KNN_Regressor_Practice

December 21, 2021

0.1 Implementation of KNN in Python KNN-Regressor & KNN Predict & Evalution

0.2 [[3, 4, 5], [6, 9, 7], [2, 4, 5],[1, 3, 2], [7, 7, 7], [5, 6,7], [4, 4 ,8],[2, 2, 2],[3 ,5 ,1]]

0.3 With This Label for Classification==> [1,2,1,1,2,2,2,1,1]

0.4 Test Data ==> [5 ,5 ,5] , [6 , 3 ,2]

[107]:

```
import pandas as pd
import numpy as np
```

1 Load the Dataset

1.1 Create DataFrame from Train Data

[108]:

```
# Use Pandas lib for Create Dataframe
TrainData = pd.DataFrame([[3, 4, 5, 1],
                          [6, 9, 7, 2],
                          [2, 4, 5, 1],
                          [1, 3, 2, 1],
                          [7, 7, 7, 2],
                          [5, 6, 7, 2],
                          [4, 4 ,8, 2],
                          [2, 2, 3, 1],
                          [3, 5, 1, 1]],columns=['F1', 'F2', 'F3','Label'])
print('Show Train Data')
TrainData
```

```
Show Train Data
```

[108]:

```
   F1  F2  F3  Label
0   3   4   5      1
1   6   9   7      2
2   2   4   5      1
3   1   3   2      1
4   7   7   7      2
5   5   6   7      2
```

```
6   4   4   8       2
7   2   2   3       1
8   3   5   1       1
```

1.2 Create DataFrame from Test Data

```
[109]: TestData = pd.DataFrame([[5, 5, 5, 0],
                                [6, 3, 2, 0]],columns=['F1', 'F2', 'F3','Label'])
       print('Show Test Data')
       TestData
```

```
Show Test Data
```

```
[109]:    F1  F2  F3  Label
       0   5   5   5      0
       1   6   3   2      0
```

1.3 Select the Features

```
[110]: # Train Data
       X_train = TrainData.iloc[:,[0,1,2]].values # Features Data
       Y_train = TrainData.iloc[:,[3]].values     # Labeled Data

       # Test Data
       X_test = TestData.iloc[:,[0,1,2]].values
       Y_test = TestData.iloc[:,[3]].values
```

```
[111]: # Show Train Data
       X_train,Y_train
```

```
[111]: (array([[3, 4, 5],
                [6, 9, 7],
                [2, 4, 5],
                [1, 3, 2],
                [7, 7, 7],
                [5, 6, 7],
                [4, 4, 8],
                [2, 2, 3],
                [3, 5, 1]]), array([[1],
                [2],
                [1],
                [1],
                [2],
                [2],
                [2],
                [1],
                [1]]))
```

[112]:

```
# Show Test Data
X_test,Y_test
```

[112]:

```
(array([[5, 5, 5],
        [6, 3, 2]]), array([[0],
        [0]]))
```

1.3.1 Define Error Metrics

As this is a regression problem, we have defined MAPE as the error metrics as shown below

[113]:

```
def MAPE(Y_actual,Y_Predicted):
    Mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
    return Mape
```

1.4 Build the Model of KNN Classification

[114]:

```
#Building the KNN.Regressor Model on our dataset
k=3
from sklearn.neighbors import KNeighborsClassifier
KNN_model = KNeighborsClassifier(n_neighbors=k,metric='euclidean') # euclidean␣
 ↪& minkowski & manhattan &
KNN_model.fit(X_train,Y_train.ravel())
```

[114]:

```
KNeighborsClassifier(metric='euclidean', n_neighbors=3)
```

The following lists the string metric identifiers and the associated distance metric classes:

Metrics intended for real-valued vector spaces:

“euclidean” = > sqrt(sum((x - y)ˆ2))

“manhattan” => sum(|x - y|)

“chebyshev” => max(|x - y|)

“minkowski” => sum(|x - y|$^{p)}$(1/p)

“wminkowski” => sum(|w * (x - y)|$^{p)}$(1/p)

“seuclidean” => sqrt(sum((x - y)ˆ2 / V))

“mahalanobis” => sqrt((x - y)’ Vˆ-1 (x - y))

1.4.1 Predict the testing Data

[115]:
```
KNN_predict = KNN_model.predict(X_test)  # Predictions on Testing data
```

[116]:
```
X_test
```

[116]:
```
array([[5, 5, 5],
       [6, 3, 2]])
```

[117]:
```
Y_test = KNN_predict # Set Predicted label put on Y_Test
Y_test   # Predicted Values
```

[117]:
```
array([1, 1])
```

1.5 Accuracy Check For KNN Classification !

[118]:
```
# Using MAPE error metrics to check for the error rate and accuracy level
KNN_MAPE = MAPE(Y_train,KNN_predict)
Accuracy_KNN = 100 - KNN_MAPE
print("MAPE: ",KNN_MAPE)
print('Accuracy of KNN model: {:0.2f}%.'.format(Accuracy_KNN))
```

```
MAPE:  22.22222222222222
Accuracy of KNN model: 77.78%.
```

1.6 Build the Model of KNN Regressor Classification

[121]:
```
#Building the KNN.Regressor Model on our dataset
k=3
from sklearn.neighbors import KNeighborsRegressor
KNN_model = KNeighborsRegressor(n_neighbors=k).fit(X_train,Y_train)
```

1.6.1 Predict the testing Data

[122]:
```
KNN_predict = KNN_model.predict(X_test) #Predictions on Testing data
```

[123]:
```
X_test
```

[123]:
```
array([[5, 5, 5],
       [6, 3, 2]])
```

[124]:
```
Y_test = KNN_predict # Set Predicted label put on Y_Test
Y_test   # Predicted Values
```

[124]:
```
array([[1.33333333],
       [1.        ]])
```

1.7 Accuracy Check For KNN Regressor Classification!

[134]:

```
# Using MAPE error metrics to check for the error rate and accuracy level
KNN_MAPE = MAPE(Y_train.reshape(1, -1),KNN_predict)
Accuracy_KNN = 100 - KNN_MAPE
print("MAPE: ",KNN_MAPE)
print('Accuracy of KNN model: {:0.2f}%.'.format(Accuracy_KNN))
```

```
MAPE:  27.77777777777778
Accuracy of KNN model: 72.22%.
```