K Nearest Neighbours Classifier

Learn how a Simple Lazy Learning Algorithm Works

Contents

Introduction to KNN

Machine Learning Algorithms

Eager Learners	Lazy Learners
Learn a model that maps relationship between the predictors and response variable and then give a decision.	Base their decisions simply on the patterns found in training data. Generalisation beyond training data is delayed until a query is made.

- K Nearest Neighbours is one of the simplest lazy learner algorithms.
- The algorithm can be used for both classification and regression problems.
- Conceptually simple yet capable of solving complex problems.

KNN stores all available cases and classifies (or gives expected value of) new cases based on a similarity measure

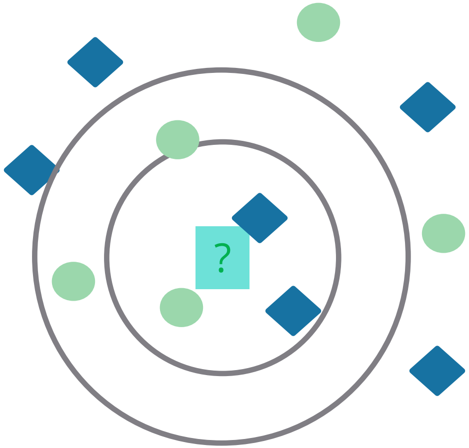
KNN for Classification

- Training dataset has 11 observations belonging to two categories.
- 12th observation is introduced, class of which is not known.
- **Nearest neighbour algorithm classifies new observation to the class of the training observation closest to it.**

When K=1, nearest one case is considered

As we go on increasing K, classification may vary

K	Classification
1	Blue
3	Blue
5	Orange



Three most important components of this method are **Distance between cases**, **Value of K** and **Voting** criteria.

Measuring Distance

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$D_1 = \sqrt{(25 - 48)^2 + (40000 - 142000)^2}$$

$$D_1 = 102000$$

Age	Current Debt	Default	Distance
25	40,000	N	102000
35	60,000	N	82000
45	80,000	N	62000
20	20,000	N	122000
35	120,000	N	22000
52	18,000	N	124000
23	95,000	Y	47000
40	62,000	Y	80000
60	100,000	Y	42000
48	220,000	Y	78000
33	150,000	Y	8000
48	**142,000**	**?**	

Here k=1, so the least distance from New observation is 8000, so it will be classified as "Y".

Measuring Distance

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Euclidean Distance

Age	Current Debt	Default
25	40,000	N
35	60,000	N
45	80,000	N
20	20,000	N
35	120,000	N
52	18,000	N
23	95,000	Y
40	62,000	Y
60	100,000	Y
48	220,000	Y
33	150,000	Y
48	**142,000**	**Y**

Distance
102000
82000
62000
122000
22000
124000
47000
80000
42000
78000
8000

However, we can see that both these **attributes** are of different scales.

So rescaling of variables before calculating distances is the preferred approach.

Distance Based on Standardised Variables

$$X_s = \frac{X - \text{Min}}{\text{Max} - \text{Min}}$$

Alternatively, $\frac{(X-\text{Mean})}{SD}$ can also be used

Age	Current Debt	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	0.61	?	

New observation will be classified as “N”

Selection of K

The second component of KNN model is selecting the appropriate value for K

- If K = 1, the case is classified using the nearest neighbour
- However, K is usually greater than 1. Consider the following when choosing K :
  - Mostly odd numbered K is preferred to avoid tie.
  - For a very large K the classifier may result in misclassification, as group of nearest neighbours may include data points which are actually located far away from it.

Thumb Rule :

K = sqrt (n)

n is the number of observations in training data

Voting Criteria

Most common criteria for classification decision is **Majority Voting.**

Frequency of each class in K instances is measured. Class having the highest frequency is attributed to the new case.

Eg. Suppose for K = 7, 4 cases belong to class A and 3 to class B. New case is given class A

Drawback:

Classification is inappropriate when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number.

?

Is there a way to correct this?
One option to remove this drawback is to create training data with equal class frequency. However, this is possible only if data is very large.

Voting Criteria

Another approach is **Inverse Distance Weighted Voting**

This approach assigns higher weights to closer neighbours. Votes are then summed and class with the highest votes is assigned to the new case.

Eg. Suppose for K = 3, 2 cases belong to class A and 1 to class B. with distances 0.4, 0.5 and 0.2 respectively. Sum of inverse distances are

Class A (1/0.4)+(1/0.5) = 4.5 and Class B (1/0.2)=5

This rule will allot class A to the new observation

Some studies also recommend inverse of squared distances or Kernel functions

Handling Ties

KNN may result in **'Ties' – nearest neighbours may have equal class frequencies or equal inverse distance sums**

Such ties are solved by either of the following ways:

- In case of binary class variables, avoid using even numbered Ks
- Increasing or decreasing K until ties are broken

Case Study – Predicting Loan Defaulters

Background

- The bank possesses demographic and transactional data of its loan customers. If the bank has a robust model to predict defaulters it can undertake better resource allocation.

Objective

- To predict whether the customer applying for the loan will be a defaulter

Available Information

- Sample size is 389
- Age group, Years at current address, Years at current employer, Debt to Income Ratio, Credit Card Debts, Other Debts are the independent variables
- Defaulter (=1 if defaulter, 0 otherwise) is the dependent variable

Data Snapshot

BANK LOAN KNN

Variables

SN	AGE	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTER

Column	Description	Type	Measurement	Possible Values
SN	Serial Number		-	-
AGE	Age Groups	Categorical	1(<28 years),2(28-40 years),3(>40 years)	3
EMPLOY	Number of years customer working at current employer	Continuous	-	Positive value
ADDRESS	Number of years customer staying at current address	Continuous	-	Positive value
DEBTINC	Debt to Income Ratio	Continuous	-	Positive value
CREDDEBT	Credit Card Debt	Continuous	-	Positive value
OTHDEBT	Other Debt	Continuous	-	Positive value
DEFAULTER	Whether customer defaulted on loan	Binary	1(Defaulter), 0(Non-Defaulter)	2

KNN Classification in R

```
# Importing the Data
bankloan<-read.csv("BANK LOAN KNN.csv",header=T)

# Preparing data by removing unwanted variables
bankloan2<-subset(bankloan,select=c(-AGE,-SN,-DEFAULTER))
```

subset() is used to remove unwanted variables. AGE is removed because it is a categorical variable.

```
head(bankloan2)
```

```
# Output
> head(bankloan2)
  EMPLOY ADDRESS DEBTINC CREDDEBT OTHDEBT
1     17      12     9.3    11.36    5.01
2      2       0    17.3     1.79    3.06
3     12      11     3.6     0.13    1.24
4      3       4    24.4     1.36    3.28
5     24      14    10.0     3.93    2.47
6      6       9    16.3     1.72    3.01
```

KNN Classification in R

```
# Preparing Variables
bankloan3<-scale(bankloan2)

head(bankloan3)
```

scale() in base R is a generic function used for centering or scaling columns of a numeric matrix. The default method for scaling is (X-Mean)/SD.

```
# Output
> head(bankloan3)
      EMPLOY    ADDRESS   DEBTINC     CREDDEBT     OTHDEBT
1  1.5656796  0.6216799 -0.2881684  3.8774339687  0.51519694
2 -0.8239988 -1.1852951  0.7889154  0.0289356115 -0.02571385
3  0.7691201  0.4710987 -1.0555906 -0.6386200074 -0.53056393
4 -0.6646869 -0.5829701  1.7448273 -0.1439854223  0.03531198
5  2.6808628  0.9228424 -0.1939235  0.8895193612 -0.18937404
6 -0.1867512  0.1699362  0.6542799  0.0007856758 -0.03958336
```

- All the continuous predictors are now scaled

KNN Classification in R

```
# Training and Testing Data Sets
```

```
install.packages("caret")
library(caret)

index<-createDataPartition(bankloan$SN,p=0.7,list=FALSE)
head(index)
```

```
# Output
```

```
> head(index)
     Resample1
[1,]         1
[2,]         3
[3,]         4
[4,]         5
[5,]         7
[6,]         8
```

```
traindata<-bankloan3[index,]
testdata<-bankloan3[-index,]

dim(traindata)
[1] 273 5
dim(testdata)
[1] 116 5
```

KNN Classification in R

```
# Creating Class Vectors
Ytrain<-bankloan$DEFAULTER[index]

Ytest<-bankloan$DEFAULTER[-index]
```

Interpretation :

- Training and testing datasets are created with a 70:30 division.
- Class variable is also split by the same proportion.

KNN Classification Using Package "class"

```
# KNN Using Package "class"
install.packages("class")
library(class)
# KNN Classification (Continuous Predictors)
model<-knn(traindata,testdata,k=20,cl=Ytrain)
```

- **knn()** in package **"class"** performs k-nearest neighbour classification of test data using train data. Distance is calculated by Euclidean measure, and the classification is decided by majority vote, with ties broken at random.
- **k=** specifies the value of k.
- **cl=** is the vector of observed Y values

KNN Classification Using Package "class"

```
table(Ytest,model)
      model
Ytest   0  1
    0 45 20
    1 13 38
```

table() gives the cross tabulation of actual and predicted classes for test data

```
# Confusion Matrix
class(model)
[1] "factor"

class(Ytest)
[1] "integer"

Ytest <- as.factor(Ytest)

library(caret)
confusionMatrix(Ytest,model)
```

*

Note : Output will be slightly different as observations are randomly assigned to train-test data.

KNN Classification in R

Output

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 45 20
         1 13 38

               Accuracy : 0.7155
                 95% CI : (0.6243, 0.7954)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.933e-06

                  Kappa : 0.431

 Mcnemar's Test P-Value : 0.2963

            Sensitivity : 0.7759
            Specificity : 0.6552
         Pos Pred Value : 0.6923
         Neg Pred Value : 0.7451
             Prevalence : 0.5000
         Detection Rate : 0.3879
   Detection Prevalence : 0.5603
      Balanced Accuracy : 0.7155

       'Positive' Class : 0
```

Interpretation :

- From the confusion matrix we can see that sensitivity is 77.6% & specificity 65.5%.

KNN for Regression

KNN algorithm can also be extended to regression problems, i.e. **when the dependent variable is continuous**

Process flow for classification and regression is the same, except for the last step

Standardise Variables → Calculate Distances → Choose Appropriate k → **Average the Response Variable for k Neighbours**

Average value of the response variable for k neighbours is calculated and assigned to the new case.

`knn.reg()` from package `"FNN"` can be used to run k-nearest neighbour regression in R, using the syntax : `knn.reg(train, test, y, k)`

`y` is the response for each observation in training set

Get an Edge!

- KNN can be used for categorical variables as well.
- Before executing knn on train-test data, categorical variables have to be converted to continuous variables by creating dummy variables.

Quick Recap

KNN for Classification	• Three most important components of this method are **Distance** between cases, **Value of K** and **Voting** criteria.
KNN for Classification in R	• **knn()** in package **class.**
KNN for Regression	• KNN algorithm can also be extended to regression problems **when the dependent variable is continuous.**
KNN for Regression in R	• **`knn.reg()`** from package **`FNN`**