Stacking and Blending

In [116…
```
#Import dataset from seaborn
import seaborn as sns
data=sns.load_dataset('tips')
data.head()
```

Out[116]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [5]:
```
data['day'].unique()
```

Out[5]:
```
['Sun', 'Sat', 'Thur', 'Fri']
Categories (4, object): ['Thur', 'Fri', 'Sat', 'Sun']
```

In [7]:
```
data['time'].unique()
```

Out[7]:
```
['Dinner', 'Lunch']
Categories (2, object): ['Lunch', 'Dinner']
```

In [117…
```
#Finding null values
data.isnull().sum()
```

Out[117]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

In [9]:
```
from sklearn.preprocessing import LabelEncoder
```

In [118…
```
#Label encoder for the dependent variable 'time'
encoder=LabelEncoder()
data['time']=encoder.fit_transform(data['time'])
```

In [119…
```
#dropping dependent variable on X axis
X=data.drop(labels=['time'],axis=1)
```

In [120…
```
X
```

Out[120]:

	total_bill	tip	sex	smoker	day	size
0	16.99	1.01	Female	No	Sun	2
1	10.34	1.66	Male	No	Sun	3
2	21.01	3.50	Male	No	Sun	3
3	23.68	3.31	Male	No	Sun	2
4	24.59	3.61	Female	No	Sun	4
...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	3
240	27.18	2.00	Female	Yes	Sat	2
241	22.67	2.00	Male	Yes	Sat	2
242	17.82	1.75	Male	No	Sat	2
243	18.78	3.00	Female	No	Thur	2

244 rows × 6 columns

In [121…
```
#Assigning dependent variable 'time' on y axis
y=data.time
```

In [18]:
```
y
```

Out[18]:
```
0      0
1      0
2      0
3      0
4      0
      ..
239    0
240    0
241    0
242    0
243    0
Name: time, Length: 244, dtype: int32
```

In [19]:
```
X
```

Out[19]:

	total_bill	tip	sex	smoker	day	size
0	16.99	1.01	Female	No	Sun	2
1	10.34	1.66	Male	No	Sun	3
2	21.01	3.50	Male	No	Sun	3
3	23.68	3.31	Male	No	Sun	2
4	24.59	3.61	Female	No	Sun	4
...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	3
240	27.18	2.00	Female	Yes	Sat	2
241	22.67	2.00	Male	Yes	Sat	2
242	17.82	1.75	Male	No	Sat	2
243	18.78	3.00	Female	No	Thur	2

244 rows × 6 columns

In [20]:
```
X['day'].value_counts()
```

Out[20]:
```
Sat     87
Sun     76
Thur    62
Fri     19
Name: day, dtype: int64
```

In [21]:
```
from sklearn.model_selection import train_test_split
```

In [122…
```
#splitting train and test datasets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=42)
```

In [24]:
```
X_train
```

Out[24]:

	total_bill	tip	sex	smoker	day	size
228	13.28	2.72	Male	No	Sat	2
208	24.27	2.03	Male	Yes	Sat	2
96	27.28	4.00	Male	Yes	Fri	2
167	31.71	4.50	Male	No	Sun	4
84	15.98	2.03	Male	No	Thur	2
…	…	…	…	…	…	…
106	20.49	4.06	Male	Yes	Sat	2
14	14.83	3.02	Female	No	Sun	2
92	5.75	1.00	Female	Yes	Fri	2
179	34.63	3.55	Male	Yes	Sun	2
102	44.30	2.50	Female	Yes	Sat	3

195 rows × 6 columns

In [25]:
```
X_test
```

Out[25]:

	total_bill	tip	sex	smoker	day	size
24	19.82	3.18	Male	No	Sat	2
6	8.77	2.00	Male	No	Sun	2
153	24.55	2.00	Male	No	Sun	4
211	25.89	5.16	Male	Yes	Sat	4
198	13.00	2.00	Female	Yes	Thur	2
176	17.89	2.00	Male	Yes	Sun	2
192	28.44	2.56	Male	Yes	Thur	2
124	12.48	2.52	Female	No	Thur	2
9	14.78	3.23	Male	No	Sun	2
101	15.38	3.00	Female	Yes	Fri	2
45	18.29	3.00	Male	No	Sun	2
233	10.77	1.47	Male	No	Sat	2
117	10.65	1.50	Female	No	Thur	2
177	14.48	2.00	Male	Yes	Sun	2
82	10.07	1.83	Female	No	Thur	1
146	18.64	1.36	Female	No	Thur	3
200	18.71	4.00	Male	Yes	Thur	3
15	21.58	3.92	Male	No	Sun	2
66	16.45	2.47	Female	No	Sat	2
142	41.19	5.00	Male	No	Thur	5
33	20.69	2.45	Female	No	Sat	4
19	20.65	3.35	Male	No	Sat	3
109	14.31	4.00	Female	Yes	Sat	2
30	9.55	1.45	Male	No	Sat	2

	total_bill	tip	sex	smoker	day	size
186	20.90	3.50	Female	Yes	Sun	3
120	11.69	2.31	Male	No	Thur	2
10	10.27	1.71	Male	No	Sun	2
73	25.28	5.00	Female	Yes	Sat	2
159	16.49	2.00	Male	No	Sun	4
156	48.17	5.00	Male	No	Sun	6
112	38.07	4.00	Male	No	Sun	3
218	7.74	1.44	Male	Yes	Sat	2
25	17.81	2.34	Male	No	Sat	4
60	20.29	3.21	Male	Yes	Sat	2
18	16.97	3.50	Female	No	Sun	3
119	24.08	2.92	Female	No	Thur	4
97	12.03	1.50	Male	Yes	Fri	2
197	43.11	5.00	Female	Yes	Thur	4
139	13.16	2.75	Female	No	Thur	2
241	22.67	2.00	Male	Yes	Sat	2
75	10.51	1.25	Male	No	Sat	2
127	14.52	2.00	Female	No	Thur	2
113	23.95	2.55	Male	No	Sun	2
16	10.33	1.67	Female	No	Sun	3
196	10.34	2.00	Male	Yes	Thur	2
67	3.07	1.00	Female	Yes	Sat	1
168	10.59	1.61	Female	Yes	Sat	2
38	18.69	2.31	Male	No	Sat	3

	total_bill	tip	sex	smoker	day	size
195	7.56	1.44	Male	No	Thur	2

Data Preprocessing

In [29]:
```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

In [30]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 195 entries, 228 to 102
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   total_bill  195 non-null    float64 
 1   tip         195 non-null    float64 
 2   sex         195 non-null    category
 3   smoker      195 non-null    category
 4   day         195 non-null    category
 5   size        195 non-null    int64   
dtypes: category(3), float64(2), int64(1)
memory usage: 7.1 KB
```

In [123…
```
#catagorical and numerical variable
cat_col=['sex','smoker','day']
num_col=['total_bill','tip','size']
```

In [126…
```
#Creating Pipeline
num_pipeline=Pipeline(

    steps=[
        ('imputer',SimpleImputer(strategy='median')),  #for missing values
        ('scaler', StandardScaler())         #for scaling
    ]
)
```

```
cat_pipeline=Pipeline(

    steps=[
        ('imputer',SimpleImputer(strategy='most_frequent')),  #for missing values
        ('encoder',OneHotEncoder())            #encoding catagory variables
    ]
)
```

In [125…
```
#Column Transformer
preprocessor=ColumnTransformer([

    ('num_pipeline',num_pipeline,num_col),
    ('cat_pipeline',cat_pipeline,cat_col)
])
```

In [127…
```
#fit the model to xtrain and xtest
X_train=preprocessor.fit_transform(X_train)
X_test=preprocessor.fit_transform(X_test)
```

In [39]:
```
X_train[0]
```

Out[39]:
```
array([-0.79306155, -0.2580329 , -0.61214068,  0.        ,  1.        ,
        1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
        0.        ])
```

In [40]:
```
y_train
```

Out[40]:
```
228    0
208    0
96     0
167    0
84     1
      ..
106    0
14     0
92     0
179    0
102    0
Name: time, Length: 195, dtype: int32
```

In [41]:
```
X_test
```

Out[41]:
```
array([[ 0.19029449,  0.48117084, -0.55602186,  0.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
         0.        ],
       [-1.00973989, -0.57426535, -0.55602186,  0.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
         0.        ],
       [ 0.70397437, -0.57426535,  1.46213155,  0.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
         0.        ],
       [ 0.8494989 ,  2.25215697,  1.46213155,  0.        ,  1.        ,
         0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
         0.        ],
       [-0.55036021, -0.57426535, -0.55602186,  1.        ,  0.        ,
         0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
         1.        ],
       [-0.01930427, -0.57426535, -0.55602186,  0.        ,  1.        ,
         0.        ,  1.        ,  0.        ,  0.        ,  1.        ,
         0.        ],
       [ 1.12642991, -0.07338038, -0.55602186,  0.        ,  1.        ,
         0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
         1.        ],
       [-0.60683242, -0.10915788, -0.55602186,  1.        ,  0.        ,
         1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         1.        ],
       [-0.35705151,  0.52589271, -0.55602186,  0.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
         0.        ],
       [-0.29189127,  0.3201721 , -0.55602186,  1.        ,  0.        ,
         0.        ,  1.        ,  1.        ,  0.        ,  0.        ,
         0.        ],
       [ 0.02413588,  0.3201721 , -0.55602186,  0.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
         0.        ],
       [-0.7925391 , -1.04831719, -0.55602186,  0.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
         0.        ],
       [-0.80557115, -1.02148407, -0.55602186,  1.        ,  0.        ,
         1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         1.        ],
       [-0.38963163, -0.57426535, -0.55602186,  0.        ,  1.        ,
         0.        ,  1.        ,  0.        ,  0.        ,  1.        ,
         0.        ],
       [-0.86855938, -0.72631971, -1.56509856,  1.        ,  0.        ,
         1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
```

```
  1.        ],
[ 0.06214602, -1.14670531,  0.45305485,  1.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.        ],
[ 0.06974805,  1.21460954,  0.45305485,  0.        ,  1.        ,
  0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
  1.        ],
[ 0.38143119,  1.14305454, -0.55602186,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
[-0.17568885, -0.15387975, -0.55602186,  1.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[ 2.51108497,  2.10904698,  2.47120825,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.        ],
[ 0.28477684, -0.1717685 ,  1.46213155,  1.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[ 0.28043282,  0.6332252 ,  0.45305485,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[-0.40809369,  1.21460954, -0.55602186,  1.        ,  0.        ,
  0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[-0.92503158, -1.06620594, -0.55602186,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[ 0.30758292,  0.76739082,  0.45305485,  1.        ,  0.        ,
  0.        ,  1.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
[-0.69262673, -0.29698974, -0.55602186,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.        ],
[-0.8468393 , -0.8336522 , -0.55602186,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
[ 0.78325266,  2.10904698, -0.55602186,  1.        ,  0.        ,
  0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[-0.17134483, -0.57426535,  1.46213155,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
[ 3.26911574,  2.10904698,  3.48028496,  0.        ,  1.        ,
```

```
  1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
[ 2.17225173,  1.21460954,  0.45305485,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
[-1.1215983 , -1.07515031, -0.55602186,  0.        ,  1.        ,
  0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[-0.02799231, -0.27015662,  1.46213155,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[ 0.24133668,  0.50800396, -0.55602186,  0.        ,  1.        ,
  0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[-0.11921664,  0.76739082,  0.45305485,  1.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
[ 0.65293218,  0.2486171 ,  1.46213155,  1.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.        ],
[-0.6557026 , -1.02148407, -0.55602186,  0.        ,  1.        ,
  0.        ,  1.        ,  1.        ,  0.        ,  0.        ,
  0.        ],
[ 2.71959773,  2.10904698,  1.46213155,  1.        ,  0.        ,
  0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
  1.        ],
[-0.53298415,  0.09656274, -0.55602186,  1.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.        ],
[ 0.49980562, -0.57426535, -0.55602186,  0.        ,  1.        ,
  0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[-0.8207752 , -1.24509343, -0.55602186,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
  0.        ],
[-0.38528761, -0.57426535, -0.55602186,  1.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.        ],
[ 0.63881413, -0.08232475, -0.55602186,  0.        ,  1.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
[-0.84032327, -0.8694297 ,  0.45305485,  1.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ],
```

```
       [-0.83923727, -0.57426535, -0.55602186,  0.        ,  1.        ,
         0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
         1.        ],
       [-1.62876215, -1.46870279, -1.56509856,  1.        ,  0.        ,
         0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
         0.        ],
       [-0.81208717, -0.92309595, -0.55602186,  1.        ,  0.        ,
         0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
         0.        ],
       [ 0.06757604, -0.29698974,  0.45305485,  0.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
         0.        ],
       [-1.14114637, -1.07515031, -0.55602186,  0.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         1.        ]])
```

In [42]:
```
y_test
```

Out[42]:

24	0
6	0
153	0
211	0
198	1
176	0
192	1
124	1
9	0
101	0
45	0
233	0
117	1
177	0
82	1
146	1
200	1
15	0
66	0
142	1
33	0
19	0
109	0
30	0
186	0
120	1
10	0
73	0
159	0
156	0
112	0
218	0
25	0
60	0
18	0
119	1
97	0
197	1
139	1
241	0
75	0
127	1
113	0
16	0

```
196    1
67     0
168    0
38     0
195    1
Name: time, dtype: int32
```

Model Building

In [47]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

In [130…
```
#Defining number of models
models={
    'random_forest':RandomForestClassifier(),
    'logistic_regression':LogisticRegression(),
    'decision_tree':DecisionTreeClassifier(),

}
```

In [131…
```
#function to fit models to xtrain ytrain xtest ytest
from sklearn.metrics import accuracy_score
def evaluate_model(X_train,y_train,X_test,y_test,models):
    report={}
    for i in range(len(models)):
        model=list(models.values())[i]
        model.fit(X_train,y_train)
        print(model.fit(X_train,y_train)
        #print('OOB score:', model.oob_score_))
)
        y_pred=model.predict(X_test)
        accuracy=round(accuracy_score(y_test,y_pred)*100,2)

        report[list(models.keys())[i]]=accuracy
    return report
```

In [70]:
```
list(models.values())[0]
```

Out[70]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [59]:
```
list(models.keys())[2]
```

Out[59]:
```
'decision_tree'
```

In [60]:
```
len(models)
```

Out[60]:
```
3
```

In [63]:
```
for i in range(len(models)):
    print(i)
```

```
0
1
2
```

In [84]:
```
evaluate_model(X_train,y_train,X_test,y_test,models)
```

Out[84]:
```
RandomForestClassifier()
LogisticRegression()
DecisionTreeClassifier()
{'random_forest': 0.9795918367346939,
 'logistic_regression': 1.0,
 'decision_tree': 0.9795918367346939}
```

In [132…
```
#hyperparameters
params={

    'n_estimators':[50,100,200],
    'criterion':['gini','entropy'],
    'max_depth':[3,5,10]
}
```

In [93]:
```
model= RandomForestClassifier(oob_score=True)
```

In [94]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [137…

```
# Randomized Search CV
cv=RandomizedSearchCV(model,param_distributions=params,scoring='accuracy',cv=5,verbose=3)
```

In [138…

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

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END criterion=gini, max_depth=10, n_estimators=200;, score=0.974 total time=   0.5s
[CV 2/5] END criterion=gini, max_depth=10, n_estimators=200;, score=0.923 total time=   0.5s
[CV 3/5] END criterion=gini, max_depth=10, n_estimators=200;, score=1.000 total time=   0.4s
[CV 4/5] END criterion=gini, max_depth=10, n_estimators=200;, score=0.949 total time=   0.4s
[CV 5/5] END criterion=gini, max_depth=10, n_estimators=200;, score=0.923 total time=   0.4s
[CV 1/5] END criterion=entropy, max_depth=3, n_estimators=50;, score=0.974 total time=   0.0s
[CV 2/5] END criterion=entropy, max_depth=3, n_estimators=50;, score=0.949 total time=   0.0s
[CV 3/5] END criterion=entropy, max_depth=3, n_estimators=50;, score=0.974 total time=   0.0s
[CV 4/5] END criterion=entropy, max_depth=3, n_estimators=50;, score=0.923 total time=   0.0s
[CV 5/5] END criterion=entropy, max_depth=3, n_estimators=50;, score=0.923 total time=   0.0s
[CV 1/5] END criterion=entropy, max_depth=3, n_estimators=200;, score=0.974 total time=   0.3s
[CV 2/5] END criterion=entropy, max_depth=3, n_estimators=200;, score=0.974 total time=   0.5s
[CV 3/5] END criterion=entropy, max_depth=3, n_estimators=200;, score=0.974 total time=   0.6s
[CV 4/5] END criterion=entropy, max_depth=3, n_estimators=200;, score=0.923 total time=   0.5s
[CV 5/5] END criterion=entropy, max_depth=3, n_estimators=200;, score=0.923 total time=   0.3s
[CV 1/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.974 total time=   0.2s
[CV 2/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.923 total time=   0.3s
[CV 3/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=1.000 total time=   0.5s
[CV 4/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.949 total time=   0.3s
[CV 5/5] END criterion=entropy, max_depth=5, n_estimators=200;, score=0.923 total time=   0.2s
[CV 1/5] END criterion=entropy, max_depth=5, n_estimators=50;, score=0.974 total time=   0.0s
[CV 2/5] END criterion=entropy, max_depth=5, n_estimators=50;, score=0.923 total time=   0.0s
[CV 3/5] END criterion=entropy, max_depth=5, n_estimators=50;, score=0.974 total time=   0.0s
[CV 4/5] END criterion=entropy, max_depth=5, n_estimators=50;, score=0.923 total time=   0.0s
[CV 5/5] END criterion=entropy, max_depth=5, n_estimators=50;, score=0.949 total time=   0.0s
[CV 1/5] END criterion=gini, max_depth=10, n_estimators=100;, score=0.974 total time=   0.1s
[CV 2/5] END criterion=gini, max_depth=10, n_estimators=100;, score=0.923 total time=   0.1s
[CV 3/5] END criterion=gini, max_depth=10, n_estimators=100;, score=1.000 total time=   0.1s
[CV 4/5] END criterion=gini, max_depth=10, n_estimators=100;, score=0.923 total time=   0.1s
[CV 5/5] END criterion=gini, max_depth=10, n_estimators=100;, score=0.923 total time=   0.1s
[CV 1/5] END criterion=gini, max_depth=5, n_estimators=100;, score=0.974 total time=   0.1s
[CV 2/5] END criterion=gini, max_depth=5, n_estimators=100;, score=0.923 total time=   0.1s
[CV 3/5] END criterion=gini, max_depth=5, n_estimators=100;, score=0.974 total time=   0.1s
[CV 4/5] END criterion=gini, max_depth=5, n_estimators=100;, score=0.949 total time=   0.1s
[CV 5/5] END criterion=gini, max_depth=5, n_estimators=100;, score=0.949 total time=   0.1s
[CV 1/5] END criterion=gini, max_depth=3, n_estimators=50;, score=0.974 total time=   0.0s
[CV 2/5] END criterion=gini, max_depth=3, n_estimators=50;, score=0.923 total time=   0.0s
[CV 3/5] END criterion=gini, max_depth=3, n_estimators=50;, score=0.974 total time=   0.0s
[CV 4/5] END criterion=gini, max_depth=3, n_estimators=50;, score=0.923 total time=   0.1s
[CV 5/5] END criterion=gini, max_depth=3, n_estimators=50;, score=0.923 total time=   0.1s
[CV 1/5] END criterion=gini, max_depth=3, n_estimators=200;, score=0.974 total time=   0.4s
[CV 2/5] END criterion=gini, max_depth=3, n_estimators=200;, score=0.949 total time=   0.5s
[CV 3/5] END criterion=gini, max_depth=3, n_estimators=200;, score=0.974 total time=   0.3s
```

```
[CV 4/5] END criterion=gini, max_depth=3, n_estimators=200;, score=0.923 total time=   0.4s
[CV 5/5] END criterion=gini, max_depth=3, n_estimators=200;, score=0.923 total time=   0.3s
[CV 1/5] END criterion=entropy, max_depth=3, n_estimators=100;, score=0.974 total time=   0.0s
[CV 2/5] END criterion=entropy, max_depth=3, n_estimators=100;, score=0.949 total time=   0.1s
[CV 3/5] END criterion=entropy, max_depth=3, n_estimators=100;, score=0.974 total time=   0.1s
[CV 4/5] END criterion=entropy, max_depth=3, n_estimators=100;, score=0.923 total time=   0.1s
[CV 5/5] END criterion=entropy, max_depth=3, n_estimators=100;, score=0.923 total time=   0.0s
```

Out[138]:

RandomizedSearchCV

estimator: RandomForestClassifier

RandomForestClassifier

In [140…

```
#finding best parameters
cv.best_params_
```

Out[140]:

```
{'n_estimators': 200, 'max_depth': 10, 'criterion': 'gini'}
```

In [141…

```
bestmodel= RandomForestClassifier(n_estimators=200,max_depth=10,criterion='gini',oob_score=True)
```

In [142…

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

Out[142]:

RandomForestClassifier

RandomForestClassifier(max_depth=10, n_estimators=200, oob_score=True)

In [143…

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

In [144…

```
best_accuracy=round(accuracy_score(y_test,y_pred)*100,2)
```

In [145…

```
best_accuracy
```

Out[145]:

```
97.96
```