**Predictive Analytics and Data Mining**

**Peer-graded Assignment: Module 3 Peer Review Assignment**

**Aya Anisa Dwinidasari – February 12nd, 2021**

* **KNN METHODS**

> X\_train = subset(X.norm, sample==TRUE) # input training

> X\_test = subset(X.norm, sample==FALSE) # input prediction accuracy

> y\_train = subset(y,sample==TRUE) # training

> y\_test = subset(y, sample==FALSE)# prediction accuracy

> # k =3

> nn <- knn(train = X\_train, test=X\_test, cl = y\_train, k=3)

> summary(nn)

High Low Medium

875 400 1125

> confusionMatrix(nn, y\_test)$overall[1]

Accuracy

0.7191667

> confusionMatrix(nn, y\_test) # evaluate other metrics

Confusion Matrix and Statistics

Reference

Prediction High Low Medium

High 654 68 153

Low 61 228 111

Medium 157 124 844

Overall Statistics

Accuracy : 0.7192

95% CI : (0.7007, 0.7371)

No Information Rate : 0.4617

P-Value [Acc > NIR] : <2e-16

Kappa : 0.5485

Mcnemar's Test P-Value : 0.7649

Statistics by Class:

Class: High Class: Low Class: Medium

Sensitivity 0.7500 0.5429 0.7617

Specificity 0.8554 0.9131 0.7825

Pos Pred Value 0.7474 0.5700 0.7502

Neg Pred Value 0.8570 0.9040 0.7929

Prevalence 0.3633 0.1750 0.4617

Detection Rate 0.2725 0.0950 0.3517

Detection Prevalence 0.3646 0.1667 0.4688

Balanced Accuracy 0.8027 0.7280 0.7721

> # define a dataframe to save accuracy for different values of K

> accuracy.df <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))

> accuracy.df

k accuracy

1 1 0

2 2 0

3 3 0

4 4 0

5 5 0

6 6 0

7 7 0

8 8 0

9 9 0

10 10 0

11 11 0

12 12 0

13 13 0

14 14 0

15 15 0

16 16 0

17 17 0

18 18 0

19 19 0

20 20 0

> # compute knn for different k on validation by looping

> for(i in 1:20) { # we will loop through K= 1 to 20

+ knn.pred <- knn(train = X\_train, test=X\_test, cl = y\_train, k = i)

+ accuracy.df[i, 2] <- confusionMatrix(knn.pred, y\_test)$overall[1]

+ }

> plot(accuracy.df) # plot accuracy for different values of K

> lines(accuracy.df)

> which.max(accuracy.df$accuracy) # optimal K

[1] 15

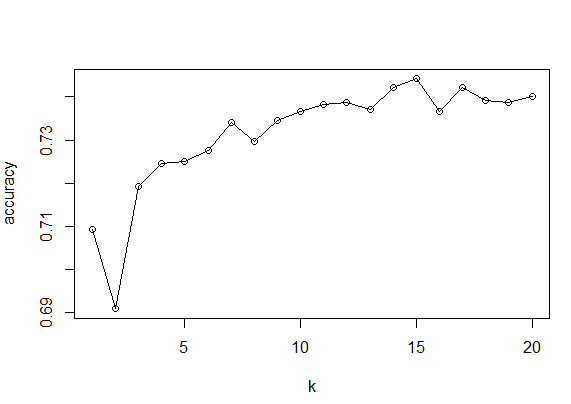
> # K = 20

> # Evaluate model at K =20

> accuracy.df[20,]

k accuracy

20 20 0.74



From the itieration and plot result it can be concluded that as K increases variance decreases and K increases bias increases.

* Naïve Bayes

if (!require("pacman")) install.packages("pacman")

> p\_load(rsample, dplyr, caTools, caret, e1071)

> # Read the data file

> real\_es <- read.csv("realEstate.csv")

> # Define input variables

> X = real\_es[,2:9]

> # Define target variable

> y = real\_es[,10]

> colnames(X)

[1] "full\_sq" "life\_sq" "floor" "max\_floor" "material" "build\_year"

[7] "num\_room" "kitch\_sq"

> # normalizing all input variables

> # Normalize the inputs

> norm.values <- preProcess(X, method=c("center", "scale"))

> X.norm <- predict(norm.values, X) # Normalized input

> # create a function that takes a variable and returns the same variables as "low", "med", "high"

> binning\_func <- function(x){

+ binx <- cut(x, c((min(x)), (mean(x)-1\*sd(x)), (mean(x)+1\*sd(x)), max(x)), labels=c("Low", "Med", "High"))

+ return(binx)

+ }

> # applied the function to all the columns in the X.norm dataframe and saved all of them in X\_binned dataframe

> X\_binned <- data.frame(apply(X.norm, 2, binning\_func))

> df <- cbind(X\_binned, y)

> set.seed(101) # random seed for replicating

> # train test split

> sample = sample.split(df$y, SplitRatio = 0.80)# random sample of 80%

> train <- df[sample==TRUE,] # training dataset

> test <- df[sample==FALSE,] # testing dataset

> # Perform Naive Bayes on 8 predictors

> Naive\_Bayes\_Model=naiveBayes(y ~ ., data=train)

> Naive\_Bayes\_Model$tables

$full\_sq

full\_sq

Y High Low Med

High 0.320711417 0.002868617 0.676419966

Low 0.057772484 0.119714116 0.822513401

Medium 0.047855530 0.069977427 0.882167043

$life\_sq

life\_sq

Y High Low Med

High 0.17346353 0.01435956 0.81217691

Low 0.03052065 0.04847397 0.92100539

Medium 0.07393172 0.02622654 0.89984174

$floor

floor

Y High Low Med

High 0.30751578 0.08376363 0.60872060

Low 0.00000000 0.29166667 0.70833333

Medium 0.09345372 0.09322799 0.81331828

$max\_floor

max\_floor

Y High Low Med

High 0.44834650 0.04653205 0.50512145

Low 0.00000000 0.35115512 0.64884488

Medium 0.00000000 0.07020316 0.92979684

$material

material

Y High Low Med

High 0.1833046 0.6219162 0.1947791

Low 0.2166667 0.6101190 0.1732143

Medium 0.1909707 0.6530474 0.1559819

$build\_year

build\_year

Y Med

High 1

Low 1

Medium 1

$num\_room

num\_room

Y High Low Med

High 0.52998565 0.08378766 0.38622669

Low 0.15792610 0.51251490 0.32955900

Medium 0.09509826 0.49152925 0.41337249

$kitch\_sq

kitch\_sq

Y High Low Med

High 0.004016064 0.014056225 0.981927711

Low 0.006547619 0.040476190 0.952976190

Medium 0.004063205 0.027539503 0.968397291

> # Accuracy on training set

> train\_predictions = predict(Naive\_Bayes\_Model, train[,-9]) # Removed the ninth column (target variable)

> confusionMatrix(train\_predictions, train$y)# comparing actual price classes and predicted price classes in the train dataset

Error: `data` and `reference` should be factors with the same levels.

> # Accuracy on the test set

> test\_predictions = predict(Naive\_Bayes\_Model, test[,-9])# removes the ninth column (target variable)

> confusionMatrix(test\_predictions, test$y)# comparing actual price classes and predicted price classes in the test dataset

Deliverable 3

Talk about the challenges you might face or elaborate your thoughts about your findings such as a) how would you handle categorical data in KNN and b) what happens if Naïve Bayes main assumption is not supported.

* We can handle categorical data in KNN by using the R function into iteration value of K which it is a function of accuracy into the training & validation data set
* So, if we want to use the method, first we have to define the initial value of the K
* If we use Naive Bayes, but the main assumption is not supported, we can use the previous value of iterated K and choose fot the maximum accuracy level.