Naive Bayes is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors.

The Naive Bayes model is easy to build and particularly useful for very large data sets. When you have a large dataset think about Naive classification.

**Naive Bayes algorithm Process Flow**

Take an example, Imagine because of current weather, cricket match will happen or not? Now, we need to classify whether players will play the match or not based on weather conditions.

1. Convert the data set into a frequency table
2. Create a Likelihood table by finding the probabilities like play the match or not
3. Based on the Naive Bayes equation calculate the posterior probability for each class. The highest posterior probability in each class is the outcome of the prediction.

It is easy to use and fast to predict class of test data set.

It perform well in case of categorical input variables compared to numerical variable(s).

Its required independent predictor variables for better performance.

Let’s see, how to execute Naïve Bayes classification in R?

**Load libraries**

library(naivebayes)

library(dplyr)

library(ggplot2)

library(psych)

**Getting Data**

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

head(data)

Launch Thickness Appearance Spreading Rank

   0         6          9         8    2

   0         5          8         7    2

   0         8          7         7    2

   0         8          8         9    1

   0         9          8         7    2

   0         7          7         7    2

Let us understand the dataset, the dataset contains 5 columns.

Launch- Response variable, 0 indicates product not launched and 1 indicates product is launched

Thickness-product thickness score

Appearance-product appearance score

Spreading- product spreading score

Rank-Rank of the product

**Frequency Identification**

Let’s calculate the frequency of response variable under each rank. The minimum frequency of each class is 5 required for analysis.

xtabs(~Launch+Rank, data = data)

Rank

    Rank

Launch  1  2  3

     0 12 21 13

     1 21 15 13

In this all-cell frequencies are greater than 5 and ideal for further analysis.

Now just look at each variable class based on str function

str(data)

data.frame':  95 obs. of  5 variables:

 $ Launch             : int  0 0 0 0 0 0 0 0 0 0 ...

 $ Thickness          : int  6 5 8 8 9 7 8 8 8 8 ...

 $ ColourAndAppearance: int  9 8 7 8 8 7 9 7 9 9 ...

 $ EaseOfSpreading    : int  8 7 7 9 7 7 8 7 9 8 ...

 $ Rank               : int  2 2 2 1 2 2 2 2 1 2 ...

Now you can see the data frame contains 95 (still small dataset you can try Naive Bayes for large datasets) observations of 5 variables. The columns Launch and Rank stored as integer variables. If these two variables appearing as integer needs to convert into factor variables.

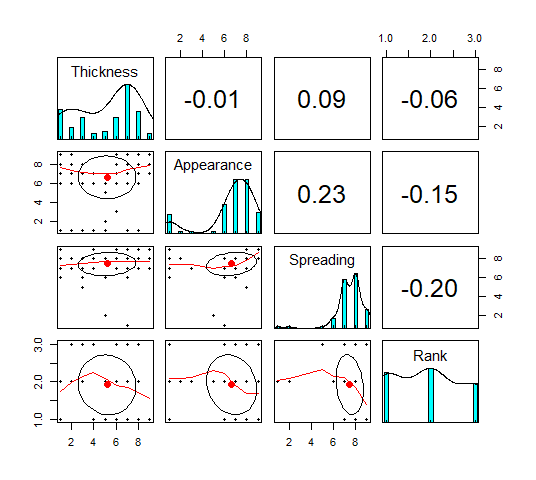
data$Rank <- as.factor(data$Rank)

data$Launch <- as.factor(data$Launch)

When we are doing naïve Bayes classification one of the assumptions is to independent variables are not highly correlated. In this case, remove the rank column and test the correlation of the predictor variables.

**Visualization**

pairs.panels(data[-1])



Low correlation was observed between independent variables.

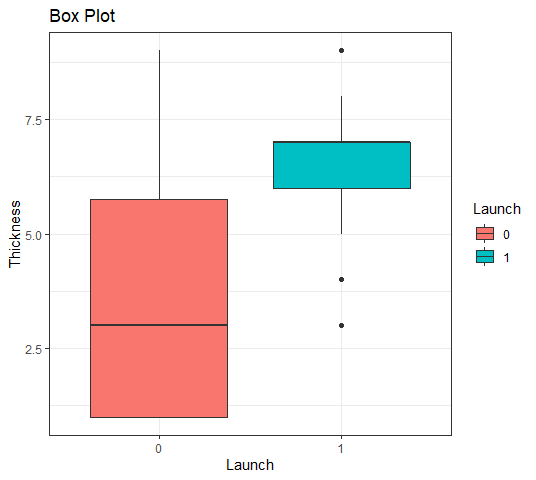
**Visualize the data based on ggplot**

data %>%

         ggplot(aes(x=Launch, y=Thickness, fill = Launch)) +

         geom\_boxplot() +theme\_bw()+

         ggtitle("Box Plot")



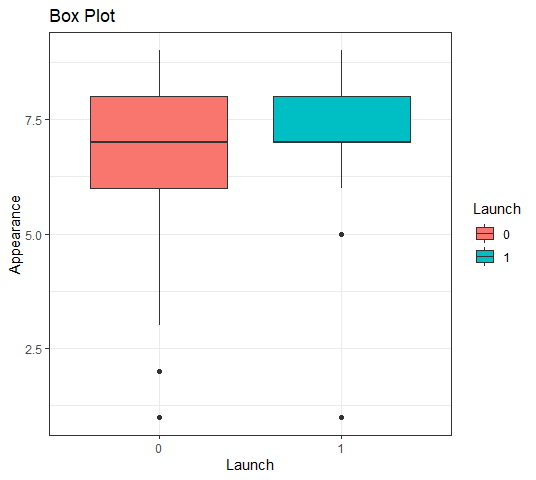
Product got highest score in the thickness got launched in the market.

data %>%

ggplot(aes(x=Launch, y=Appearance, fill = Launch)) +

geom\_boxplot() +theme\_bw()+

ggtitle("Box Plot")

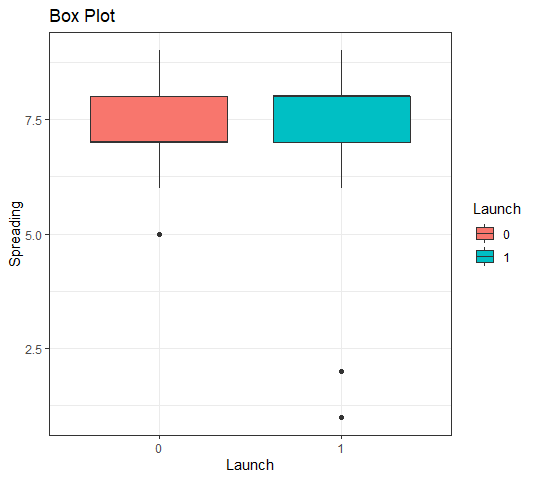


data %>%

  ggplot(aes(x=Launch, y=Spreading, fill = Launch)) +

  geom\_boxplot() +theme\_bw()+

  ggtitle("Box Plot")



**Data Partition**

Let’s create train and test data sets for training the model and testing.

set.seed(1234)

ind <- sample(2, nrow(data), replace = T, prob = c(0.8, 0.2))

train <- data[ind == 1,]

test <- data[ind == 2,]

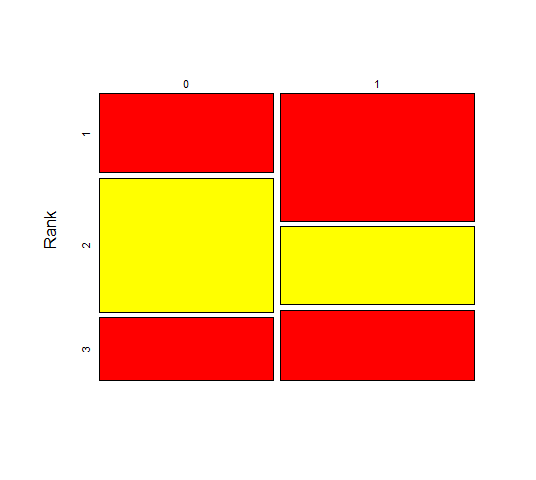
**Naive Bayes Classification**

Naive Bayes Classification in R

model <- naive\_bayes(Launch ~ ., data = train, usekernel = T)

model plot(model)

You can try usekernel = T without also, based on model accuracy you can adjust the same.



Product received rank 1 score launch chances are very high and products received rank 3 also have some chances to a successful launch.

**Prediction**

p <- predict(model, train, type = 'prob')

head(cbind(p, train))

        0            1 Launch Thickness Appearance Spreading Rank

1 0.9999637 3.629982e-05      0         1          9         8    2

2 0.9998770 1.229625e-04      0         1          8         7    1

3 0.9998804 1.196174e-04      0         1          7         7    1

4 0.9997236 2.764280e-04      0         1          8         9    1

6 0.9998804 1.196174e-04      0         1          7         7    1

7 0.9999637 3.629982e-05      0         1          9         8    2

Basis first row, Low thickness, high appearance, Spreading and Rank score 2, has very low chance of product launch.

**Confusion Matrix – train data**

p1 <- predict(model, train)

(tab1 <- table(p1, train$Launch))

p1 0 1

0 28 2

1 7 37

1 - sum(diag(tab1)) / sum(tab1)

0.1351351

Misclassification is around 14%.

Training model accuracy is around 86% not bad!.

**Confusion Matrix – test data**

p2 <- predict(model, test)

(tab2 <- table(p2, test$Launch))

p2 0 1

0 8 0

1 3 10

1 - sum(diag(tab2)) / sum(tab2)

0.1428571

**Conclusion**

Based on Naive Bayes Classification in R, misclassification is around 14% in test data. You can increase model accuracy in the train test while adding more observations.