**What is knn algorithm?**

K Nearest Neighbour is a supervised learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points.

Let’s look at the student dataset with GPA and GRE scores for classification problems and Boston housing data for a regression problem.

The euclidian distance used for calculating the distance between k neighbors and some of the variables have different magnitudes, so standazation is important.

Some of the popular application examples are

* Recommendation system
* Loan Approval
* Anamoly Detection
* Text Categorization
* Finance
* Medicine

Let’s see how do we apply knn algorithm in classification and regression.

**Classification Approach**

**Load Libraries**

library(caret)

library(pROC)

library(mlbench)

**Getting Data**

data <- read.csv("D:/RStudio/knn/binary.csv", header = T)

str(data)

You can access the dataset from this [link](https://github.com/finnstats/finnstats/blob/main/binary.csv)

'data.frame': 400 obs. of  4 variables:

 $ admit: int  0 1 1 1 0 1 1 0 1 0 ...

 $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...

 $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...

 $ rank : int  3 3 1 4 4 2 1 2 3 2 ...

The data frame contains 400 observations and 4 variables and rank variables stored as integer currently need to convert into factor variable. Admit is the response variable or dependent variable let’s recode 0 and 1 into No and Yes.

data$admit[data$admit == 0] <- 'No'

data$admit[data$admit == 1] <- 'Yes'

data$admit <- factor(data$admit)

**Data Partition**

Let’s create independent samples and create training and test dataset for prediction.

set.seed(1234)

ind <- sample(2, nrow(data), replace = T, prob = c(0.7, 0.3))

training <- data[ind == 1,]

test <- data[ind == 2,]

str(training)

'data.frame': 284 obs. of  4 variables:

 $ admit: Factor w/ 2 levels "No","Yes": 1 2 2 2 2 2 1 2 1 1 ...

 $ gre  : int  380 660 800 640 760 560 400 540 700 800 ...

 $ gpa  : num  3.61 3.67 4 3.19 3 2.98 3.08 3.39 3.92 4 ...

 $ rank : int  3 3 1 4 2 1 2 3 2 4 ...

The training dataset contains now 284 observations with 4 variables and the test dataset contains 116 observations and 4 variables.

**KNN Model**

Before making knn model we need to create train control. Let’s create train control based on below code.

trControl <- trainControl(method = "repeatedcv",

                          number = 10,

                          repeats = 3,

                          classProbs = TRUE,

                          summaryFunction = twoClassSummary)

trainControl is from caret package

number of iteration is 10 times.

Repeat the cross validation is 3 times.

set.seed(222)

fit <- train(admit ~ .,

             data = training,

             method = 'knn',

             tuneLength = 20,

             trControl = trControl,

             preProc = c("center", "scale"),

             metric = "ROC",

             tuneGrid = expand.grid(k = 1:60))

**Model Performance**

fit

k-Nearest Neighbors

284 samples

  3 predictor

  2 classes: 'No', 'Yes'

Pre-processing: centered (3), scaled (3)

Resampling: Cross-Validated (10 fold, repeated 3 times)

Summary of sample sizes: 256, 256, 256, 256, 255, 256, ...

Resampling results across tuning parameters:

  k   ROC   Sens  Spec

   1  0.54  0.71  0.370

   2  0.56  0.70  0.357

   3  0.58  0.80  0.341

   4  0.56  0.78  0.261

   5  0.59  0.81  0.285

   6  0.59  0.82  0.277

   7  0.59  0.86  0.283

   8  0.59  0.86  0.269

   9  0.60  0.87  0.291

  10  0.59  0.87  0.274

  11  0.60  0.88  0.286

  12  0.59  0.87  0.277

  13  0.59  0.87  0.242

  14  0.59  0.89  0.257

  15  0.60  0.88  0.228

  16  0.61  0.90  0.221

  17  0.63  0.90  0.236

  18  0.63  0.90  0.215

  19  0.63  0.90  0.229

  20  0.64  0.90  0.222

  21  0.64  0.91  0.211

  22  0.64  0.91  0.225

  23  0.64  0.92  0.214

  24  0.64  0.93  0.217

  25  0.65  0.92  0.200

  26  0.65  0.93  0.199

  27  0.66  0.93  0.203

  28  0.66  0.94  0.210

  29  0.67  0.94  0.199

  30  0.67  0.94  0.199

.....................

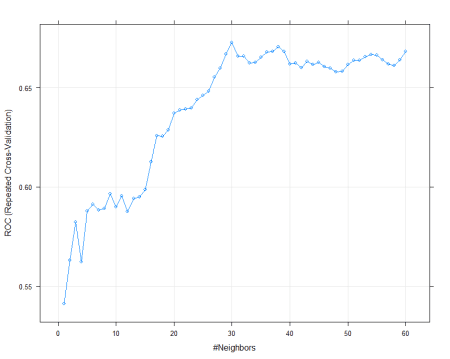
59  0.66  0.96  0.096

  60  0.67  0.96  0.100

ROC was used to select the optimal model using the largest value.

The final value used for the model was k = 30.

We have carried out 10 cross-validations and the best ROC we got at k=30

plot(fit) 

varImp(fit)

ROC curve variable importance

     Importance

gpa       100.0

rank       25.2

gre         0.0

gpa is more important followed by rank and gre is not important.

pred <- predict(fit, newdata = test)

confusionMatrix(pred, test$admit)

Confusion Matrix and Statistics

          Reference

Prediction No Yes

       No  79  29

       Yes  3   5

               Accuracy : 0.724

                 95% CI : (0.633, 0.803)

    No Information Rate : 0.707

    P-Value [Acc > NIR] : 0.385

                  Kappa : 0.142

 Mcnemar's Test P-Value : 9.9e-06

            Sensitivity : 0.963

            Specificity : 0.147

         Pos Pred Value : 0.731

         Neg Pred Value : 0.625

             Prevalence : 0.707

         Detection Rate : 0.681

   Detection Prevalence : 0.931

      Balanced Accuracy : 0.555

       'Positive' Class : No

Model accuracy is 72% with 84 correct classifications out of 116 classifications.

**Regression**

Let’s look at the Bostonhousing data

data("BostonHousing")

data <- BostonHousing)

str(data)

'data.frame': 506 obs. of  14 variables:

 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...

 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...

 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...

 $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...

 $ rm     : num  6.58 6.42 7.18 7 7.15 ...

 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...

 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...

 $ rad    : num  1 2 2 3 3 3 5 5 5 5 ...

 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...

 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...

 $ b      : num  397 397 393 395 397 ...

 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...

 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

Medv is the response or dependent variable with numeric values. The data frame contains a total of 506 observations and 14 variables.

**Data Partition**

Let’s do the data partition for prediction.

set.seed(1234)

ind <- sample(2, nrow(data), replace = T, prob = c(0.7, 0.3))

training <- data[ind == 1,]

test <- data[ind == 2,]

**KNN Model**

trControl <- trainControl(method = 'repeatedcv',

                          number = 10,

                        repeats = 3)

set.seed(333)

Let’s fit the regression model

fit <- train(medv ~.,

             data = training,

             tuneGrid = expand.grid(k=1:70),

             method = 'knn',

             metric = 'Rsquared',

             trControl = trControl,

             preProc = c('center', 'scale'))

**Model Performance**

fit

k-Nearest Neighbors

355 samples

 13 predictor

Pre-processing: centered (13), scaled (13)

Resampling: Cross-Validated (10 fold, repeated 3 times)

Summary of sample sizes: 320, 320, 319, 320, 319, 319, ...

Resampling results across tuning parameters:

  k   RMSE  Rsquared  MAE

   1  4.2   0.78      2.8

   2  4.0   0.81      2.7

   3  4.0   0.82      2.6

   4  4.1   0.81      2.7

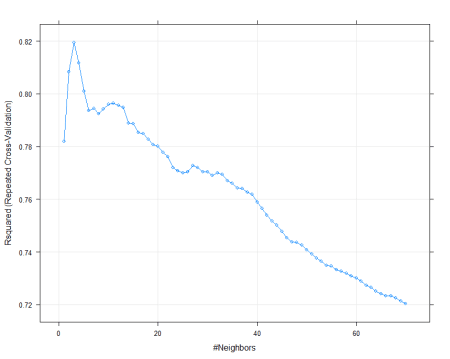
......................

  70  5.9   0.72      4.1

R-squared was used to select the optimal model using the largest value.

The final value used for the model was k = 3.

This model is based on 10 fold cross-validation with 3 repeats.

plot(fit) 

varImp(fit)

loess r-squared variable importance

        Overall

rm        100.0

lstat      98.0

indus      87.1

nox        82.3

tax        68.4

ptratio    50.8

rad        41.3

dis        41.2

zn         37.9

crim       34.5

b          24.2

age        22.4

chas        0.0

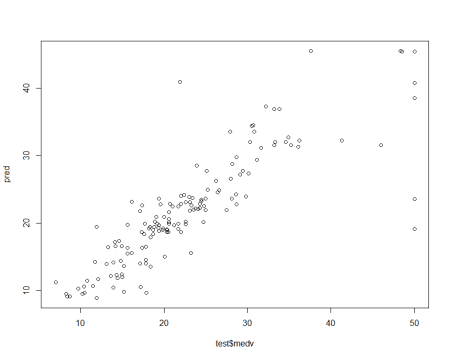
rm is the most important variable and followed by lstat, indus, nox etc..

pred <- predict(fit, newdata = test)

RMSE(pred, test$medv)

6.1

plot(pred ~ test$medv)



**Conclusion**

Based on the knn machine algorithm we can make insights for classification and regression problems.