ROll NO:RBT21CB013

KNN implementation on iris Dataset

KNN can be used for both classification and regression predictive problems. KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labelled dataset consiting of training observations (x,y) and would like to capture the relationship between x and y . More formally, our goal is to learn a function h:X⟶Y so that given an unseen observation x , h(x) can confidently predict the corresponding output y .

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('Iris.csv')


df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
#Encoding the categorical column
df = df.replace({"class":  {"Iris-setosa":1,"Iris-versicolor":2, "Iris-virginica":3}})
#Visualize the new dataset
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

X=df[['SepalLengthCm','SepalWidthCm','PetalLengthCm']]
Y=df['Species']
X
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	5.1	3.5	1.4
1	4.9	3.0	1.4
2	4.7	3.2	1.3
3	4.6	3.1	1.5
4	5.0	3.6	1.4
...	...	...	...
145	6.7	3.0	5.2

```
Y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: Species, Length: 150, dtype: int64
```

```
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size = .25,random_state=0)
X_train.shape
```

```
(112, 3)
```

```
y_train.shape
```

```
(112,)
```

```
X_test.shape
```

```
(38, 3)
```

```
y_test.shape
```

```
(38,)
```

```
Y.dtype
```

```
dtype('int64')
```

```
knn=KNeighborsClassifier(n_neighbors=50, metric='minkowski')

knn.fit(X_train,y_train)
```

```
▾        KNeighborsClassifier
KNeighborsClassifier(n_neighbors=50)
```

```
import numpy as np

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

```
array([2, 1, 0, 2, 0, 2, 0, 2, 2, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 2])
```

```
prediction1=knn.predict([[2,0,1]])  # new mass, width,height
prediction1
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsClassifier
  warnings.warn(
array([0])
```

```
from sklearn.metrics import accuracy_score
print("Accuracy is",accuracy_score(y_test, y_pred))
```

```
     Accuracy is 0.868421052631579
```