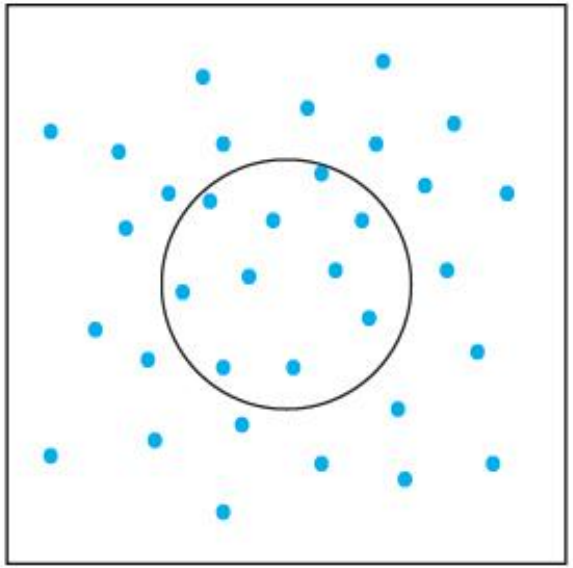
Machine Learning

By EvolkAI

Model Name: NearestCentroid

Nearest Centroid is a simple and intuitive algorithm used for classification tasks. It works by calculating the centroid, or mean, of each class in the training data. During classification, it assigns a new data point to the class whose centroid is closest to the point.
In simpler terms, Nearest Centroid finds the average location of each class based on the features of the training data. When a new data point needs to be classified, the algorithm calculates its distance from each centroid and assigns it to the class with the closest centroid.

Nearest Centroid is particularly useful when the classes have distinct and well-separated centroids. However, it may not perform well if the classes overlap or have complex boundaries.
The algorithm is computationally efficient and has low memory requirements, making it suitable for large datasets. It is commonly used in applications where interpretability and simplicity are valued, such as document classification and image recognition.



```
import pandas as pd
data = pd.read_csv('iris_dataset.csv')
data.info()
```

In First line we Import pandas library as pd, then we read iris_dataset.csv file using the read_csv() function, and prints information about the data using the info() method.

```
feature = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
predection_class = ['species']
X = data[feature].values
y = data[predection_class].values
```

This defines the ‘feature’ and ‘predection_class’ variables, which specify the columns of the data to use as ‘features’ and the column to use as the prediction target.
The code then creates ‘X’ and ‘y’ arrays containing the values of these columns from the data DataFrame.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,y,test_size=0.30)
```

This imports the train_test_split function from the sklearn.model_selection module and uses it to split the data into training and testing sets. The test_size parameter specifies that 30% of the data should be used for testing.

```
print(f"Shape of X_test is {X_test.shape}")
print(f"Shape of X_train is {X_train.shape}")
print(f"Shape of Y_test is {Y_test.shape}")
print(f"Shape of Y_train is {Y_train.shape}")
```

These lines print the shapes of the training and testing data arrays. This output totally depends on the test size we took while train_test_split.

OUTPUT

```
Shape of X_test is (45, 4)
Shape of X_train is (105, 4)
Shape of Y_test is (45, 1)
Shape of Y_train is (105, 1)
```

```
from sklearn.neighbors import NearestCentroid
clf = NearestCentroid()
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

This imports the *NearestCentroid* model from the sklearn neighbors library, creates a *NearestCentroid* classifier object, fits the classifier with the training data using the *fit()* method, and use the *predict()* method to generate predictions for the testing data.

```
from sklearn import metrics
print("Accuracy", metrics.accuracy_score(Y_test, Y_pred)*100)
```

This imports the metrics module from sklearn and uses the accuracy_score() function to calculate the accuracy of the model on the testing data. The result is printed in the console.

OUTPUT

```
Accuracy 91.11111111111111
```

Conclusion

In conclusion, Nearest Centroid is a simple yet effective algorithm for classification tasks. By calculating the centroids of each class and assigning new data points to the closest centroid, it provides a straightforward approach to classifying data. The algorithm is computationally efficient and has low memory requirements, making it suitable for large datasets.

While Nearest Centroid may not handle complex or overlapping classes well, it excels in scenarios where classes have distinct and well-separated centroids. Its interpretability and simplicity make it a valuable tool in applications such as document classification and image recognition. Overall, Nearest Centroid offers a practical and efficient solution for certain classification tasks.

Thank You