23/01/2022
Sunday

EXERCISE-05

K-NEAREST NEIGHBOUR ALGORITHM

INTRODUCTION

K-Nearest Neighbour is one of the simplest machine learning algorithms based on supervised Learning Technique. KNN Algorithm Assumes the similarity between new case/ data and available cases and put the new cases into the category that is most similar to the available categories. It stores all the available data and classifies a new data point based on similarity. KNN Algorithm can be used for Regression as well as for classification. But mostly it is used for classification problem. It is a non-parametric algorithm, which means it doesn't make any assumptions on underlying data. It is also called Lazy Learner Algorithm because it does not learn from the training set, Instead it stores the dataset.

Why KNN Algorithm is used?

Suppose there are two categories, i.e, category A and category B, and we have a new data point x_1, So this data point will lie in which of these categories. To solve this type of problem, we need a KNN algorithm. With the help of KNN, we can easily identify the category or class of a particular dataset.

AIM

To implement KNN Algorithm.

ALGORITHM

```
- Load the data
- Choose K value
- For each data point in the data,
    - Find the Euclidean distance to all training data samples.
    - Store the distances on an ordered list and sort it.
    - Choose the top K entries from the sorted list.
    - Label the test point based on the majority of classes
      present in the selected points.
- End.
```

To validate the accuracy of the KNN classification, a confusion matrix is used. Other statistical methods such as the liklihood-ratio test are also used for validation.

There is no particular way to determine the best value of 'K', so we need to try some values to find the best out of them.

OUTPUT

```
----------------------------------------
Extracting Features

['sepal length (cm)', 'sepal width (cm)', 'petal
 length (cm)', 'petal width (cm)']
----------------------------------------
Extracting Targets

['setosa' 'versicolor' 'virginica']
----------------------------------------
Number of rows of columns in test data
(45, 4)
----------------------------------------
Number of rows of columns in train data
(105, 4)
----------------------------------------
Accuracy
0.9777777777777777
----------------------------------------
Predicted values
['versicolor']
----------------------------------------
```

PROGRAM

```
from sklearn.datasets import load_iris
iris = load_iris()
x = iris.data
y = iris.target
print("----------------------")
print("Extracting Features\n")
f_name = iris.feature_names
print(f_name)
print("----------------------")
print("Extracting Targets\n")
t_name = iris.target_names
print(t_name)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_-
                     size = 0.3, random_state = 1)
print("----------------------")
print("Number of rows of columns in test data\n")
print(x_test.shape)
print("----------------------")
print("Number of rows of columns in the train data\n")
print(x_train.shape).
```

```
from sklearn.neighbours import KNeighborsClassifier
c_knn = KNeighborsClasiffer(n_neighbors=3)
c_knn.fit(x_train,y_train)
y_pred = c_knn.predict(x_test)
from sklearn import metrices
print("- - - - - - - - - - - - - - - - - - -")
print("Accuracy\n")
print(metrics.accuracy_score(y_test,y_pred))
Sample = [[5,5,3,2]]
pred = c_knn.predict(sample)
pred_values = [iris.target_names[P] for P in pred]
print("- - - - - - - - - - - - - - - - - - -")
print("Predicted Values\n")
print(pred_val)
print("- - - - - - - - - - - - - - - - - - -")
```

RESULT

The program is executed successfully and the output is verified.