(max_iter=500)
```

In [191]:
```
y_pred=model.predict(X_test)
```

In [192]:
```
print(y_pred)
```

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

In [193]:
```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[193]:
```
array([[89, 10],
       [24, 31]], dtype=int64)
```

In [194]:
```
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.90      0.84        99
           1       0.76      0.56      0.65        55

    accuracy                           0.78       154
   macro avg       0.77      0.73      0.74       154
weighted avg       0.78      0.78      0.77       154
```

In [195]:
```
#gridsearch
search=GridSearchCV(model,space,scoring='accuracy',n_jobs=-1,cv=cv)
```

In [196]:
```
result=search.fit(X,y)
```

In [197]:
```
print('Best Score: %s' % result.best_score_)
print('Best Hyperparameters: %s' % result.best_params_)
```

```
Best Score: 0.7760025062656641
Best Hyperparameters: {'penalty': 'l2', 'solver': 'newton-cg'}
```

In [198]:
```
#try same for random forest classifier
```

In [199]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [200]:
```
model=RandomForestClassifier(n_estimators=100)
```

In [201]:
```
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
```

In [202]:
```
space=dict()
space['criterion']=['gini','entrophy']
space['max_features']=['auto','sqrt','log2']
```

In [203]:
```
search=RandomizedSearchCV(model,space,n_iter=500,scoring='accuracy',n_jobs=-1,cv=cv,random_state=1)
```

In [204]:
```
result=search.fit(X,y)
```

```
C:\Users\manoj\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:278: UserWarning: The total space of
parameters 6 is smaller than n_iter=500. Running 6 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

In [205]:
```
print('Best Score: %s' % result.best_score_)
print('Best Hyperparameters: %s' % result.best_params_)
```

```
Best Score: 0.7642914103440419
Best Hyperparameters: {'max_features': 'log2', 'criterion': 'gini'}
```

In [206]:
```
model=RandomForestClassifier(max_features='log2', criterion= 'gini',n_estimators=100)
```

In [207]:
```
model.fit(X_train,y_train)
```

Out[207]:
```
RandomForestClassifier(max_features='log2')
```

In [208]:
```
y_pred=model.predict(X_test)
```

In [209]:
```
print(y_pred)
```

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

In [210]:
```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[210]:
```
array([[88, 11],
       [18, 37]], dtype=int64)
```

In [211]:
```
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        99
           1       0.77      0.67      0.72        55

    accuracy                           0.81       154
   macro avg       0.80      0.78      0.79       154
weighted avg       0.81      0.81      0.81       154
```

In [ ]:
```

```