Train and Test Data(Data Science)

September 1, 2019

[6]:
```
from sklearn.datasets import load_iris
iris=load_iris()
X=iris.data
y=iris.target
#importing train_test_split
from sklearn.model_selection import train_test_split
#70% of data is taken for training and 30% is taken for testing
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

[7]:
```
#printing total number of rows and columns(rows=150,column=4)
print(X.shape)
```

```
(150, 4)
```

[8]:
```
#70% of data is taken for training
print(X_train.shape)
#30% of data is taken for testing
print(X_test.shape)
```

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

[9]:
```
print(y_train.shape)
print(y_test.shape)
```

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

[10]:
```
#Fitting the Model using LogisticRegressionb
from sklearn.linear_model import LogisticRegression
logisticreg=LogisticRegression()
logisticreg.fit(X_train,y_train)
```

```
/home/sakil/anaconda/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
```

```
  FutureWarning)
/home/sakil/anaconda/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
```

[10]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

[11]:
```
#predicting the target_name
y_pred=logisticreg.predict(X_test)
```

[15]:
```
#Accuracy of the algorithm
from sklearn import metrics
print(metrics.accuracy_score(y_test,y_pred))
```

```
0.9555555555555556
```

[16]:
```
#Another way of predicting the accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

[16]:
```
0.9555555555555556
```

we can say that LogisticRegression algorithm gives an accuracy of 95.55% Now we are testing the accuracy of KNN Algorithm

[22]:
```
#importing KNNClassifier from sklearn
from sklearn.neighbors import KNeighborsClassifier
#n_neighbors=1,remaining parameters are default
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
y_pred=knn.predict(X_test)
print(metrics.accuracy_score(y_test,y_pred))
```

```
0.9777777777777777
```

We can say that KNNClassifier gives 97.77% of accuracy
K Fold Cross Validation

[31]:
```
from sklearn.model_selection import cross_val_score
knn5=KNeighborsClassifier(n_neighbors=5)
scores=cross_val_score(knn5,X,y,cv=10,scoring='accuracy')
print(scores)
```

```
[1.         0.93333333 1.         1.         0.86666667 0.93333333
 0.93333333 1.         1.         1.        ]
```

[32]:
```
print(scores.mean())
```

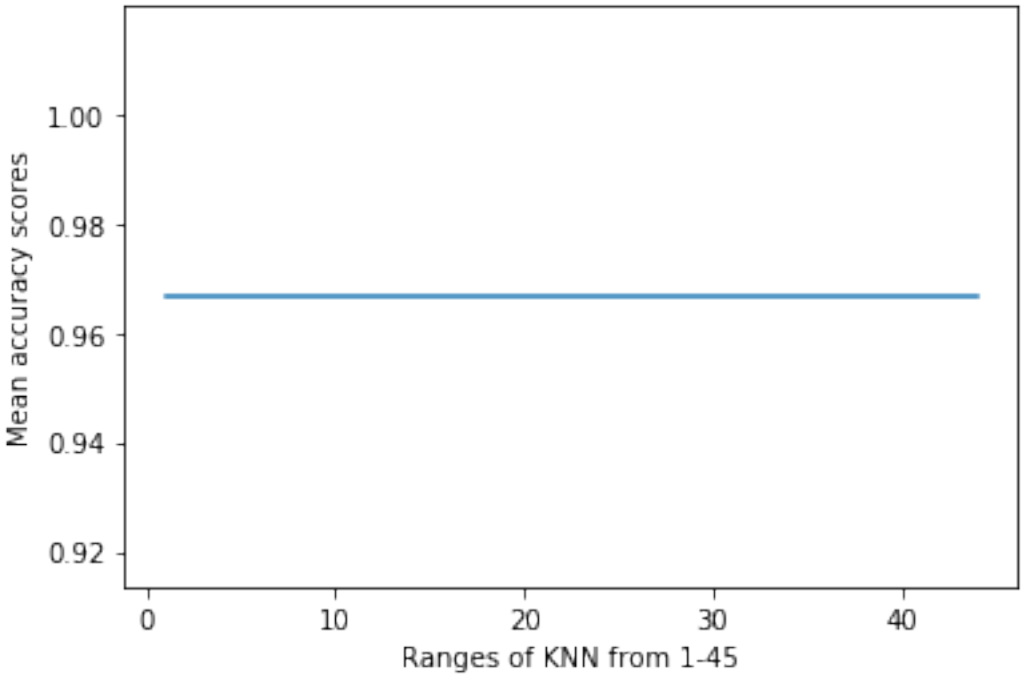
```
0.9666666666666668
```

[33]:
```
k_range=range(1,45)
k_score=[]
for k in k_range:
    knn_range=KNeighborsClassifier(n_neighbors=5)
    scores=cross_val_score(knn_range,X,y,cv=10,scoring='accuracy')
    k_score.append(scores.mean())
print(k_score)
```

```
[0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668]
```

[37]:
```
#plotting k_range vs k_score in grraph
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(k_range,k_score)
plt.xlabel("Ranges of KNN from 1-45")
plt.ylabel("Mean accuracy scores")
```

[37]:
```
Text(0, 0.5, 'Mean accuracy scores')
```


1.00
0.98
0.96
0.94
0.92
Mean accuracy scores
0
10
20
30
40
Ranges of KNN from 1-45