Machine Learning

2022-12-22

K -Nearest Neighbor Prediction

The linear regression makes huge assumptions about the structure of data and yields stable but possibly inaccurate predictions. On the other hand, the method of **K - Nearest Neighbor** makes very mild structural assumptions and yields to accurate but unstable ˆ[Stable predictions means that when the data is changed a little, the predictions change a little. With the linear model all of the data contributes to the predictions for any particular x. This gives low variance, hence its predictions can be **stable** — but high bias if the linear model is not a good approximation. With the k-nearest neighbors method, for any particular x only the k-nearest neighbors contribute to the prediction. As a result the variance is higher, hence its predictions can be **unstable** — this is especially so if kk is much smaller than n. The gain is that by only using close points, the approximation could be better, so potentially lower bias.] predictions.

The parameter K in KNN refers to the number of labelled points (neighbors) considered for classification. The value of k indicates the number of pairs used to determine the result.

Given a data point whose class we don't know, we can try to understand which points in our feature space are closest to it. These pairs are known as the k-nearest neighbors. It is very likely that the point belong to the same class as its neighbors. It is usually known as lazy learner algorithm because it only stores a training dataset versus undergoing a training stage. It means that computations only occur when classification or prediction is being made. The algorithm is as follows;

step 1: Select the number of k-neighbors;

step 2: Calculate the **Euclidean distance** (p = 2) between the point and the k - nearest neighbors.

The Euclidean distance between point A and point B is calculated as;

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

More Generally;

$$d(x_i, y_i) = \sqrt{\sum (y_i - x_i)}$$

This is basically the distance between the query point and the other data points. We the nearest neighbors to determine the group of the data point. Suppose another unclassified data point Y is placed between group A and group B. If K is equal to 10, we pick the group that gets the most votes, meaning that we classify Y to the group in which it has the most number of neighbors. For example, if Y has seven neighbors in group B and three neighbors in group A, it belongs to group B.

Consider an example where we have data from the questionnaire survey asking people's opinion about a tissue paper based on two attributes; acid durability and strength. The people classify the tissue paper as good or bad. Here is four training samples

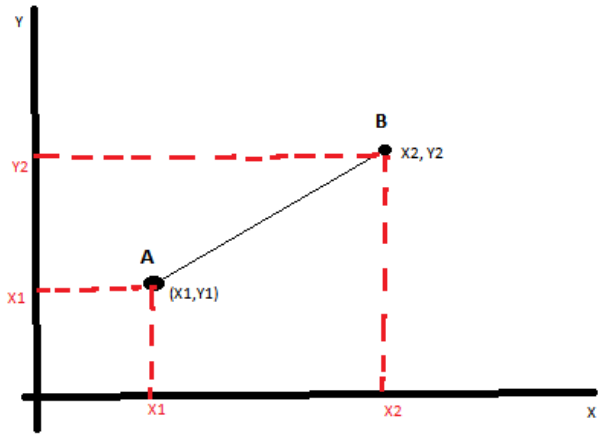


Figure 1: Euclidean distance between two points

X1 **(Acid Durability)**	x2 (Strength)	**Y** **(Classification)**
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that pass laboratory test with x1 = 3 and x2 = 7./ Without, an expensive survey, can we guess the classification of the new tissue? We first select the parameter K. Suppose that $K = 3$. The next step is to calculate the Euclidean distance between the query instance and all the training samples as follows;

X1 (Acid Durability)	x2 (Strength)	Distance
7	7	$(7-3)^2 + (7-7)^2 = \sqrt{16}$
7	4	$(7-3)^2 + (4-7)^2 = \sqrt{25}$
3	4	$(3-3)^2 + (4-7)^2 = \sqrt{9}$
1	4	$(1-3)^@ + (4-7)^2 = \sqrt{13}$

The next step is to sort the distance to determine the nearest neighbor

X1 (Acid Durability)	x2 (Strength)	Distance	Rank	Neighbors?
7	7	$(7-3)^2+(7-7)^2 = \sqrt{16}$	3	Yes
7	4	$(7-3)^2+(4-7)^2 = \sqrt{25}$	4	No

X1 (Acid Durability)	x2 (Strength)	Distance	Rank	Neighbors?
3	4	$(3-3)^2+(4-7)^2=\sqrt{9}$	1	Yes
1	4	$(1-3)^{@}+(4-7)^2=\sqrt{13}$	2	Yes

The next step is to gather the category of the nearest neighbors;

X1 (Acid Durability)	x2 (Strength)	D istance	Rank	Neighbors?	Y (Category)
7	7	4	3	Yes	Bad
7	4	5	4	No	-
3	4	3	1	Yes	Good
1	4	3.60	2	Yes	Good

Using the simple majority of the category of the nearest neighbor as the prediction value of the query instance, we have 2 good and 1 bad, and since 2>1, then we conclude that the new paper tissue that has the laboratory pass of x1= 3 and x2 = 7 is included in the good category.

Suppose that we wish to determine whether a person has diabetes given some factors. We can do the knn in R as follows;

```
#Choosing
set.seed(123)
split_s <- .8
split_size3 <- floor(split_s * nrow(pima))
indicies3 <- sample(seq_len(nrow(pima)), size = split_size3)

#Removing the Outcome variable since it is our target
pima.subset <- pima[,1:8]

trainpima <- pima.subset[indicies3,]
testpima <- pima.subset[-indicies3,]
testpima <- testpima[, -9]
```

After creating the training and testing data sets, we now create a new data set for the target variable *outcome* as follows;

```
set.seed(123)
train.pima_labels <- pima[indicies3,9]
test.pima_labels <- pima[-indicies3, 9]
```

We then find the number of observations so that we can be able to calculate the parameter k as the square root of the total number of observations in the train data set of the outcome variable as follows;

```
NROW(train.pima_labels)
```

```
[1] 601
```

The square root of 601 is 24.5, thus the parameter k will be either 24 or 25 which implies that we shall have two models done as follows;

```
library(class)
set.seed(123)
knn.24 <- knn(train = trainpima, test = testpima, cl = train.pima_labels, k = 24)
```

After building the model its time to calculate the model accuracy, by calculating the confusion matrix from the infamous Caret package as follows;

```
library(caret)
set.seed(123)
confusionMatrix(table(knn.24, test.pima_labels))
```

```
Confusion Matrix and Statistics

             test.pima_labels
knn.24        Diabetic NotDiabetic
  Diabetic          84          29
  NotDiabetic       12          26

               Accuracy : 0.7285
                 95% CI : (0.6502, 0.7976)
    No Information Rate : 0.6358
    P-Value [Acc > NIR] : 0.01009

                  Kappa : 0.3723

 Mcnemar's Test P-Value : 0.01246

            Sensitivity : 0.8750
            Specificity : 0.4727
         Pos Pred Value : 0.7434
         Neg Pred Value : 0.6842
             Prevalence : 0.6358
         Detection Rate : 0.5563
   Detection Prevalence : 0.7483
      Balanced Accuracy : 0.6739

       'Positive' Class : Diabetic
```

For the k = 25 model;

```
set.seed(123)
library(caret);library(class)
knn.25 <- knn(train = trainpima, test = testpima, cl = train.pima_labels, k = 25)

confusionMatrix(table(knn.25, test.pima_labels))
```

```
Confusion Matrix and Statistics

             test.pima_labels
knn.25        Diabetic NotDiabetic
  Diabetic          82          30
```

```
  NotDiabetic       14          25

               Accuracy : 0.7086
                 95% CI : (0.6292, 0.7796)
    No Information Rate : 0.6358
    P-Value [Acc > NIR] : 0.03631

                  Kappa : 0.3292

 Mcnemar's Test P-Value : 0.02374

            Sensitivity : 0.8542
            Specificity : 0.4545
         Pos Pred Value : 0.7321
         Neg Pred Value : 0.6410
             Prevalence : 0.6358
         Detection Rate : 0.5430
   Detection Prevalence : 0.7417
      Balanced Accuracy : 0.6544

       'Positive' Class : Diabetic
```

The best model is the one with the parameter k as 24, and it's accuracy is 72.85% which a good accuracy We can also write a function to look up for the value of K which yields to the highest accuracy as follows;

```
set.seed(123)
knnParameter <- function()
{
  library(class)
  set.seed(123)
  vector <- as.vector(1:28)
  i = 1
  knn.accuracy = 1
  for (i in vector )
    {
      knn.mod <- knn(train = trainpima, test = testpima, cl = train.pima_labels, k = i)
      knn.accuracy[i] <- 100 * sum(test.pima_labels == knn.mod)/NROW(test.pima_labels)
      k = i
      cat(k,'=',knn.accuracy[i],'
')
  }
}

knnParameter()
```

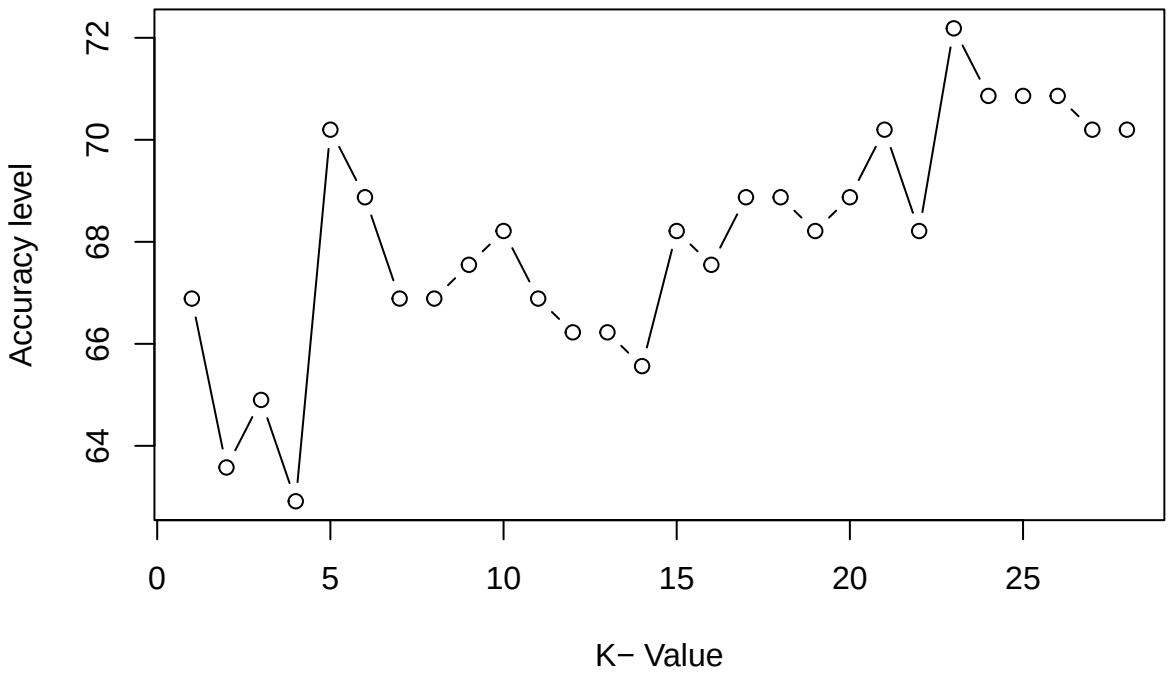
```
1 = 66.88742
2 = 63.57616
3 = 64.90066
4 = 62.91391
5 = 70.19868
6 = 68.87417
7 = 66.88742
```

```
8 = 66.88742
9 = 67.54967
10 = 68.21192
11 = 66.88742
12 = 66.22517
13 = 66.22517
14 = 65.56291
15 = 68.21192
16 = 67.54967
17 = 68.87417
18 = 68.87417
19 = 68.21192
20 = 68.87417
21 = 70.19868
22 = 68.21192
23 = 72.18543
24 = 70.86093
25 = 70.86093
26 = 70.86093
27 = 70.19868
28 = 70.19868
```

Thus the parameter K that yields the highest accuracy is $k = 23$, which can also be seen from the graph below

```
knnParameter <- function()
{
  library(class)
  set.seed(123)
  vector <- as.vector(1:28)
  i = 1
  knn.accuracy = 1
  for (i in 1:28 )
    {
      knn.mod <- knn(train = trainpima, test = testpima, cl = train.pima_labels, k = i)
      knn.accuracy[i] <- 100 * sum(test.pima_labels==knn.mod)/NROW(test.pima_labels)
  }
  plot(knn.accuracy, type="b", xlab="K- Value",ylab="Accuracy level")
}

knnParameter()
```



Thus the best model is written as follows;

```
library(class)
set.seed(123)
knn.23 <- knn(train = trainpima, test = testpima, cl = train.pima_labels, k = 23)
```

The confusion matrix is;

```
set.seed(123)
confusionMatrix(table(knn.23, test.pima_labels))
```

```
Confusion Matrix and Statistics

             test.pima_labels
knn.23        Diabetic NotDiabetic
  Diabetic          84          30
  NotDiabetic       12          25

               Accuracy : 0.7219
                 95% CI : (0.6432, 0.7916)
    No Information Rate : 0.6358
    P-Value [Acc > NIR] : 0.015923

                  Kappa : 0.3543
```

```
 Mcnemar's Test P-Value : 0.008712

            Sensitivity : 0.8750
            Specificity : 0.4545
         Pos Pred Value : 0.7368
         Neg Pred Value : 0.6757
             Prevalence : 0.6358
         Detection Rate : 0.5563
   Detection Prevalence : 0.7550
      Balanced Accuracy : 0.6648

       'Positive' Class : Diabetic
```

Thus the accuracy is 72.19%.