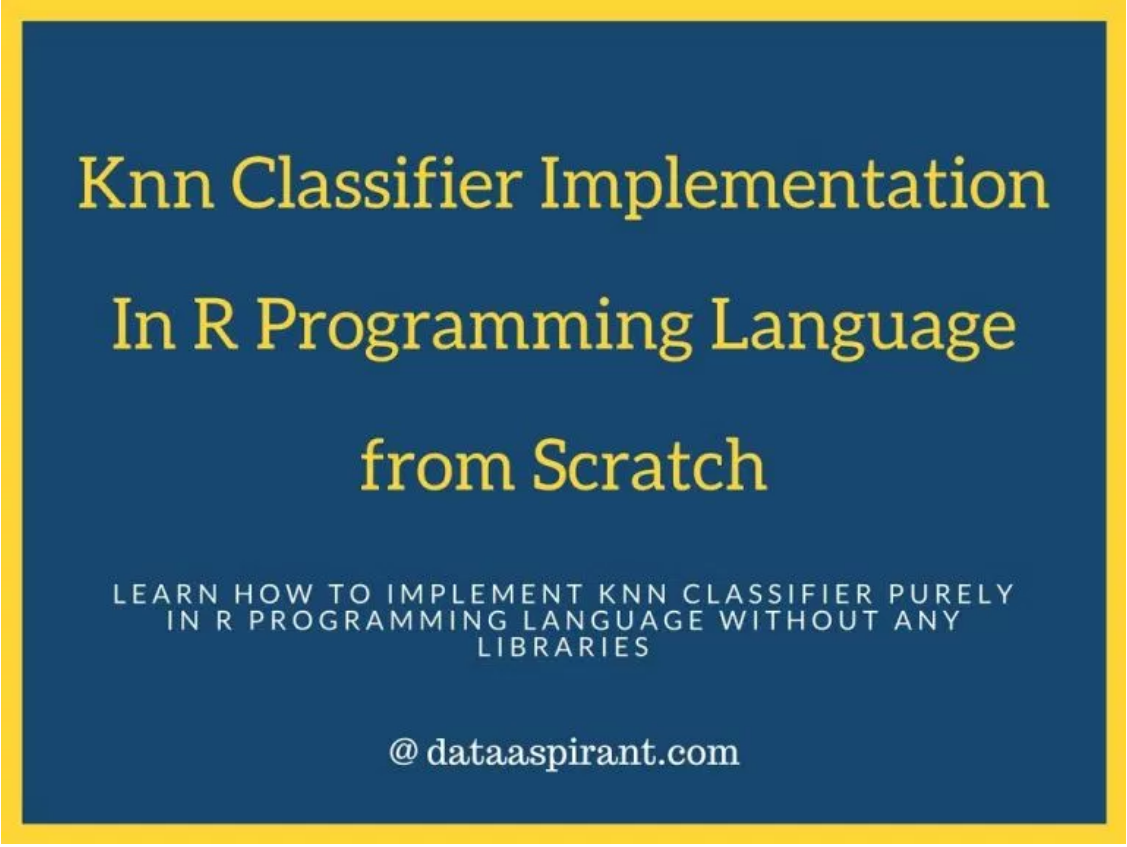
KNN R, K-NEAREST NEIGHBOR CLASSIFIER IMPLEMENTATION IN R PROGRAMMING FROM SCRATCH

January 2, 2017 Rahul Saxena 3 Comments Data Science, Machine Learning, R



knn in implementation r from scratch

K-Nearest neighbor algorithm implement in R Programming from scratch

In the introduction to k-nearest-neighbor algorithm article, we have learned the core concepts of the knn algorithm. Also learned about the applications using knn algorithm to solve the real world problems.

In this post, we will be implementing K-Nearest Neighbor Algorithm on a dummy data set using R programming language from scratch. Along the way, we will implement a prediction model to predict classes for data.

Knn Implementation in R

Why we need to implement knn algorithm from scratch in R Programming Language

Implementation of K-Nearest Neighbor algorithm in R language from scratch will help us to apply the concepts of Knn algorithm. As we are going implement each every component of the knn algorithm and the other components like how to use the datasets and find the accuracy of our implemented model etc.

Problem Set

We will use a sample dataset extracted from ionosphere database by John Hopkins University. We have converted the database into a small dataset so as to simplify the learning curve for our readers.

Our objective is to program a **Knn classifier in R programming language without using any machine learning package**. We have two classes “g”(good) or “b”(bad), it is the response of radar from the ionosphere. The classifier could be capable of predicting **“g”** or **“b”** class for new records from training data.

Ionosphere Dataset Description

This dummy dataset consists of 6 attributes and 30 records. Out Of these 5 attributes are continuous variables with values ranging from -1 to +1 i.e, [-1,+1]. Last(6th) attribute is a categorical variable with values as “g”(good) or “b”(bad) according to the definition summarized above. This is a binary classification task.

K-Nearest Neighbor Algorithm Pseudocode

Let (X_i, C_i) where i = 1, 2……., n be data points. X_i denotes feature values & C_i denotes labels for X_i for each i.

Assuming the number of classes as 'c'
$C_i \in \{1, 2, 3, ……, c\}$ for all values of i

Let x be a point for which label is not known, and we would like to find the label class using k-nearest neighbor algorithms.

Procedure:

1. Calculate "$d(x, x_i)$" i =1, 2, ….., **n**; where **d** denotes the Euclidean distance between the points.
2. Arrange the calculated **n** Euclidean distances in non-decreasing order.
3. Let **k** be a +ve integer, take the first **k** distances from this sorted list.
4. Find those **k**-points corresponding to these **k**-distances.
5. Let $\mathbf{k}_i$ denotes the number of points belonging to the i^{th} class among **k** points i.e. $k \geq 0$
6. If $k_i > k_j \; \forall \; i \neq j$ then put x in class i.

Let's use the above pseudocode for implementing the knn algorithm in R Language.

Prerequisites:

1. Basic programming experience is required
2. Install R-Studio on your system.

K-Nearest neighbor algorithm implement in R Language from scratch

We are going to follow the below workflow for implementing the knn algorithm in R:

1. Getting Data
2. Train & Test Data Split
3. Euclidean Distance Calculation

4. KNN prediction function
5. Accuracy calculation

Let's get our hands dirty and start the coding stuff.

Getting Data in R

For any programmatic implementation on the dataset, we first need to import it. Using read.csv(), we are importing dataset into knn.df dataframe. Since dataset has no header so, we are using header= FALSE. sep parameter is to define the literal which separates values our document. knn.df is a dataframe. A dataframe is a table or 2-D array, in which each column contains measurements on one variable, and each row contains one record.

```
knn.df <- read.csv('i_data_sample_30.csv', header = FALSE, sep = ',')
```

For checking dimensions of the dataset, we can call dim() method and be passing data frame as a parameter.

```
dim(knn.df)
[1] 30  6
```

It shows that the data frame consists of 30 records and 6 columns.

To check summary of our dataset, we can use summary() method.

```
summary(knn.df)

       V1                V2                V3                V4
 Min.   :-1.0000   Min.   :-1.0000   Min.   :-1.0000   Min.   :-1.0000   Min.
 1st Qu.:-0.1079   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu
 Median : 0.0000   Median : 0.8099   Median : 0.4059   Median : 0.5891   Median
 Mean   :-0.0137   Mean   : 0.5611   Mean   : 0.3246   Mean   : 0.2769   Mean
 3rd Qu.: 0.1336   3rd Qu.: 1.0000   3rd Qu.: 0.9730   3rd Qu.: 0.8269   3rd Qu
 Max.   : 1.0000   Max.   : 1.0000   Max.   : 1.0000   Max.   : 1.0000   Max.
 V6
 b:15
 g:15
```

It shows that records with bad class and good class are 15 each.

Train & Test Data split in R

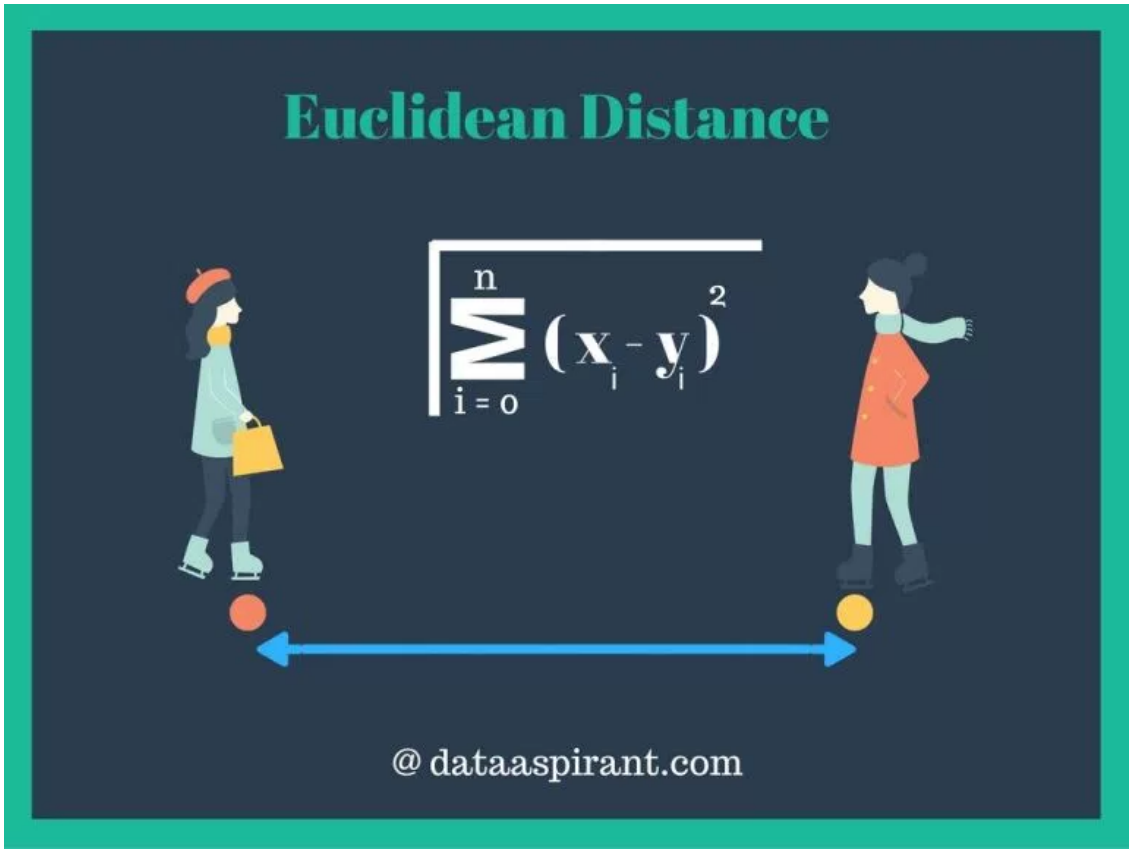
Before Train & Test data split, we need to distribute it randomly. In R, we can use sample() method. It helps to randomize all the records of dataframe. Please use set.seed(2), seed() method is used to produce reproducible results.

In the next line we are passing sample() method inside dataframe. This is to randomize all 30 records of knn.df. Now, we are ready for a split. For dividing train, test data we are splitting them in 70:30 ratio i.e., 70% of data will be considered as train set & 30% as the test set.

```
set.seed(2)
knn.df<- knn.df[sample(nrow(knn.df)),]
train.df <- knn.df[1:as.integer(0.7*30),]
test.df <- knn.df[as.integer(0.7*30 +1):30,]
```

Euclidean Distance Calculation in R

Below snippet consists of a function defined in R to calculate Euclidean distance between 2 points a & b. The formula of Euclidean distance is:



Euclidean Distance

```
euclideanDist <- function(a, b){
  d = 0
  for(i in c(1:(length(a)-1) ))
  {
    d = d + (a[[i]]-b[[i]])^2
  }
  d = sqrt(d)
  return(d)
}
```

KNN prediction function in R

This function is the core part of this tutorial. We are writing a function knn_predict. It takes 3 arguments: test data, train data & value of K. It loops over all the records of test data and train data. It returns the predicted class labels of test data.

```
knn_predict <- function(test_data, train_data, k_value){
  pred <- c()  #empty pred vector
  #LOOP-1
  for(i in c(1:nrow(test_data))){   #looping over each record of test data
    eu_dist =c()          #eu_dist & eu_char empty  vector
    eu_char = c()
    good = 0              #good & bad variable initialization with 0 value
    bad = 0

    #LOOP-2-looping over train data
    for(j in c(1:nrow(train_data))){

      #adding euclidean distance b/w test data point and train data to eu_dist
      eu_dist <- c(eu_dist, euclideanDist(test_data[i,], train_data[j,]))

      #adding class variable of training data in eu_char
      eu_char <- c(eu_char, as.character(train_data[j,][[6]]))
    }

    eu <- data.frame(eu_char, eu_dist) #eu dataframe created with eu_char & eu_

    eu <- eu[order(eu$eu_dist),]       #sorting eu dataframe to gettop K neighb
    eu <- eu[1:k_value,]               #eu dataframe with top K neighbors

    #Loop 3: loops over eu and counts classes of neibhors.
    for(k in c(1:nrow(eu))){
      if(as.character(eu[k,"eu_char"]) == "g"){
        good = good + 1
      }
      else
        bad = bad + 1
    }

    # Compares the no. of neighbors with class label good or bad
    if(good > bad){          #if majority of neighbors are good then put "g" ir

      pred <- c(pred, "g")
    }
    else if(good < bad){
                    #if majority of neighbors are bad then put "b" in pred vecto
      pred <- c(pred, "b")
    }

  }
  return(pred) #return pred vector
}
```

It returns a vector with predicted classes of test dataset. These predictions can be used to calculate accuracy metric.

Accuracy Calculation in R

The accuracy metric calculates the ratio of the number of correctly predicted class labels to the total number of predicted labels.

```
accuracy <- function(test_data){
  correct = 0
  for(i in c(1:nrow(test_data))){
    if(test_data[i,6] == test_data[i,7]){
      correct = correct+1
    }
  }
  accu = correct/nrow(test_data) * 100
  return(accu)
}
```

KNN Algorithm accuracy print: In this code snippet we are joining all our functions. We are calling the knn_predict function with train and test dataframes that we split earlier and K value as 5.
We are appending the prediction vector as the 7th column in our test dataframe and then using accuracy() method we are printing accuracy of our KNN model.

```
K = 5
predictions <- knn_predict(test.df, train.df, K) #calling knn_predict()

test.df[,7] <- predictions #Adding predictions in test data as 7th column
print(accuracy(test.df))
```

Script Output:

```
Accuracy of our KNN model is
77.77778
```

It prints accuracy of our knn model. Here our accuracy is **77.78%.** That's pretty good 🙂 for our randomly selected dummy dataset.
You can download the RMD file of this code from our GitHub repository.

Finally, we have implemented our KNN model in R programming without using any specific R packages.Hope you enjoyed learning it.

Related Articles To Read

- Introduction to knn algorithm
- Knn algorithm implementation purely in python without any machine learning libraries

- Cancer tumor detection with knn sklearn

Follow us:

FACEBOOK| QUORA |TWITTER| GOOGLE+ | LINKEDIN| REDDIT | FLIPBOARD | MEDIUM | GITHUB

I hope you like this post. If you have any questions, then feel free to comment below. If you want me to write on one particular topic, then do tell it to me in the comments below.

Related Courses:

Do check out unlimited data science courses

Title of the course	Course Link	Course Link
R Programming A-Z: R For Data Science With Real Exercises!	R Programming A-Z: R For Data Science With Real Exercises!	• This cou step-by- new tut build or already move o forward • After ev learn a concept apply ri the best you lear exampl • In sumr course l designe levels a

		have nc progran statistic you will in this c
R Programming: Advanced Analytics In R For Data Science	R Programming: Advanced Analytics In R For Data Science	• Perform Prepara locate n datafra • Apply th Analysis replace records • Work w and sub for repl • Use lap sapply() working vectors. • Use lap sapply() working vectors.
Data Mining with R: Go from Beginner to Advanced!	Data Mining with R: Go from Beginner to Advanced!	• Use R sc data im export, explora visualiz data an

includin
a compr
of data
operatic

- Apply th
include
cases ar
using re
scripts t
unique
and dat
problen

- Effective
number
contem
mining
techniq
demanc
includin
Decisior
classific
regressi
(CART);
forests;
logistic
and (4) V
cluster a
techniq

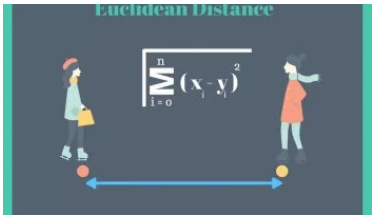
Share this:

12 58

Related

[K-nearest neighbor algorithm implementation in Python from scratch]
December 27, 2016
In "Machine Learning"

[KNN R, K-Nearest Neighbor implementation in R using caret package]
January 9, 2017
In "Machine Learning"



[Knn sklearn, K-Nearest Neighbor implementation with scikit learn]
December 30, 2016
In "Data Science"

Classification k-nearest neighbor R Programming