In [1]:
```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
```

In [2]:
```
dataset=load_iris()
x=dataset.data
y=dataset.target
print(x.shape)
print(y.shape)
```

```
(150, 4)
(150,)
```

In [3]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3, random_
print(xtrain.shape)
print(xtest.shape)
print(ytrain.shape)
print(ytest.shape)
```

```
(105, 4)
(45, 4)
(105,)
(45,)
```

AdaBoost

In [4]:
```
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier(n_estimators=50)
model.fit(xtrain, ytrain)
```

Out[4]:
```
AdaBoostClassifier
AdaBoostClassifier()
```

In [5]:
```
ypred = model.predict(xtest)
```

In [6]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(ytest, ypred))
```

```
0.9555555555555556
```

Gradient Boosting

In [7]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier
```

In [9]:
```
train_data = pd.read_csv("train (1).csv")
test_data = pd.read_csv("test (1).csv")
```

In [10]:
```
ytrain = train_data['Survived']
train_data.drop(labels='Survived', axis=1, inplace=True)
```

In [11]:
```
test_data.head()
```

Out[11]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

In [12]:
```
drop_columns=['Name','Age','Parch','Ticket','Cabin','Embarked']
train_data.drop(labels=drop_columns, axis=1, inplace=True)
test_data.drop(labels=drop_columns, axis=1, inplace=True)
drop_columns
```

Out[12]:
```
['Name', 'Age', 'Parch', 'Ticket', 'Cabin', 'Embarked']
```

In [13]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train_data['Sex'] = le.fit_transform(train_data['Sex'])
test_data['Sex'] = le.transform(test_data['Sex'])
le
```

Out[13]:
```
LabelEncoder
LabelEncoder()
```

In [14]:
```
# Ensure you are using the correct 'train_data' for defining xtrain be
xtrain = train_data
xtrain
```

Out[14]:

	PassengerId	Pclass	Sex	SibSp	Fare
0	1	3	1	1	7.2500
1	2	1	0	1	71.2833
2	3	3	0	0	7.9250
3	4	1	0	1	53.1000
4	5	3	1	0	8.0500
...	...	...	...	...	...
886	887	2	1	0	13.0000
887	888	1	0	0	30.0000
888	889	3	0	1	23.4500
889	890	1	1	0	30.0000
890	891	3	1	0	7.7500

891 rows × 5 columns

In [15]:
```
state=12
test_size=0.3
xtrain,xval,ytrain,yval=train_test_split(xtrain,ytrain,test_size=test_
```

In [16]:
```
lr_list = [0.05,0.075,0.1,0.25,0.5,0.75,1]
for learning_rate in lr_list:
    gb_clf = GradientBoostingClassifier(n_estimators=20, learning_rat
                                        max_features=2, max_depth=2,
    gb_clf.fit(xtrain, ytrain)
    print("Learning rate: ", learning_rate)
    print("Accuracy score (training): {0:.3f}".format(gb_clf.score(xt
    print("Accuracy score (testing): {0:.3f}".format(gb_clf.score(xva
```

```
Learning rate:  0.05
Accuracy score (training): 0.806
Accuracy score (testing): 0.735
Learning rate:  0.075
Accuracy score (training): 0.822
Accuracy score (testing): 0.776
Learning rate:  0.1
Accuracy score (training): 0.822
Accuracy score (testing): 0.784
Learning rate:  0.25
Accuracy score (training): 0.835
Accuracy score (testing): 0.765
Learning rate:  0.5
Accuracy score (training): 0.848
Accuracy score (testing): 0.757
Learning rate:  0.75
Accuracy score (training): 0.856
Accuracy score (testing): 0.772
Learning rate:  1
Accuracy score (training): 0.865
Accuracy score (testing): 0.754
```

In [17]:
```
gb_clf2 = GradientBoostingClassifier(n_estimators=20, learning_rate=0
                                     max_features=2, max_depth=2, ran
gb_clf2.fit(xtrain, ytrain)
```

Out[17]:
```
GradientBoostingClassifier(learning_rate=0.5, max_depth=2, max_features=
2,
                           n_estimators=20, random_state=0)
```

In [18]:
```
predictions = gb_clf2.predict(xval)
predictions
print("Confusion Matrix:")
print(confusion_matrix(yval, predictions))
```

```
Confusion Matrix:
[[136  25]
 [ 40  67]]
```

In [ ]:

```
1
```