

Sistemas Inteligentes para la Gestión en la Empresa

Practice 1

Pre-processing of data and binary classification

**Author**

**Abdullah Taher Sadoon AL-Musawi**



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

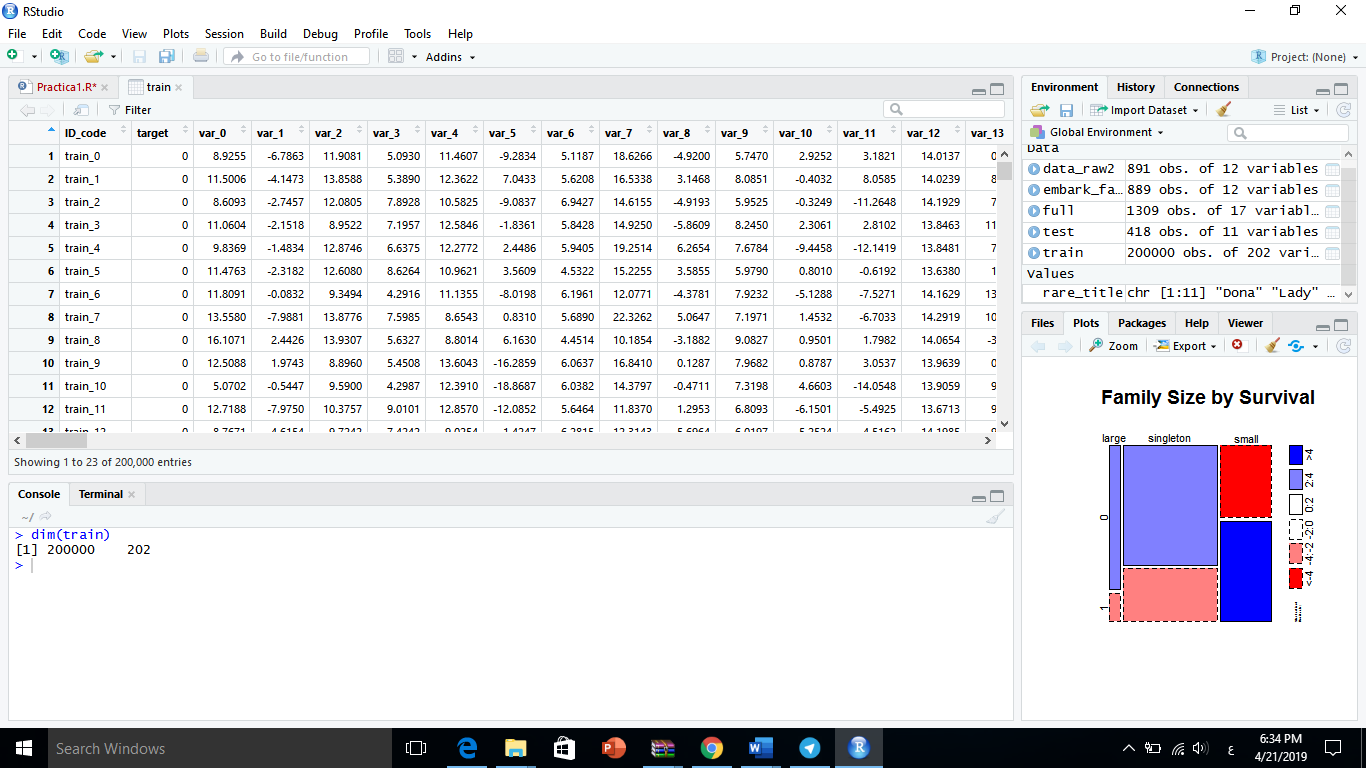
—

Granada, mayo de 2019

**index of contents**

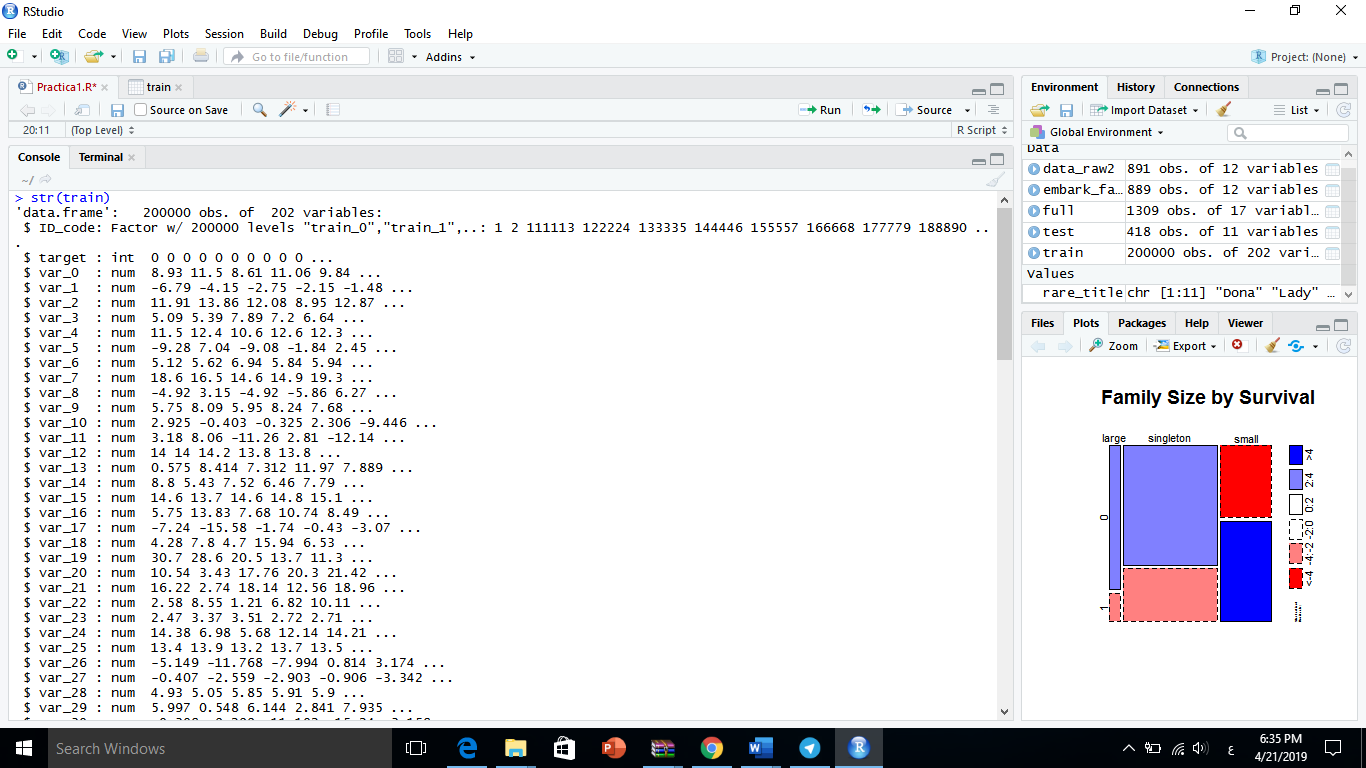
1. Data exploration …………………………………………………………………. 3
2. Data preprocessing………………………………………………………………. 6
3. Classification techniques and discussion of results………………. 11
4. Conclusions………………………………………………………………………… . 21
5. Bibliography……………………………………………………………………… … 20
6. **Data exploration**

Initially Will be loading our dataset ´The Train\_ok´ in R studio, to carry out the verification of the variables that contains and for checking the list provided in the excel file (Data Dictionary) provided for the realization of the practice.

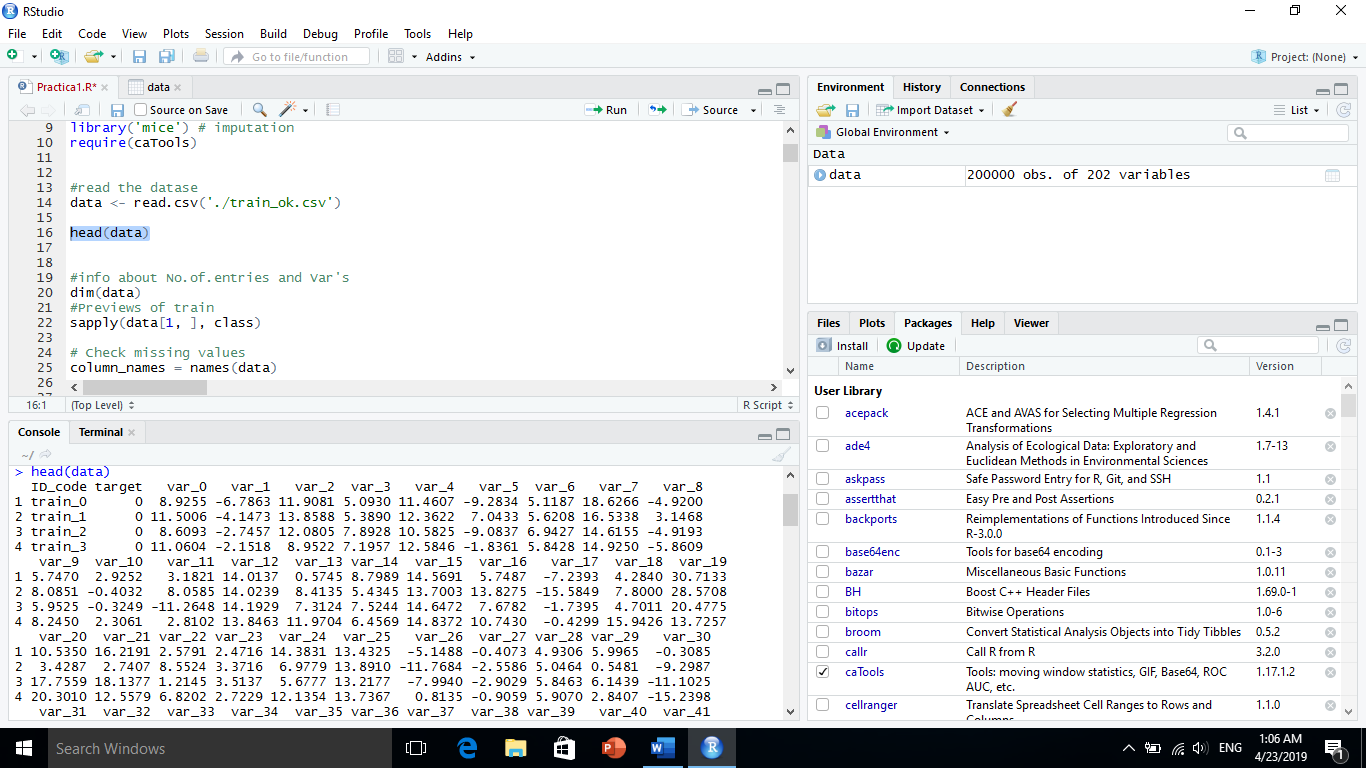


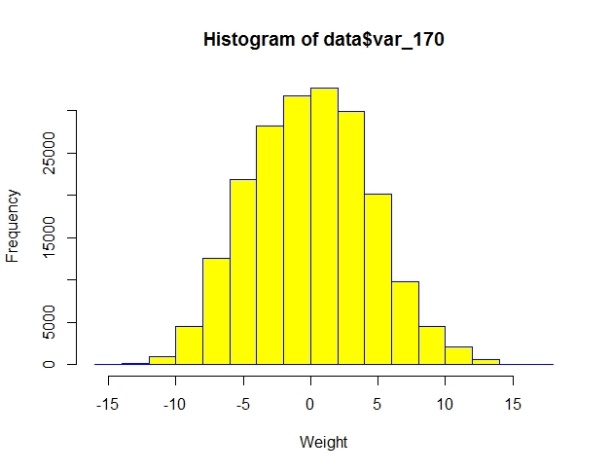
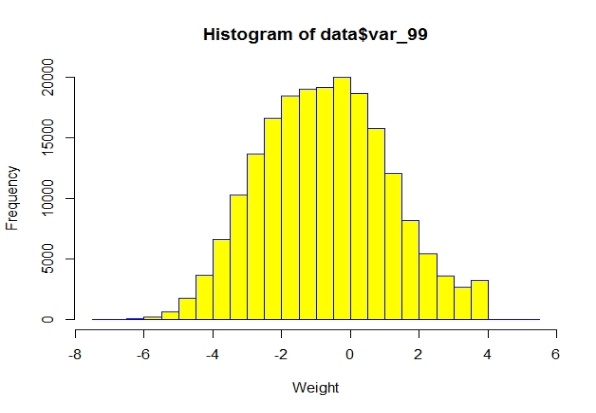
So, we have 202 variables(ID\_code,target,var\_0……var\_199) and 200000 entries, so We will have a data set that you can handle with it.

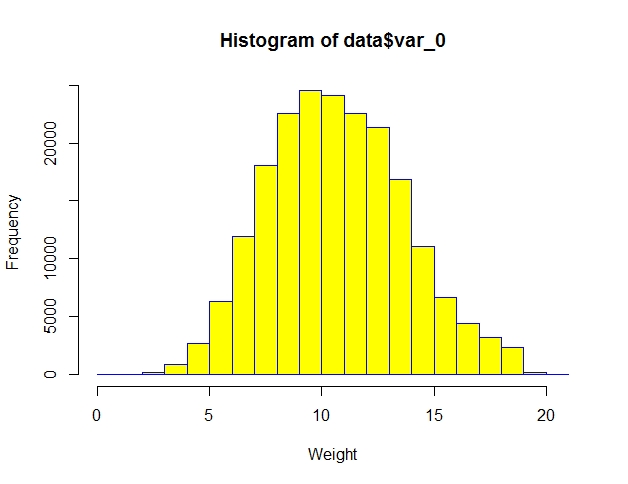
Also, to get more information about the dataset, we can see the internal structure that has:

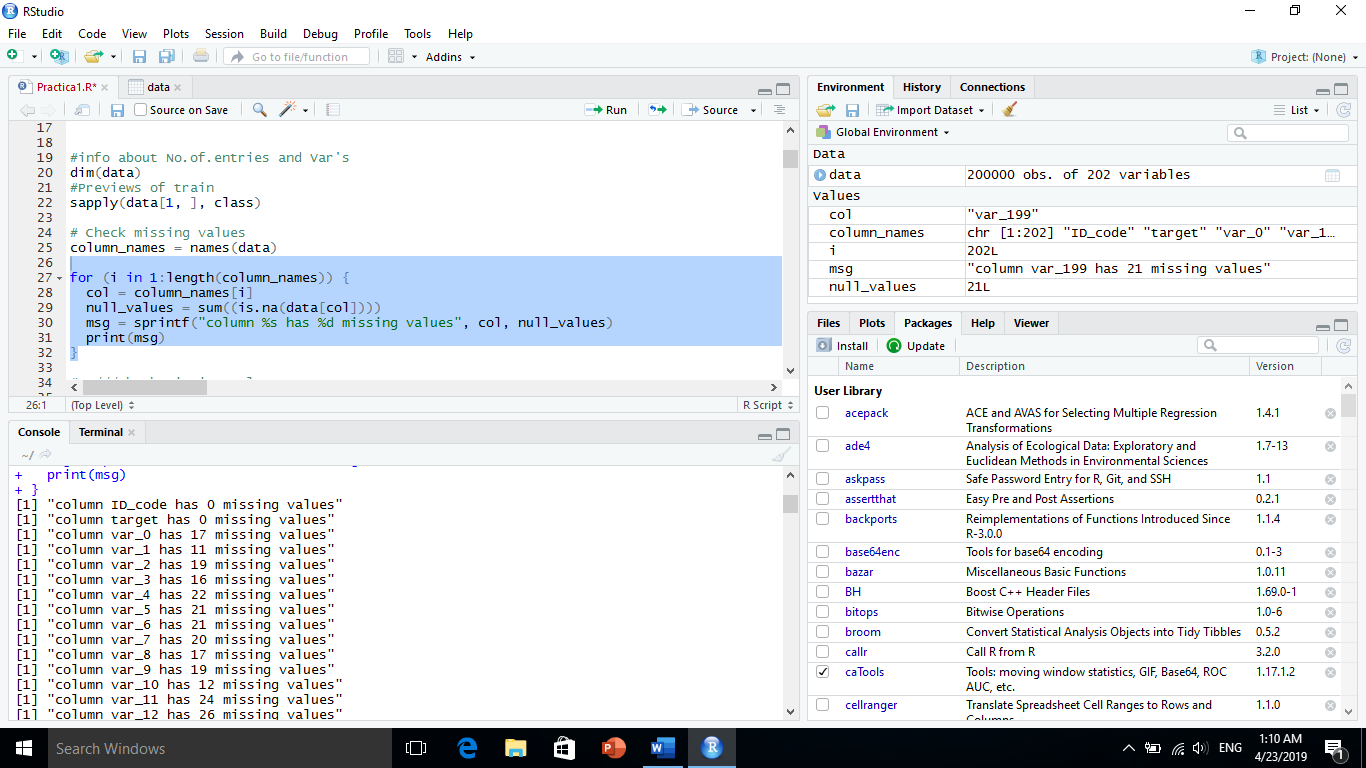


As we go exploring the dataset, we realize that we have variables very unbalanced cases for example with the variables var\_0 ………var\_199, so we will normalize these vars to make its value between 0-1.







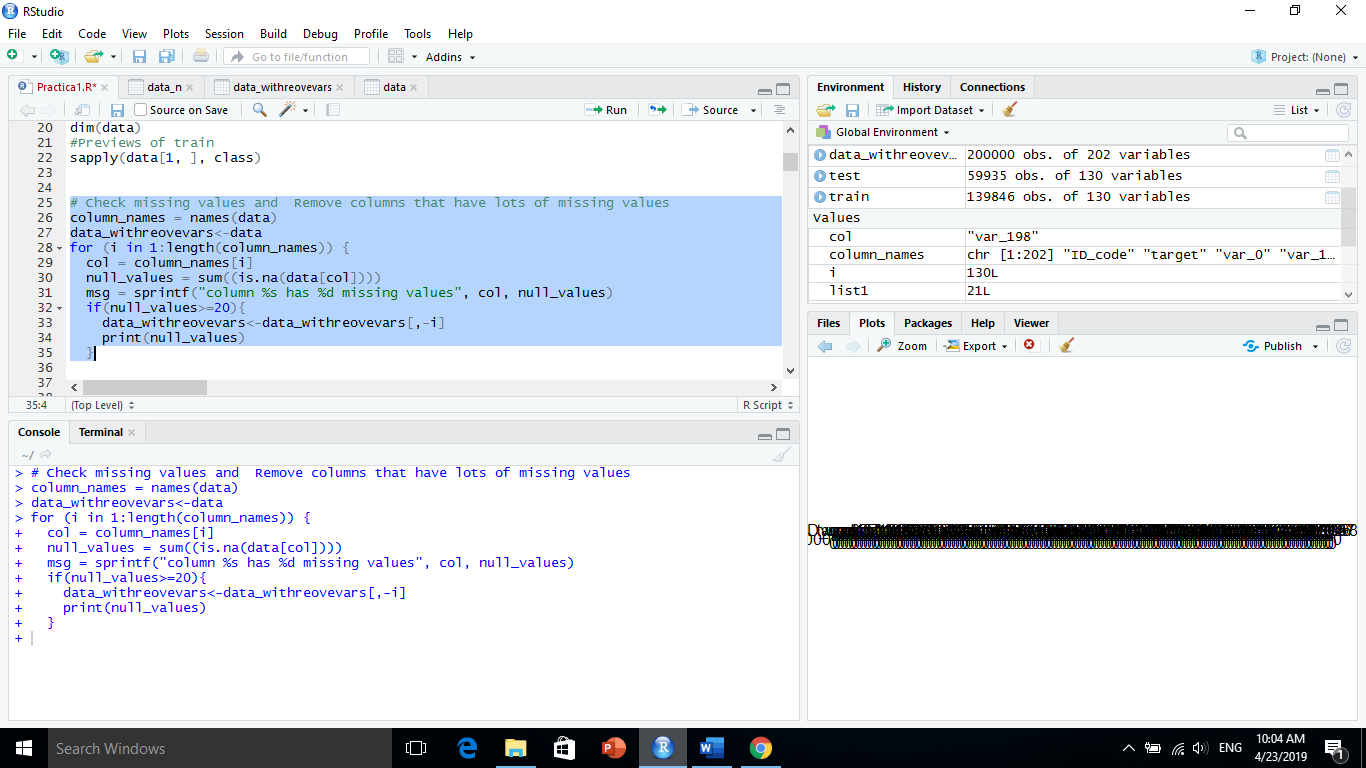
 that we have many missing values as NA’s for example with the variables var\_0…. var\_199 for example var\_0 has 17 missing values and var\_4 has 22 missing values, so we have to complete it before doing classification processing.

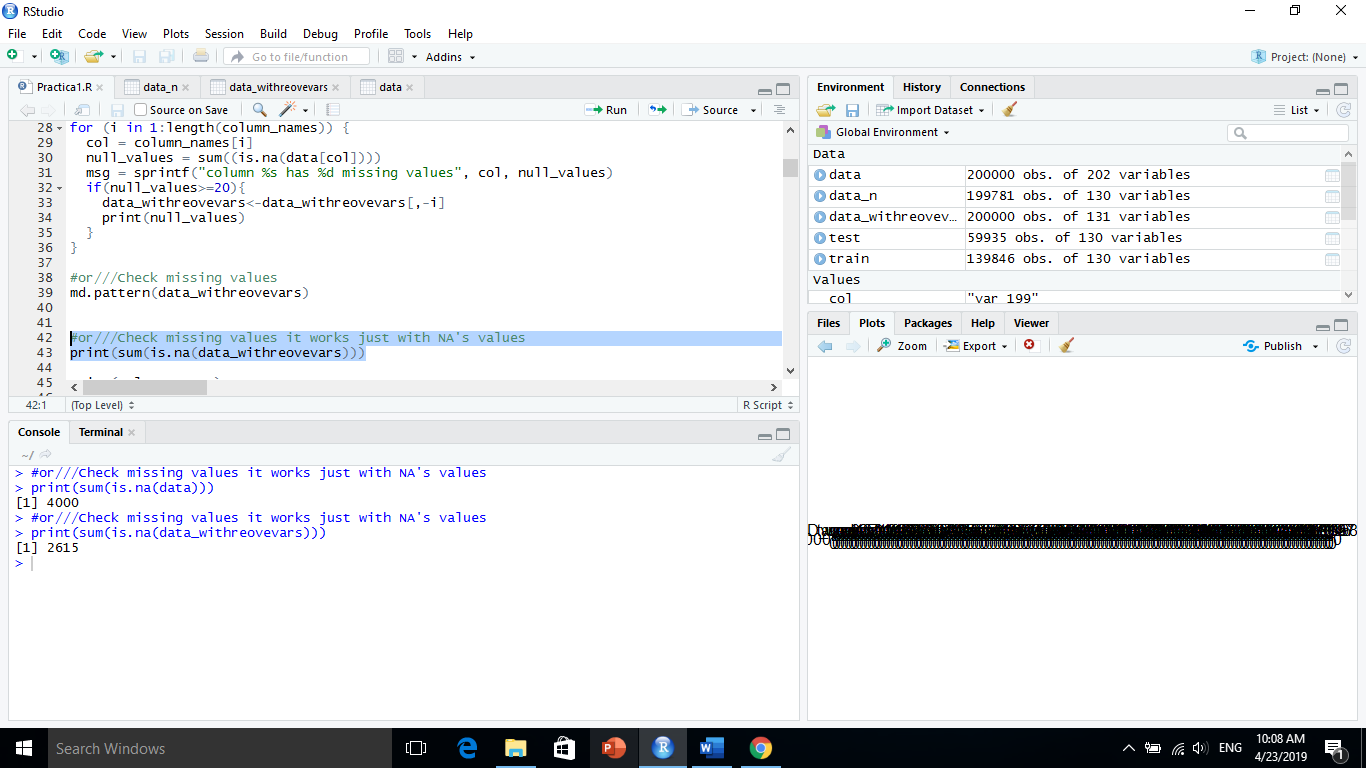
1. **Preparing the data**

Now we’re ready to start exploring missing data and rectifying it through imputation. There are a number of different ways we could go about doing this. Given the small size of the dataset, we probably should not opt for deleting either entire observations (rows) or variables (columns) containing missing values. We’re left with the option of either replacing missing values with a sensible value given the distribution of the data, e.g., the mean, median or mode. Finally, we could go with prediction. We’ll use both of the two latter methods and I’ll rely on some data visualization to guide our decisions.

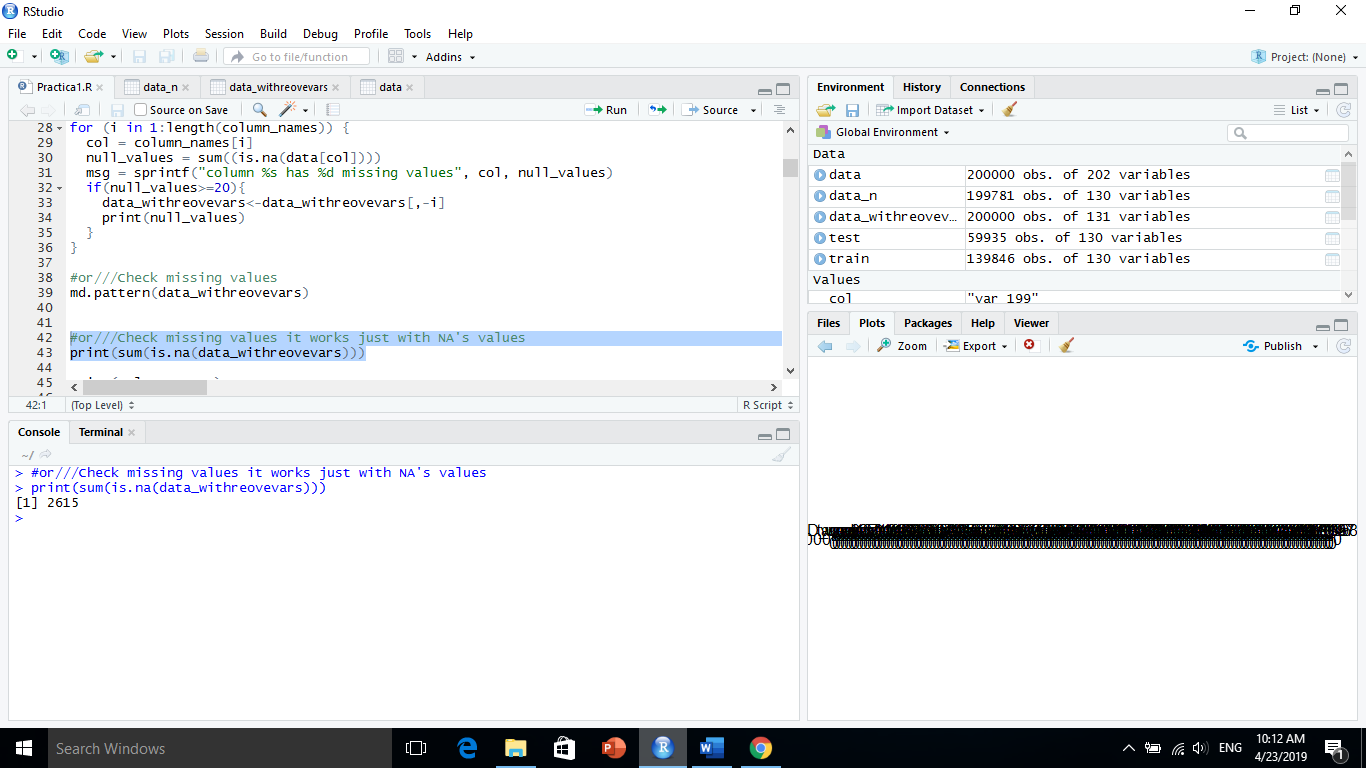
1. **Elimination of variables with less information**

* **Remove variables have 20 or more missing value**

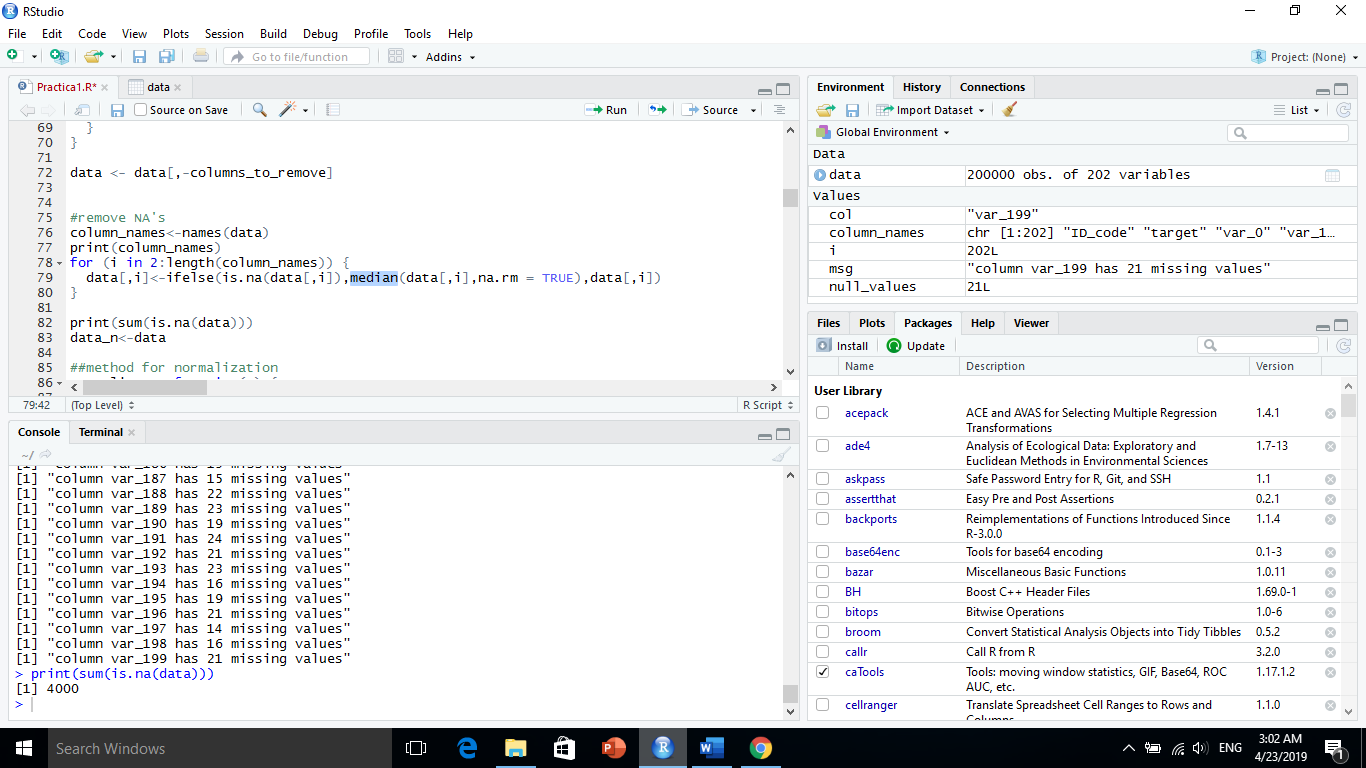


So we can see that we have variables with less missing values, the missing values have less to half of the origin data

1. **missing values**

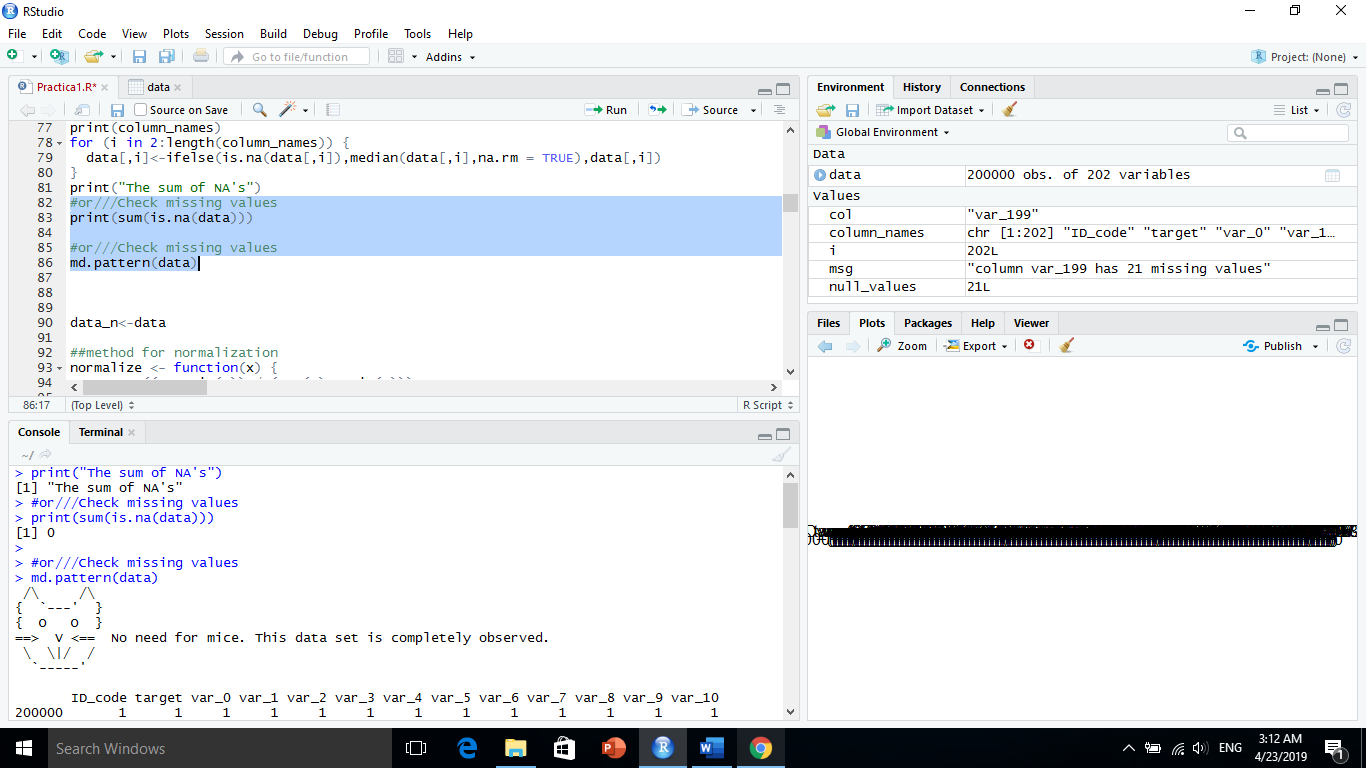
One of the most common problems I have faced in Data Cleaning/Exploratory Analysis is handling the missing values. Firstly, understand that there is NO good way to deal with missing data. In our data set there are a lot of missing values as a NA’s missing value there are 2615 missing values exactly, we can see them with that figure Below

so, we can fix it with library('mice') # imputation by using ‘median’ of variables



After that we can Check missing values for our data set by ***md.pattern(data)*** Now we can see new data set without missing values, see the figure Below

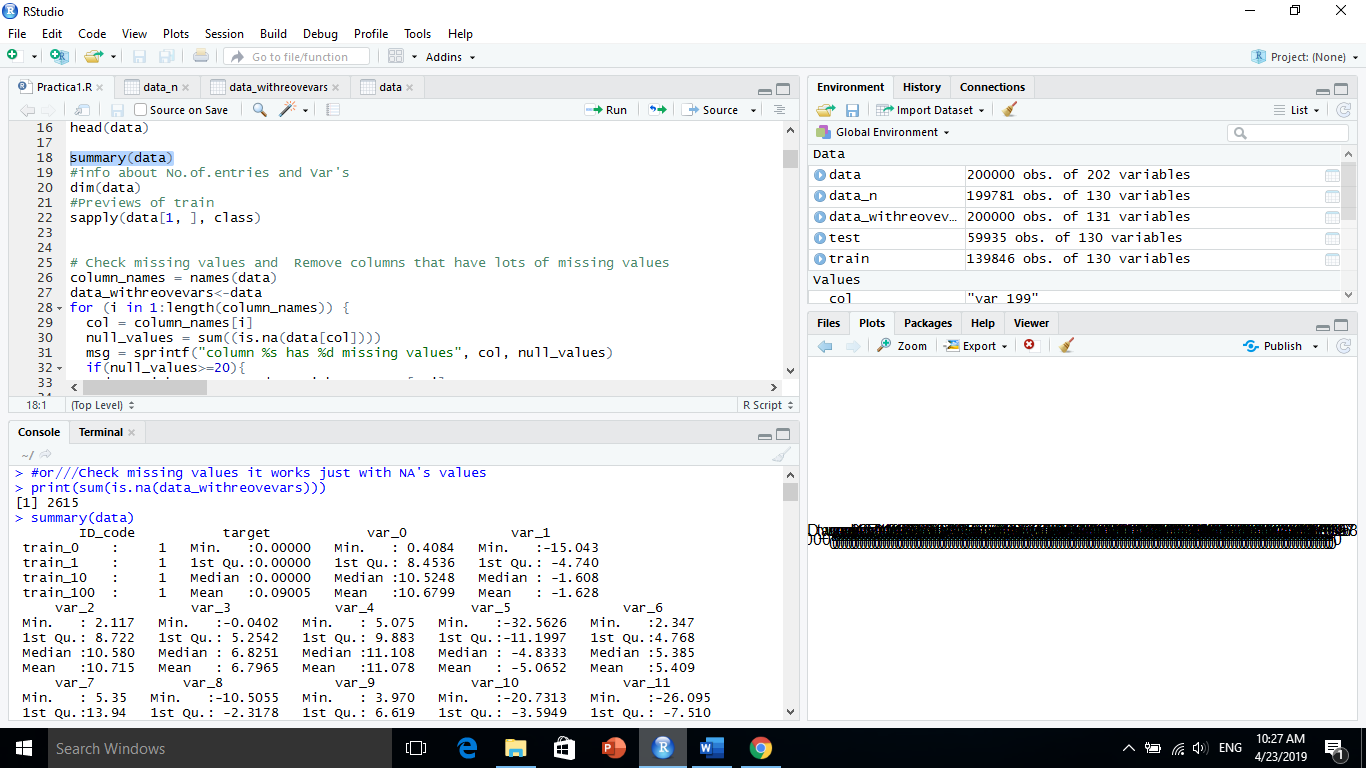
.



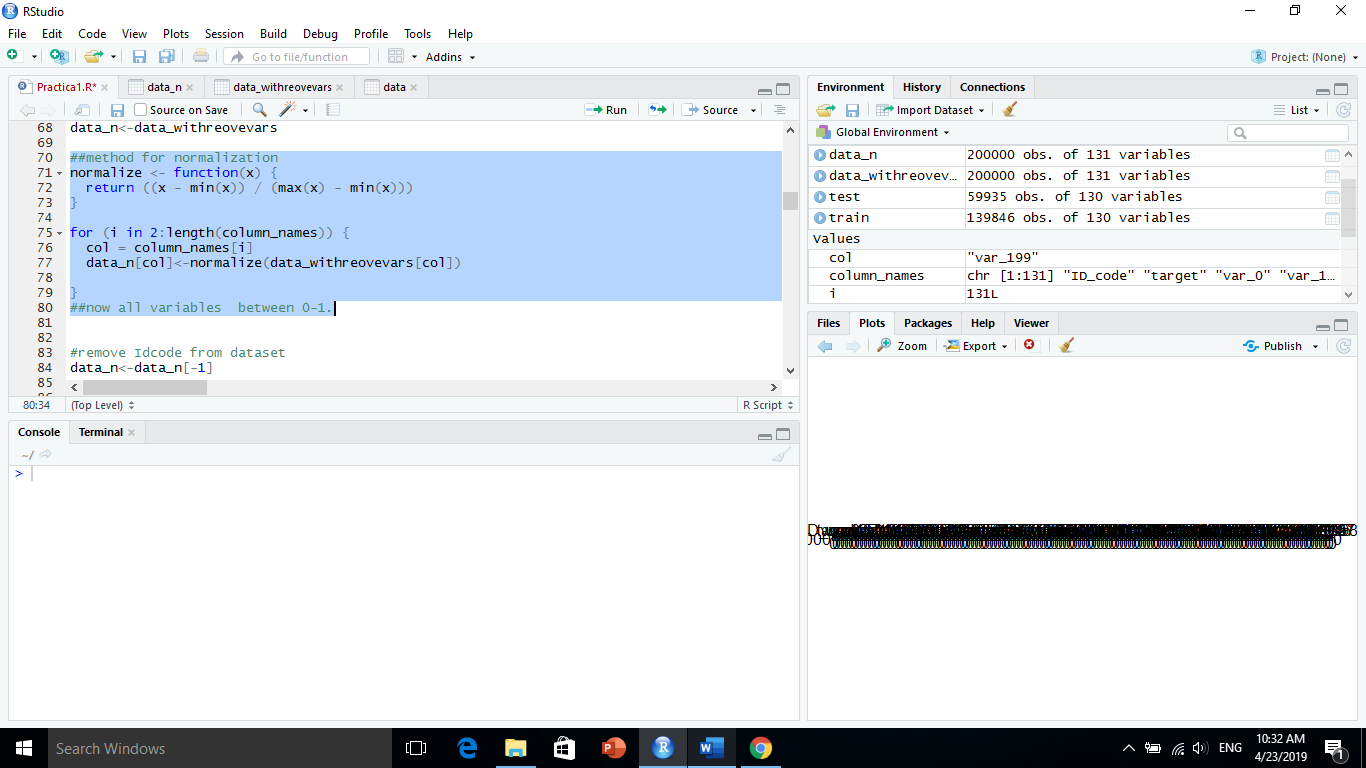
1. **Normalization**

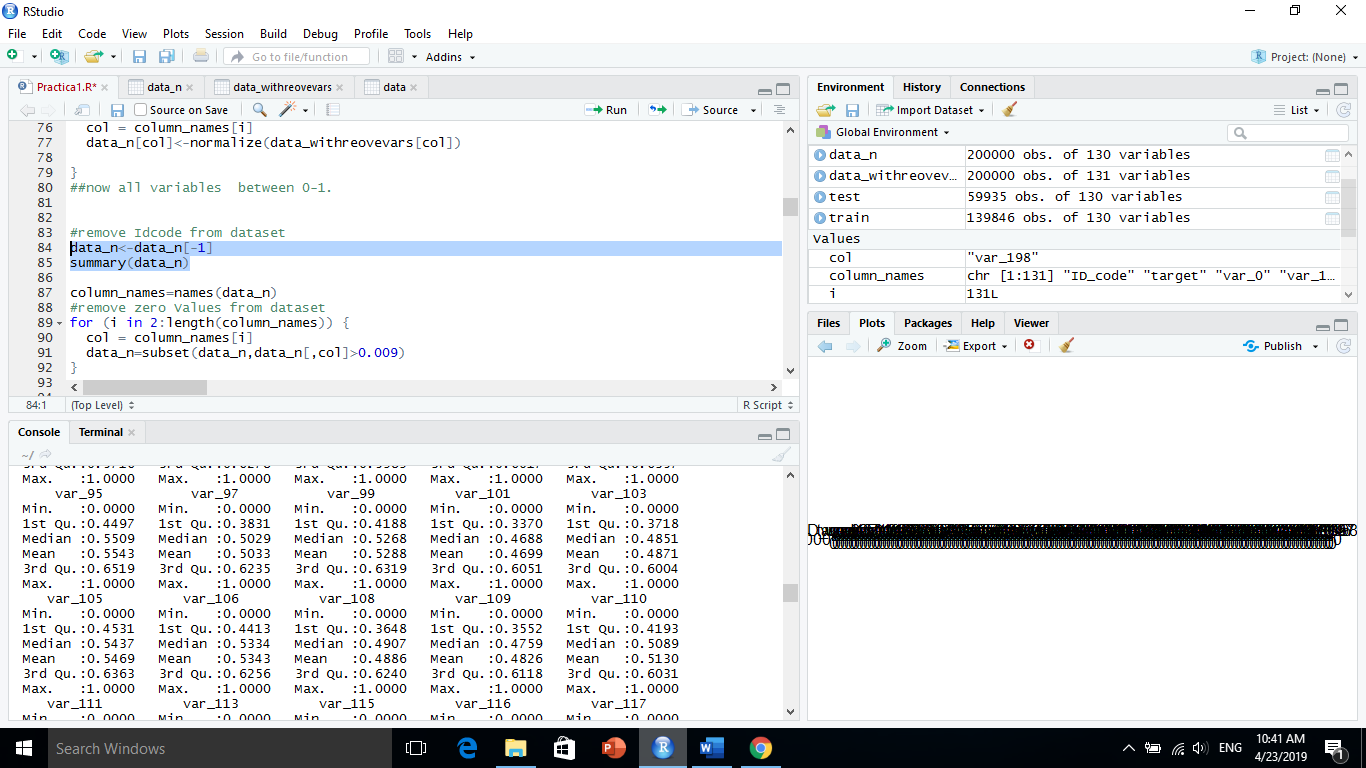
we realize that we have a lot of variables it’s very unbalanced

for example, the values of var\_0 be between (0.4084….10.6799) either that the values of var\_1 be between (-15.043…. -1.628) and we have many cases same that, the Figure below show that



My solve has been making all of values of variables between 0-1 by using (max-min values),



 After that Solution will Appear values equal zero or very near of zero,

so will remove them with simple R code that be

column\_names=names(data\_n)

#remove zero Values from dataset

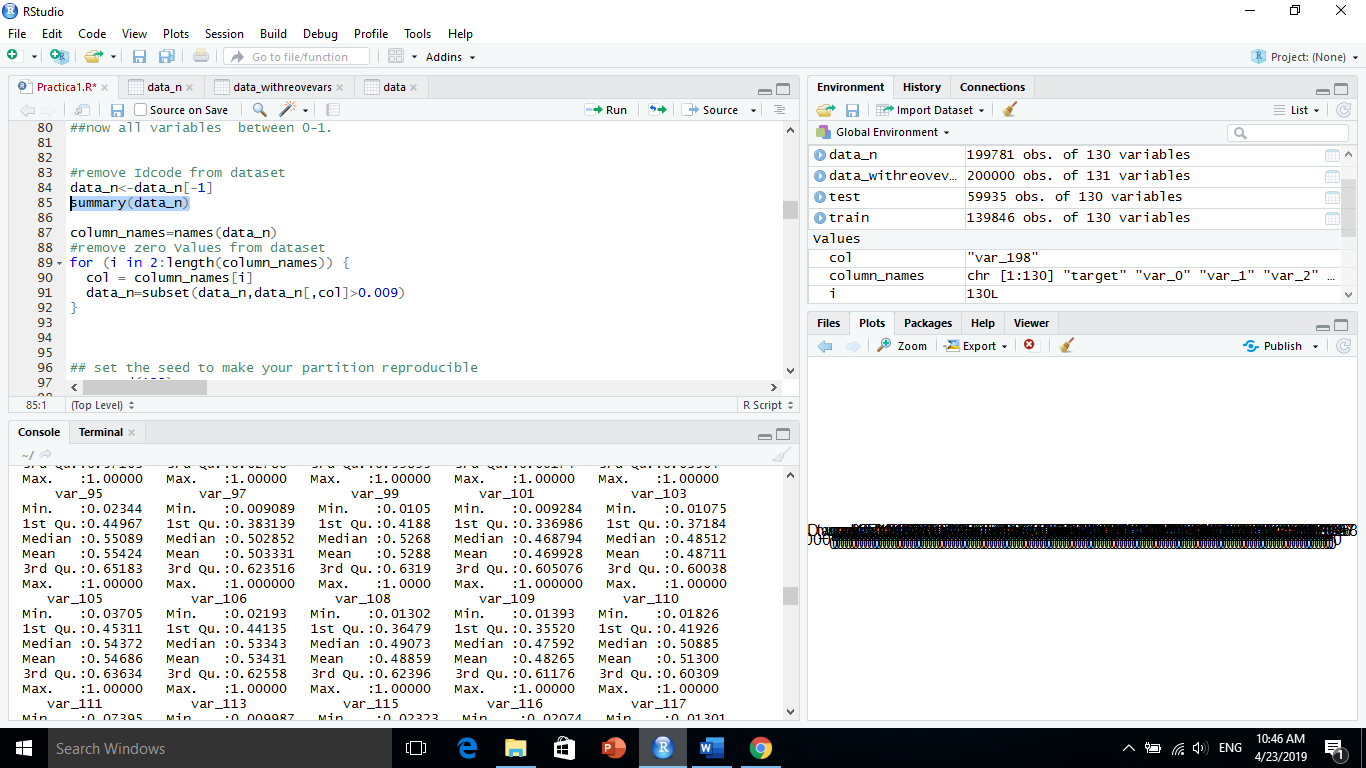
for (i in 2:length(column\_names)) {

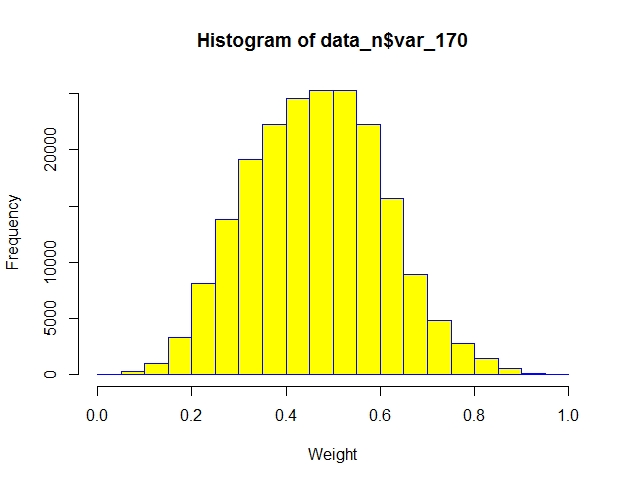
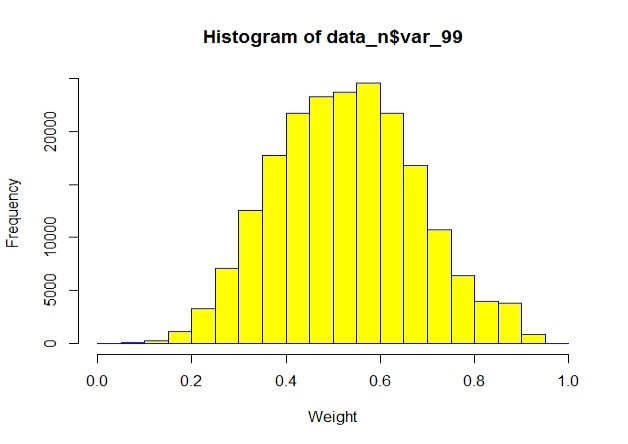
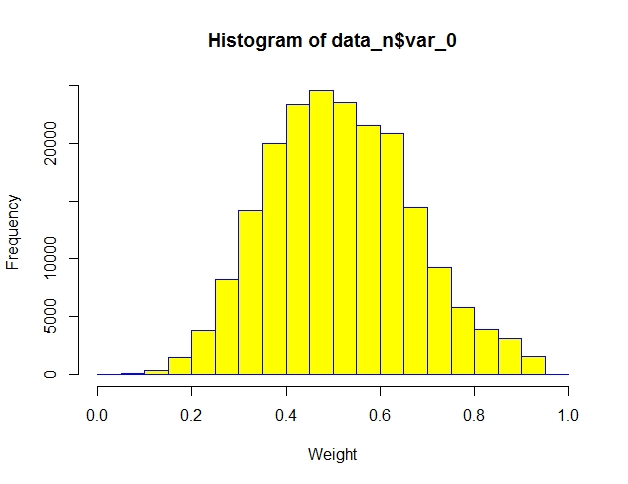
col = column\_names[i]

data\_n=subset(data\_n,data\_n[,col]>0.009)

}

The summary of dataset with normalization will be (<1 and > 0.009) like these values in the figure below





Now our dataset is ready to split into two datasets ‘’train and test’’, The data will be fragmented depending on target variable, the 70% of data will be in train dataset and 30% will be in test dataset the R code below show that.

## set the seed to make your partition reproducible

set.seed(123)

sample = sample.split(data\_n$target, SplitRatio = .7)

train = subset(data\_n, sample == TRUE)

test = subset(data\_n, sample == FALSE)

///

|  |
| --- |
| > dim(train)  [1] 139846 130  > dim(test)  [1] 59935 130 |
|  |
| |  | | --- | | > | |

**3- Classification techniques**

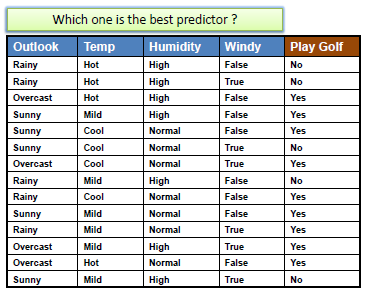
* **OneR:**

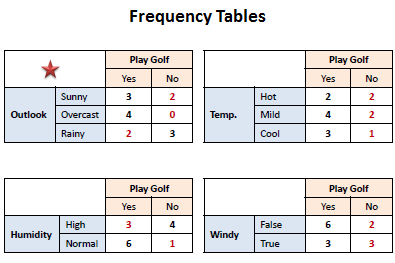
OneR, short for "One Rule", is a simple, yet accurate, classification algorithm that generates one rule for each predictor in the data, then selects the rule with the smallest total error as its "one rule".  To create a rule for a predictor, we construct a frequency table for each predictor against the target. It has been shown that OneR produces rules only slightly less accurate than state-of-the-art classification algorithms while producing rules that are simple for humans to interpret.

* **OneR Algorithm:**

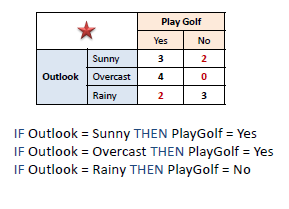
For each predictor, For each value of that predictor, make a rule as follows; Count how often each value of target (class) appears Find the most frequent class Make the rule assign that class to this value of the predictor Calculate the total error of the rules of each predictor Choose the predictor with the smallest total error.

|  |
| --- |
|  |
| *Example:* |  |  |
| Finding the best predictor with the smallest total error using OneR algorithm  based on related frequency tables. |  |  |

****

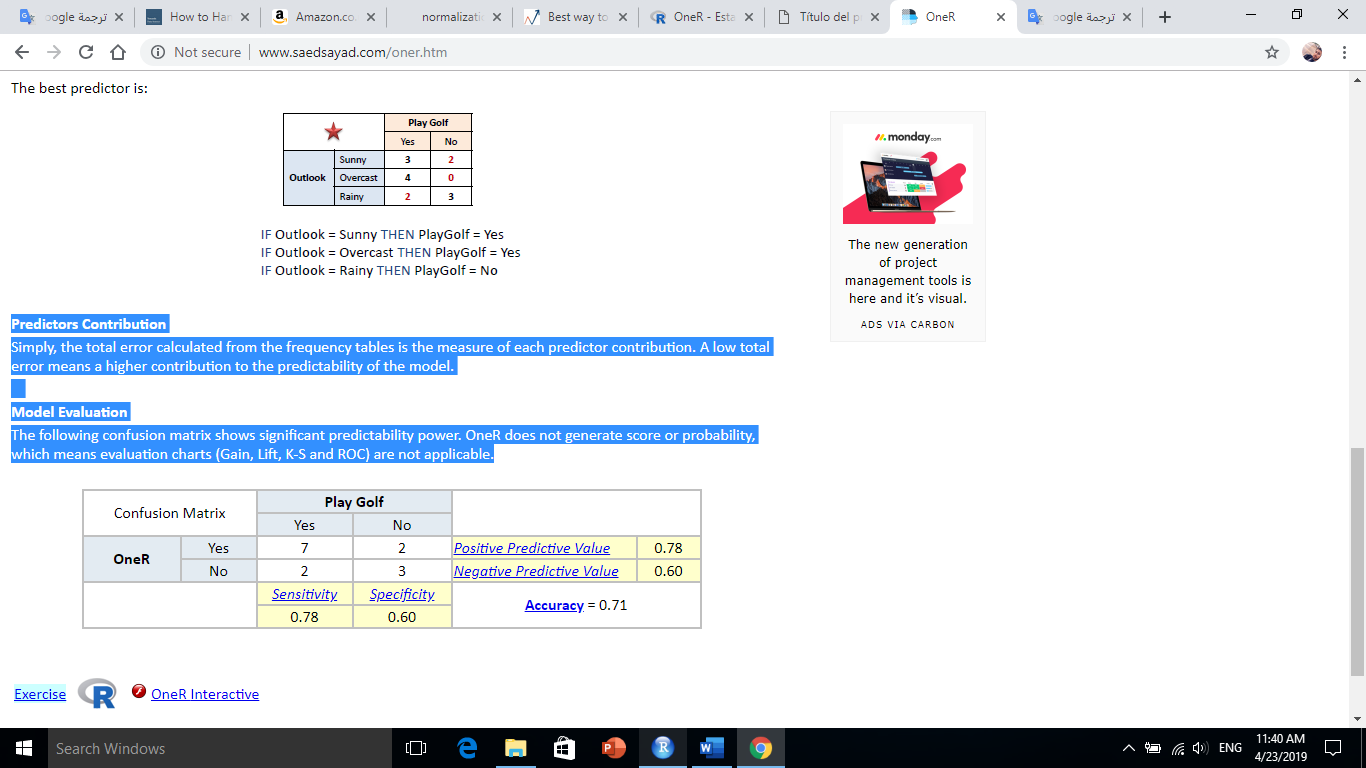


**The best predictor is:**

****

**Predictors Contribution**

Simply, the total error calculated from the frequency tables is the measure of each predictor contribution. A low total error means a higher contribution to the predictability of the model.

**Model Evaluation** The following confusion matrix shows significant predictability power. OneR does not generate score or probability, which means evaluation charts (Gain, Lift, K-S and ROC) are not applicable.

* **The working with OneR**

It’s so easy to work with it in R Language we will need ‘library(OneR)’

Now we have train dataset to apply this algorithm and test for do testing and show the result, accuracy depending on the target that are 0,1 (Yes, No), the R code will be :

library(OneR)

train <- optbin(target ~ ., data = train)

model <- OneR(train, verbose = TRUE)

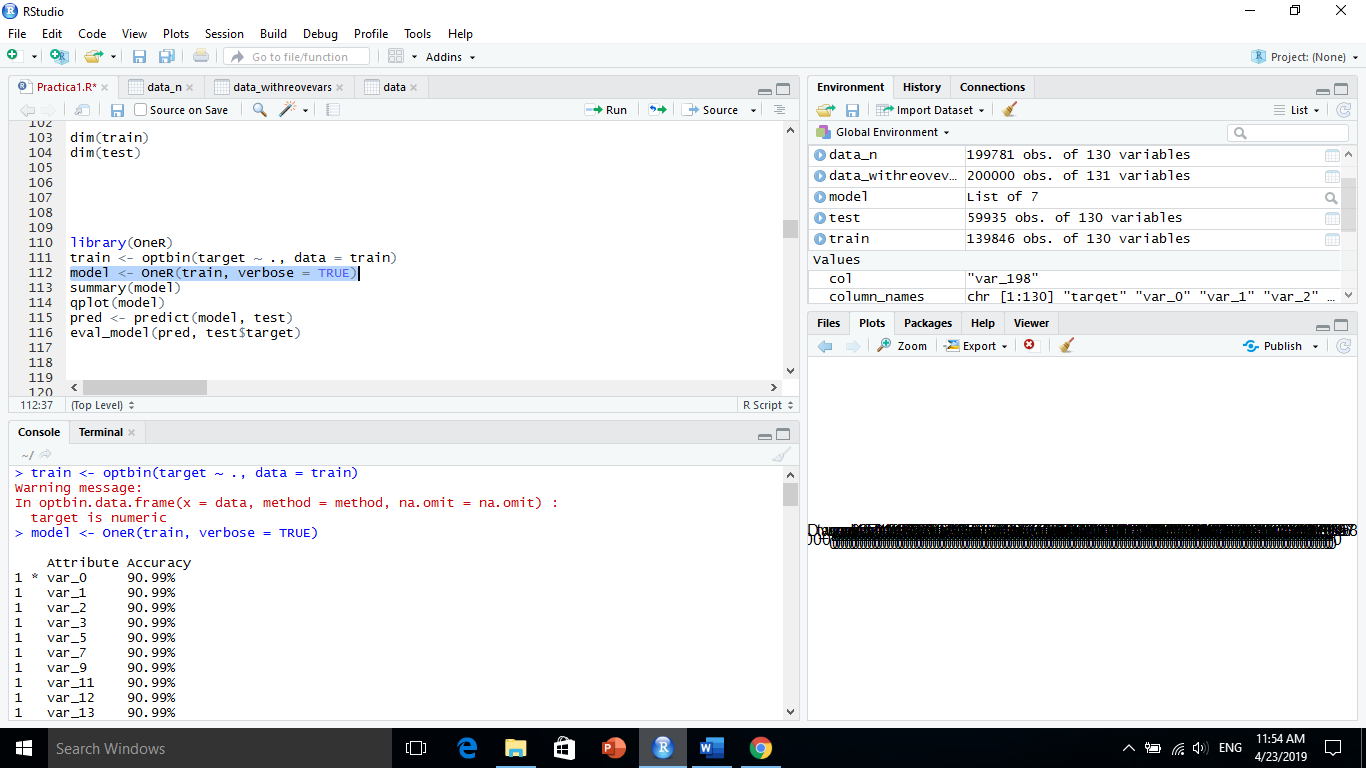
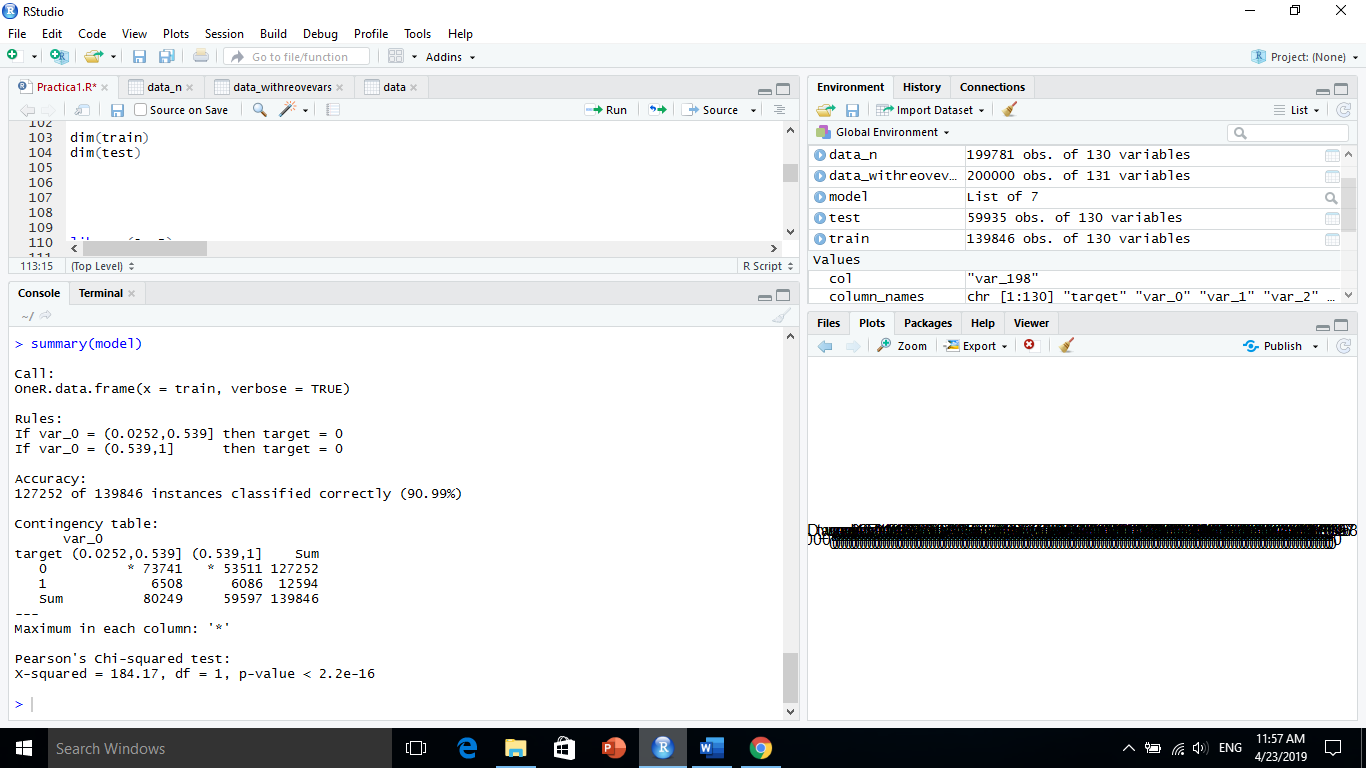
summary(model)

qplot(model)

pred <- predict(model, test)

eval\_model(pred, test$target)

Now the result will be:





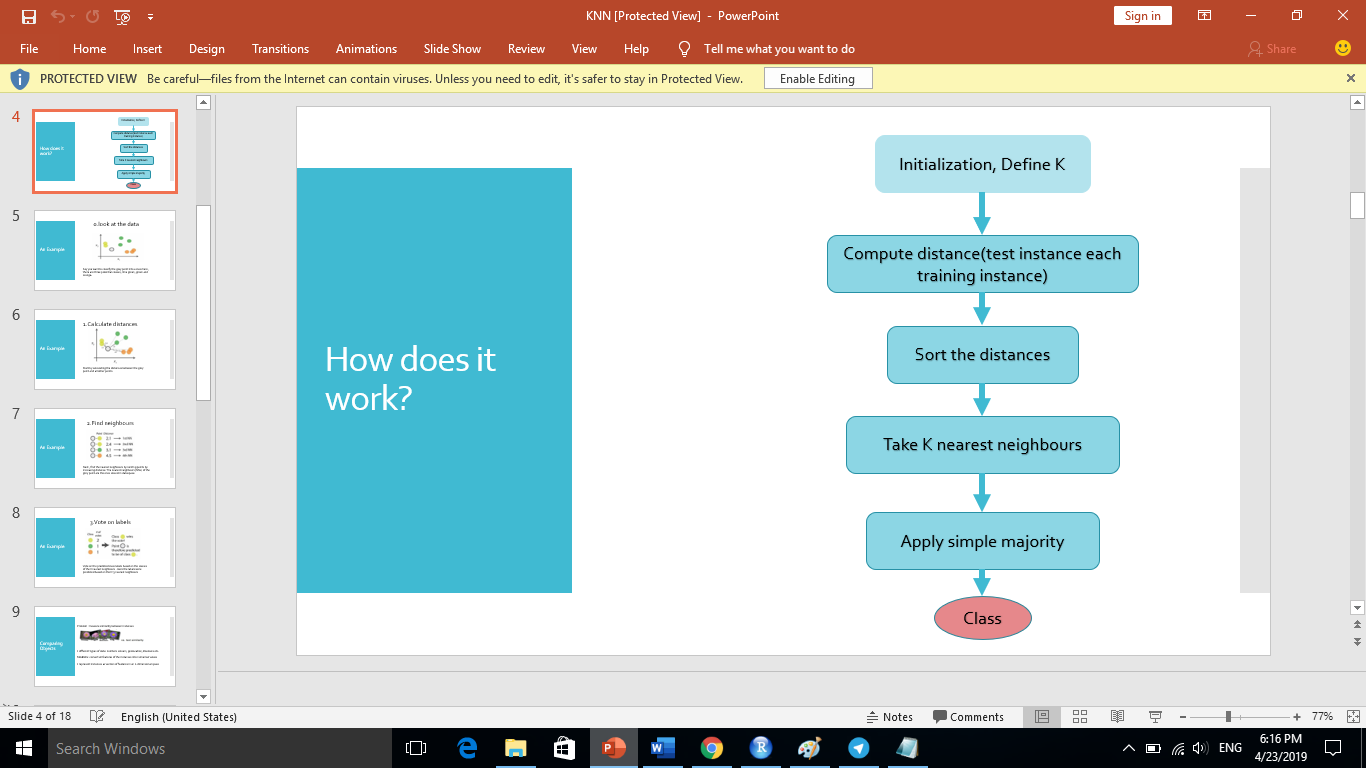
* **KNN (K-Nearest Neighbour):**

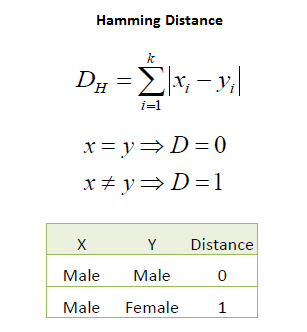
K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970’s as a non-parametric technique.

* **KNN algorithm**

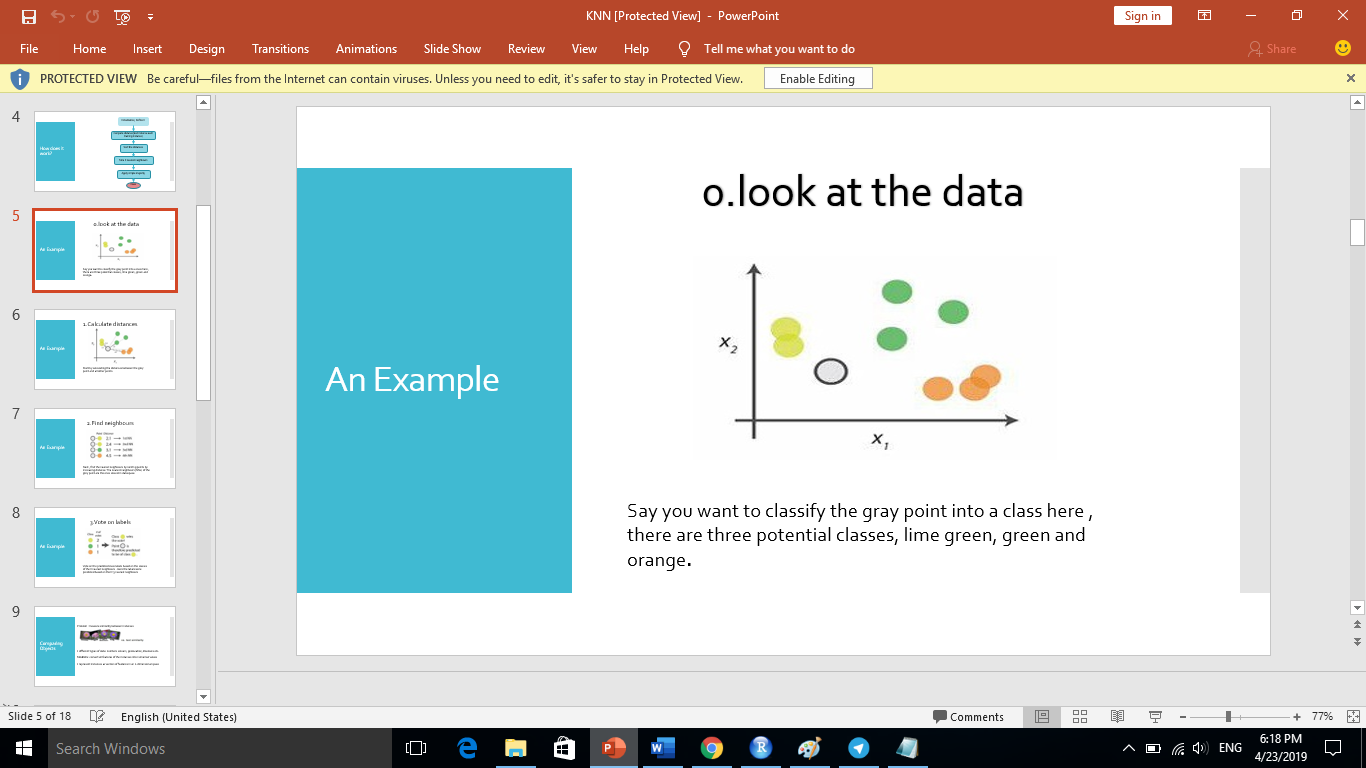
A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

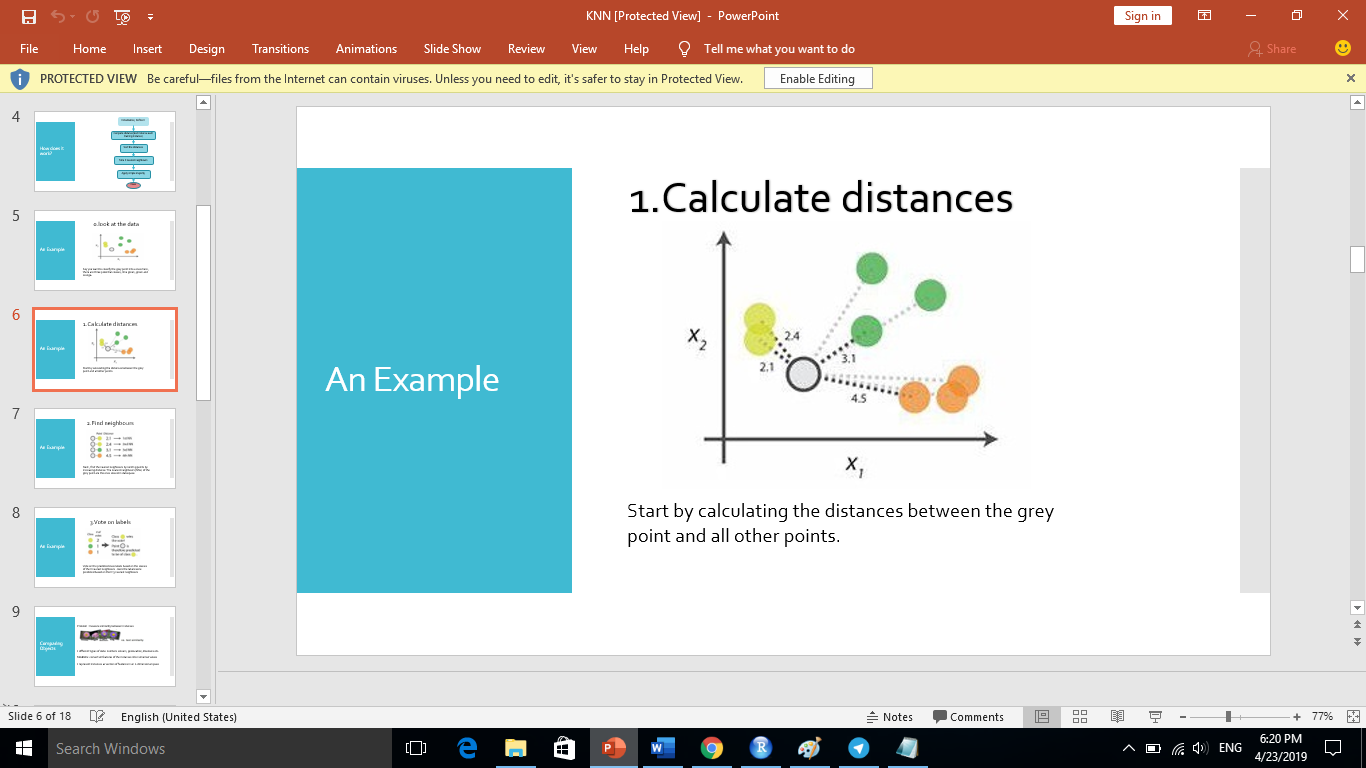
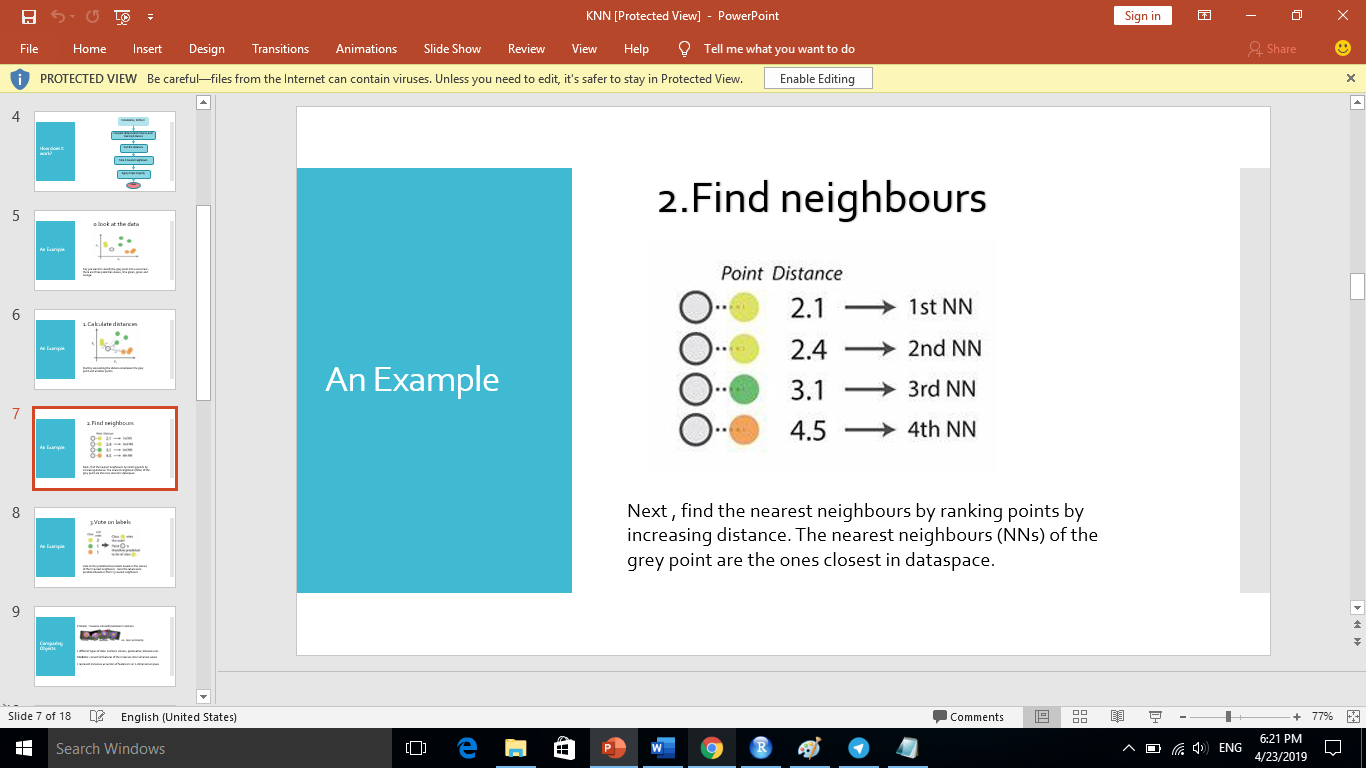


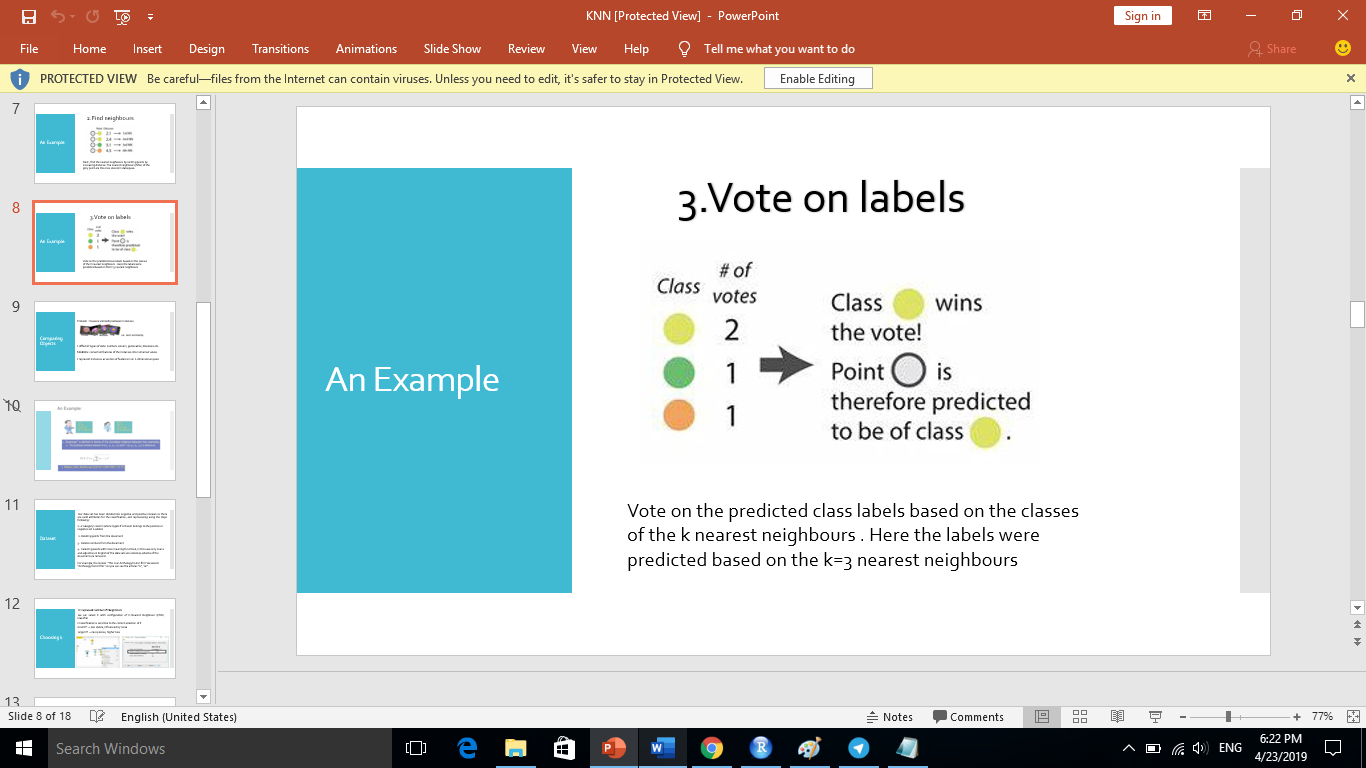


It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables, the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

*Example*:





* **The working with OneR**

It’s so easy to work with it in R Language we will need ‘library(class)’

Now we have train dataset to apply this algorithm and test for do testing and show the result, accuracy depending on the target that are 0,1 (Yes, No), We have built the model we also need to check the accuracy of the predicted values in test\_s as to whether they match up with the known values in test\_labels. To ensure this, we need to use the CrossTable() function available in the package ‘gmodels’, the R code will be :

train\_lables <- data\_withreovevars[sample,1]

test\_lables <- data\_withreovevars[140001:200000,1]

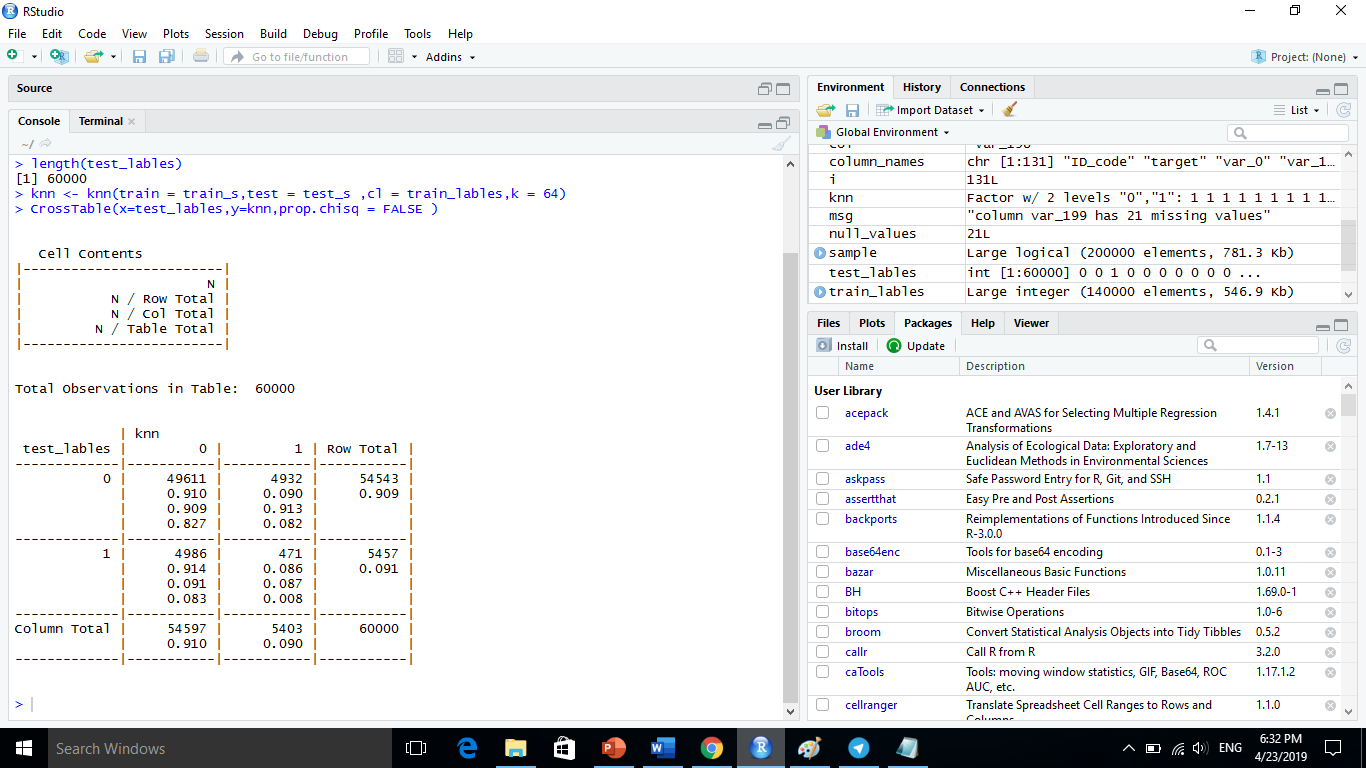
library(class)

library(gmodels)

knn <- knn(train = train\_s,test = test\_s ,cl = train\_lables,k = 64)

CrossTable(x=test\_lables,y=knn,prop.chisq = FALSE )

Now the result with K=64 will be:



The test data consisted of 60000 observations. Out of which 49611 cases have been accurately predicted (TN->True Negatives) as zero (NO) in nature which constitutes 82.7%. Also, 471 out of 60000 observations were accurately predicted (TP-> True Positives) as One (Yes) in nature which constitutes 8%. Thus a total of 471 out of 60000 predictions where TP i.e, True Positive in nature.

The FN’s if any poses a potential threat for the same reason and the main focus to increase the accuracy of the model is to reduce FN’s.

There were 4932 cases of False Positives (FP) meaning 4932 cases were actually benign in nature but got predicted as One(yes).

The total accuracy of the model is 90.7 %((TN+TP)/60000) which shows that there may be chances to improve the model performance.

* **Improve the performance of the model**

This can be taken into account by repeating the steps 3 and 4 and by changing the k-value. Generally, it is the square root of the observations and in this case, we took k=64.The k-value may be fluctuated in and around the value of 64 to check the increased accuracy of the model. We will try it out with values of your choice to increase the accuracy! Also, we need remembering, to keep the value of FN’s as low as possible.

But we should remember that Classification is sensitive to the correct selection of k

small k? → less stable, influenced by noise

larger k? → less precise, higher bias

**4-** **Conclusions**

the analysis of the dataset has been a laborious task from its size and the little intuitiveness of its values could not be seen at first view the most differentiating variables addition the great time that has taken the cleaning itself of the dataset which has taken most of the time since many tests have been done to finally stay with the most important variables,

In addition, the numerous problems obtained to obtain the models have been added, since the vast majority have taken several hours its execution which prevented that there was much room to test the different possibilities as for example in the case that has been attempted apply ***KNN*** but because it took more than 4 hours,

Finally, it should be noted that the model that has yielded the best results has been that of OneR. The problem of the dataset has been mainly the great imbalance between all values.

**5- Bibliography**

* [**http://www.saedsayad.com/oner.htm**](http://www.saedsayad.com/oner.htm)
* [**https://www.saedsayad.com/k\_nearest\_neighbors.htm**](https://www.saedsayad.com/k_nearest_neighbors.htm)
* [**https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/**](https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/)
* [**https://cran.rproject.org/web/packages/OneR/vignettes/OneR.html**](https://cran.rproject.org/web/packages/OneR/vignettes/OneR.html)
* [**https://www.tutorialspoint.com/r/r\_histograms.htm**](https://www.tutorialspoint.com/r/r_histograms.htm)
* [**https://www.rdocumentation.org/packages/psych/versions/1.8.12/topics/multi.hist**](https://www.rdocumentation.org/packages/psych/versions/1.8.12/topics/multi.hist)