**Roll No : B-05**

**Name : Anuj Somani**

**Batch : TC1**

**Assignment Title : To predict the class of flower based on the available attributes using KNN Classifier.**

**Problem Statement : Use of Iris Dataset to predict Flower**

**Theory:**

import pandas as pd

from sklearn.datasets import load\_iris

In [1]:

iris=load\_iris()

In [2]:

iris.feature\_names

In [3]:

Out[3]: ['sepal length (cm)',

'sepal width (cm)',

'petal length (cm)', 'petal width (cm)']

iris.target\_names

In [4]:

Out[4]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')

x=iris.data y=iris.target

In [5]:

x.shape

In [6]:

Out[6]: (150, 4)

y.shape

In [7]:

Out[7]: (150,)

x[:5]

In [14]:

Out[14]: array([[5.1, 3.5, 1.4, 0.2],

[4.9, 3. , 1.4, 0.2],

[4.7, 3.2, 1.3, 0.2],

[4.6, 3.1, 1.5, 0.2],

[5. , 3.6, 1.4, 0.2]])

y[100:150]

In [15]:

Out[15]: array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,

2, 2, 2, 2, 2, 2])

iris.target\_names

In [16]:

Out[16]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')

df\_x=pd.DataFrame(x,columns=iris.feature\_names) df\_x

In [17]:

Out[17]: sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)

0 5.1 3.5 1.4 0.2

1 4.9 3.0 1.4 0.2

2 4.7 3.2 1.3 0.2

3 4.6 3.1 1.5 0.2

4 5.0 3.6 1.4 0.2

... ... ... ... ...

145 6.7 3.0 5.2 2.3

146 6.3 2.5 5.0 1.9

147 6.5 3.0 5.2 2.0

148 6.2 3.4 5.4 2.3

149 5.9 3.0 5.1 1.8

150 rows × 4 columns

df\_y=pd.DataFrame(y,columns=['Output']) df\_y

In [18]:

Out[18]: Output

0 0

1 0

2 0

3 0

4 0

... ...

145 2

146 2

147 2

148 2

149 2

150 rows × 1 columns

df\_y['Output'].unique()

In [19]:

Out[19]: array([0, 1, 2])

df\_y['Output'].value\_counts()

In [20]:

Out[20]: 0 50

1 50

2 50

Name: Output, dtype: int64

iris.target\_names[0]

In [21]:

Out[21]: 'setosa'

df=pd.concat([df\_x,df\_y],axis=1) # row wise concatination of two data frames

df.head()

In [22]:

Out[22]: sepal length (cm) sepal width (cm) petal length (cm) petal width (cm) Output

0 5.1 3.5 1.4 0.2 0

1 4.9 3.0 1.4 0.2 0

2 4.7 3.2 1.3 0.2 0

3 4.6 3.1 1.5 0.2 0

4 5.0 3.6 1.4 0.2 0

#x=df.iloc[:-1].values

In [23]:

#y=df.iloc[:,-1].values

In [24]:

from sklearn.model\_selection import train\_test\_split X\_train,X\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.4,random\_state=41)

In [25]:

from sklearn.neighbors import KNeighborsClassifier

knn\_classifier=KNeighborsClassifier(n\_neighbors=3) knn\_classifier.fit(X\_train,y\_train) # Training the KNN classifier

y\_pred=knn\_classifier.predict(X\_test) #Predicting output by the KNN classifier

# Finding accuracy by comparing actual output values(y\_test)with predicted #output value(y\_pred)

from sklearn import metrics

print("Accuracy:", metrics.accuracy\_score(y\_test,y\_pred))

In [26]:

Accuracy: 0.9333333333333333

# Providing sample data and the model will make prediction out of that data

sample = [[5, 5, 3, 2], [2, 4, 3, 5]]

preds = knn\_classifier.predict(sample)

In [27]:

preds

In [28]:

Out[28]: array([0, 2])

iris.target\_names[preds[1]]

In [34]:

Out[34]: 'virginica'

iris.target\_names[y\_pred[:5]]

In [49]:

Out[49]: array(['virginica', 'virginica', 'virginica', 'versicolor', 'virginica'], dtype='<U10')

import numpy as np

np.sum(y\_pred!=y\_test) #sum of rows whose predicted output is not equal to actual output

In [30]:

Out[30]: 4

df\_actual=pd.DataFrame(y\_test,columns=['Actual\_Output']) df\_predicted=pd.DataFrame(y\_pred,columns=['Predicted\_Output']) df\_output=pd.concat([df\_actual,df\_predicted],axis=1) df\_output[:5]

In [47]:

Out[47]:

Actual\_Output Predicted\_Output 0 2 2

3 1 1

1 2 2

2 2 2

4 2 2

df\_output[df\_output['Actual\_Output']!=df\_output['Predicted\_Output']]

In [46]:

Out[46]:

Actual\_Output Predicted\_Output 15 2 1

39 1 2

17 1 2

32 2 1

**Conclusion: Here we have learned about knn classifier using iris dataset.**