In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [2]:
```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
```

In [30]:
```python
df = pd.read_csv("bank.csv")
df.head()
df.shape
```

Out[30]:
```
(11162, 17)
```

In [31]:
```python
df.head()
```

Out[31]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	0	1	1	0	2343	1	0	2	5	8	1042	1	-1	0	3	1
1	56	0	1	1	0	45	0	0	2	5	8	1467	1	-1	0	3	1
2	41	9	1	1	0	1270	1	0	2	5	8	1389	1	-1	0	3	1
3	55	7	1	1	0	2476	1	0	2	5	8	579	1	-1	0	3	1
4	54	0	1	2	0	184	0	0	2	5	8	673	2	-1	0	3	1

In [5]:
```python
df.deposit.value_counts()
```

Out[5]:
```
0    5873
1    5289
Name: deposit, dtype: int64
```

In [6]:
```python
x = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [7]:
```python
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.3, random_state=1)
```

In [8]:
```python
def mymodel(model):
    model.fit(xtrain,ytrain)
    ypred=model.predict(xtest)
    print(classification_report(ytest,ypred))
    return model
```

In [9]:
```python
dt=DecisionTreeClassifier()
```

In [10]:
```python
mymodel(dt)
```

```
              precision    recall  f1-score   support

           0       0.79      0.80      0.80      1760
           1       0.78      0.77      0.77      1589

    accuracy                           0.79      3349
   macro avg       0.78      0.78      0.78      3349
weighted avg       0.78      0.79      0.78      3349
```

Out[10]:
```
DecisionTreeClassifier()
```

In [11]:
```python
dt.score(xtrain,ytrain)
```

Out[11]:
```
1.0
```

In [12]:
```python
dt.score(xtest,ytest)
```

Out[12]:
```
0.78501045088086
```

In [13]:
```python
dt.feature_importances_
```

Out[13]:
```
array([0.08045239, 0.03509133, 0.01761428, 0.01205608, 0.00045636,
       0.08208469, 0.04261978, 0.00669448, 0.06214084, 0.07658374,
       0.09705957, 0.35562815, 0.02089599, 0.06073951, 0.0131865 ,
       0.03669631])
```

hyperparameter tunning

In [14]:
```python
dt1=DecisionTreeClassifier(max_depth=10)
mymodel(dt1)
```

```
              precision    recall  f1-score   support

           0       0.83      0.83      0.83      1760
           1       0.81      0.81      0.81      1589

    accuracy                           0.82      3349
   macro avg       0.82      0.82      0.82      3349
weighted avg       0.82      0.82      0.82      3349
```

Out[14]:
```
DecisionTreeClassifier(max_depth=10)
```

In [15]:
```python
for i in range(1,50):
    dt2=DecisionTreeClassifier(max_depth=i)
    dt2.fit(xtrain,ytrain)
    ypred=dt2.predict(xtest)
    print(f"{i}: {accuracy_score(ytest,ypred)}")
```

```
1: 0.7121528814571514
2: 0.7121528814571514
3: 0.771275007464915
4: 0.78501045088086
5: 0.7993430874888027
6: 0.8005374738727978
7: 0.814272917288743
8: 0.8181546730367274
9: 0.8118841445207524
10: 0.8196476560167214
11: 0.8166616900567334
12: 0.8139743206927441
13: 0.8074051955807704
14: 0.8008360704687967
15: 0.798745894296805
16: 0.7894893998208421
17: 0.794266945356823
18: 0.78501045088086
19: 0.7936697521648253
20: 0.7897879964168408
21: 0.7868020304568528
22: 0.7882950134368468
23: 0.7868020304568528
24: 0.7856076440728575
25: 0.7891908032248433
26: 0.7885936100328457
27: 0.7903851896088384
28: 0.7885936100328457
29: 0.7871006270528516
30: 0.7882950134368468
31: 0.7873992236488504
32: 0.7871006270528516
33: 0.7832188713048671
34: 0.7838160644968647
35: 0.7862048372648551
36: 0.78501045088086
37: 0.7885936100328457
38: 0.7856076440728575
39: 0.7918781725888325
40: 0.787996416840848
41: 0.7888922066288444
42: 0.7856076440728575
43: 0.7936697521648253
44: 0.7832188713048671
45: 0.7891908032248433
46: 0.7853090474768588
47: 0.7841146610928635
48: 0.7817258883248731
49: 0.7921767691848313
```

In [16]:
```python
dt3=DecisionTreeClassifier(max_depth=11)
mymodel(dt3)
```

```
              precision    recall  f1-score   support

           0       0.83      0.82      0.83      1760
           1       0.81      0.81      0.81      1589

    accuracy                           0.82      3349
   macro avg       0.82      0.82      0.82      3349
weighted avg       0.82      0.82      0.82      3349
```

Out[16]:
```
DecisionTreeClassifier(max_depth=11)
```

In [17]:
```python
dt4=DecisionTreeClassifier(min_samples_leaf=10)  #The minimum number of samples required to be at a leaf node.
mymodel(dt4)
```

```
              precision    recall  f1-score   support

           0       0.82      0.83      0.82      1760
           1       0.81      0.80      0.80      1589

    accuracy                           0.81      3349
   macro avg       0.81      0.81      0.81      3349
weighted avg       0.81      0.81      0.81      3349
```

Out[17]:
```
DecisionTreeClassifier(min_samples_leaf=10)
```

In [18]:
```python
for i in range(1,75):
    dt2=DecisionTreeClassifier(min_samples_leaf=i)
    dt2.fit(xtrain,ytrain)
    ypred=dt2.predict(xtest)
    print(f"{i}: {accuracy_score(ytest,ypred)}")
```

```
1: 0.7906837862048373
2: 0.7715736040609137
3: 0.7897879964168408
4: 0.7888922066288444
5: 0.7915795759928337
6: 0.7948641385488205
7: 0.8083009853687668
8: 0.8133771275007465
9: 0.8097939683487608
10: 0.8145715138847417
11: 0.8127799343087488
12: 0.8130785309047477
13: 0.8127799343087488
14: 0.8145715138847417
15: 0.8166616900567334
16: 0.820244849208719
17: 0.8214392355927143
18: 0.818751866228725
19: 0.8217378321887131
20: 0.8274111675126904
21: 0.8262167811286951
22: 0.8274111675126904
23: 0.8283069573006868
24: 0.8259181845326963
25: 0.8268139743206927
26: 0.8262167811286951
27: 0.8262167811286951
28: 0.8256195879366975
29: 0.8274111675126904
30: 0.8238280083607047
31: 0.8262167811286951
32: 0.8262167811286951
33: 0.8250223947447
34: 0.826515377724694
35: 0.8253209913406987
36: 0.8253209913406987
37: 0.8259181845326963
38: 0.8256195879366975
39: 0.828008360704688
40: 0.8262167811286951
41: 0.829501343684682
42: 0.8286055538966856
43: 0.8289041504926844
44: 0.8253209913406987
45: 0.8253209913406987
46: 0.8253209913406987
47: 0.8253209913406987
48: 0.8253209913406987
49: 0.8253209913406987
50: 0.8250223947447
51: 0.8277097641086891
52: 0.8277097641086891
53: 0.8247237981487011
54: 0.8235294117647058
55: 0.8244252015527023
56: 0.8244252015527023
57: 0.8238280083607047
58: 0.8214392355927143
59: 0.8214392355927143
60: 0.8214392355927143
61: 0.8214392355927143
62: 0.8214392355927143
63: 0.8214392355927143
64: 0.8214392355927143
65: 0.8217378321887131
66: 0.8160644968647357
67: 0.8163630934607345
68: 0.8196476560167214
69: 0.8184532696327261
70: 0.8184532696327261
71: 0.8175574798447298
72: 0.8175574798447298
73: 0.8178560764407286
74: 0.8178560764407286
```

In [19]:
```python
dt5=DecisionTreeClassifier(min_samples_leaf=41)
mymodel(dt5)
```

```
              precision    recall  f1-score   support

           0       0.86      0.81      0.83      1760
           1       0.80      0.85      0.83      1589

    accuracy                           0.83      3349
   macro avg       0.83      0.83      0.83      3349
weighted avg       0.83      0.83      0.83      3349
```

Out[19]:
```
DecisionTreeClassifier(min_samples_leaf=41)
```

In [20]:
```python
dt6=DecisionTreeClassifier(criterion="gini",min_samples_leaf=41)
mymodel(dt6)
```

```
              precision    recall  f1-score   support

           0       0.86      0.81      0.83      1760
           1       0.80      0.85      0.83      1589

    accuracy                           0.83      3349
   macro avg       0.83      0.83      0.83      3349
weighted avg       0.83      0.83      0.83      3349
```

Out[20]:
```
DecisionTreeClassifier(min_samples_leaf=41)
```

In [32]:
```python
dt7=DecisionTreeClassifier(criterion="entropy",min_samples_leaf=41)
mymodel(dt7)
```

```
              precision    recall  f1-score   support

           0       0.84      0.82      0.83      1760
           1       0.80      0.83      0.82      1589

    accuracy                           0.82      3349
   macro avg       0.82      0.82      0.82      3349
weighted avg       0.82      0.82      0.82      3349
```

Out[32]:
```
DecisionTreeClassifier(criterion='entropy', min_samples_leaf=41)
```

In [22]:
```python
dt8=DecisionTreeClassifier(criterion="gini",max_depth=11)
mymodel(dt8)
```

```
              precision    recall  f1-score   support

           0       0.83      0.82      0.83      1760
           1       0.81      0.81      0.81      1589

    accuracy                           0.82      3349
   macro avg       0.82      0.82      0.82      3349
weighted avg       0.82      0.82      0.82      3349
```

Out[22]:
```
DecisionTreeClassifier(max_depth=11)
```

In [23]:
```python
dt9=DecisionTreeClassifier(criterion="entropy",max_depth=11)
mymodel(dt9)
```

```
              precision    recall  f1-score   support

           0       0.81      0.83      0.82      1760
           1       0.81      0.78      0.79      1589

    accuracy                           0.81      3349
   macro avg       0.81      0.81      0.81      3349
weighted avg       0.81      0.81      0.81      3349
```

Out[23]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=11)
```

In [39]:
```python
from sklearn.model_selection import GridSearchCV
params = {'criterion' : ['gini', 'entropy'],
          'max_depth' : [ 3, 4, 5, 7],
          'min_samples_leaf' : [10, 20, 50,100,150],
         }
grid_search = GridSearchCV(dt, param_grid= params)
```

In [41]:
```python
grid_search.fit(xtrain, ytrain)
```

Out[41]:
```
GridSearchCV(estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 4, 5, 7],
                         'min_samples_leaf': [10, 20, 50, 100, 150]})
```

In [42]:
```python
grid_search.best_params_
```

Out[42]:
```
{'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 50}
```

In [43]:
```python
my_best_preds = grid_search.predict(xtest)
```

In [44]:
```python
accuracy_score(ytest, my_best_preds)
```

Out[44]:
```
0.809495371752762
```

In [45]:
```python
print(classification_report(ytest, my_best_preds))
```

```
              precision    recall  f1-score   support

           0       0.81      0.83      0.82      1760
           1       0.81      0.78      0.80      1589

    accuracy                           0.81      3349
   macro avg       0.81      0.81      0.81      3349
weighted avg       0.81      0.81      0.81      3349
```

RandomForestClassifier

In [47]:
```python
from sklearn.ensemble import RandomForestClassifier
my_rf_classifier = RandomForestClassifier()
```

In [49]:
```python
my_rf_classifier.fit(xtrain, ytrain)
```

Out[49]:
```
RandomForestClassifier()
```

In [50]:
```python
my_predictions = my_rf_classifier.predict(xtest)
```

In [52]:
```python
print(accuracy_score(ytest, my_predictions))
```

```
0.841743804120633
```

In [53]:
```python
print(classification_report(ytest, my_predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.81      0.84      1760
           1       0.81      0.87      0.84      1589

    accuracy                           0.84      3349
   macro avg       0.84      0.84      0.84      3349
weighted avg       0.84      0.84      0.84      3349
```

Voting Classifier

In [56]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

In [57]:
```python
df_clf = DecisionTreeClassifier()
log_clf = LogisticRegression()
svm_clf = SVC()
```

In [58]:
```python
voting_clf = VotingClassifier(estimators=[('lr', log_clf), ('df', df_clf), ('svc', svm_clf)])
```

In [60]:
```python
voting_clf.fit(xtrain, ytrain)
```

```
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[60]:
```
VotingClassifier(estimators=[('lr', LogisticRegression()),
                             ('df', DecisionTreeClassifier()), ('svc', SVC())])
```

In [61]:
```python
from sklearn.metrics import accuracy_score
```

In [62]:
```python
for clf in (log_clf, df_clf, svm_clf, voting_clf):
    clf.fit(xtrain, ytrain)
    y_pred = clf.predict(xtest)
    print(clf.__class__.__name__, accuracy_score(ytest, ypred))
```

```
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression 0.8178560764407286
DecisionTreeClassifier 0.8178560764407286
SVC 0.8178560764407286
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
VotingClassifier 0.8178560764407286
```

Bagging Classifier

In [64]:
```python
from sklearn.ensemble import BaggingClassifier
```

In [65]:
```python
bag_clf = BaggingClassifier(log_clf,n_estimators=10)
```

In [67]:
```python
bag_clf.fit(xtrain,ytrain)
```

```
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91760\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[67]:
```
BaggingClassifier(base_estimator=LogisticRegression())
```

In [68]:
```python
ypred = bag_clf.predict(xtest)
accuracy_score(ypred, ytest)
```

Out[68]:
```
0.7703792176769185
```

In [ ]: