IS 777: Data Analytics

Deliverable 4
Resampling

Group B
Arjun Kumar
Milind Pawar
Sneha Rachkar
Pranitha Lolla
Vishal Sarvagod

Topic: COVID-19 patient condition

Description:

One of the biggest challenges that healthcare providers have encountered over the entire course of the pandemic is the lack of medical services without an adequate strategy to deliver them effectively. As the COVID-19 curve has tilted very unpredictably, they have been in the dark not understanding how much resource they can even in the very next week. In these trying times, it would be a great advantage to the authorities to be able to foresee what kind of resource a person would need at the time of being confirmed positive or even before that, and they will be able to obtain and arrange the services required to save the life of that patient.

The data set Link :- https://www.kaggle.com/tanmoyx/covid19-patient-precondition-dataset which is released by Mexican government on https://www.gob.mx/salud/documentos/datos- abiertos-bases-historicas-direccion-general-de-epidemiologia

Number of columns in this data set are 23
Number of rows in this data set are 56602
Size of data set is 43.8 MB

N number of observations: 56602
P number of Predictor variables: 18
Response variable: 1

Attribute Description:

Variable Name	**Description**	**Value Type**	**Predictor/ Response variable**
id	Case identifier number	TEXT,	
sex	Identify the sex of the patient.	Categorical Value [1,2]	P
patient_type	Sentinel surveillance is carried out	Categorical Value	P
	through the system of respiratory disease monitoring units (USMER). The USMER includes medical units of the first, second or	[1,2]	

	third level of care, and third level units also participate as USMERs, which due to their characteristics contribute to broadening the epidemiological information panorama, including those with a specialty in pulmonology, infectiology or pediatrics . (Categories in Annex Catalog).		
entry_date	Identify the date of admission of the patient to the care unit.	Timestamp	
date_symptoms date_died	Identifies the date on which the patient's symptoms began. Identify the date the patient died.	Timestamp	P R
intubed	Identify if the patient required	Timestamp Categorical Value	R
Pneumonia	intubation. Identify if the patient was diagnosed with pneumonia.	[1,2,97] 97 :missing data. Categorical Value [1,2,97]	P
Age	Identify the age of the patient.	97 :missing data. Numeric Value	P
pregnancy	Identify if the patient is pregnant.	Categorical Value [1,2,97]	P
diabetes	Identify if the patient has a diagnosis of diabetes.	97 :missing data. Categorical Value [1,2]	P
copd	Identify if the patient has a diagnosis of COPD.	Categorical Value [1,2]	P
asthma	Identify if the patient has	Categorical Value	P
	a diagnosis of asthma.	[1,2]	
inmsupr	Identify if the patient is immunosuppressed.	Categorical Value [1,2]	P

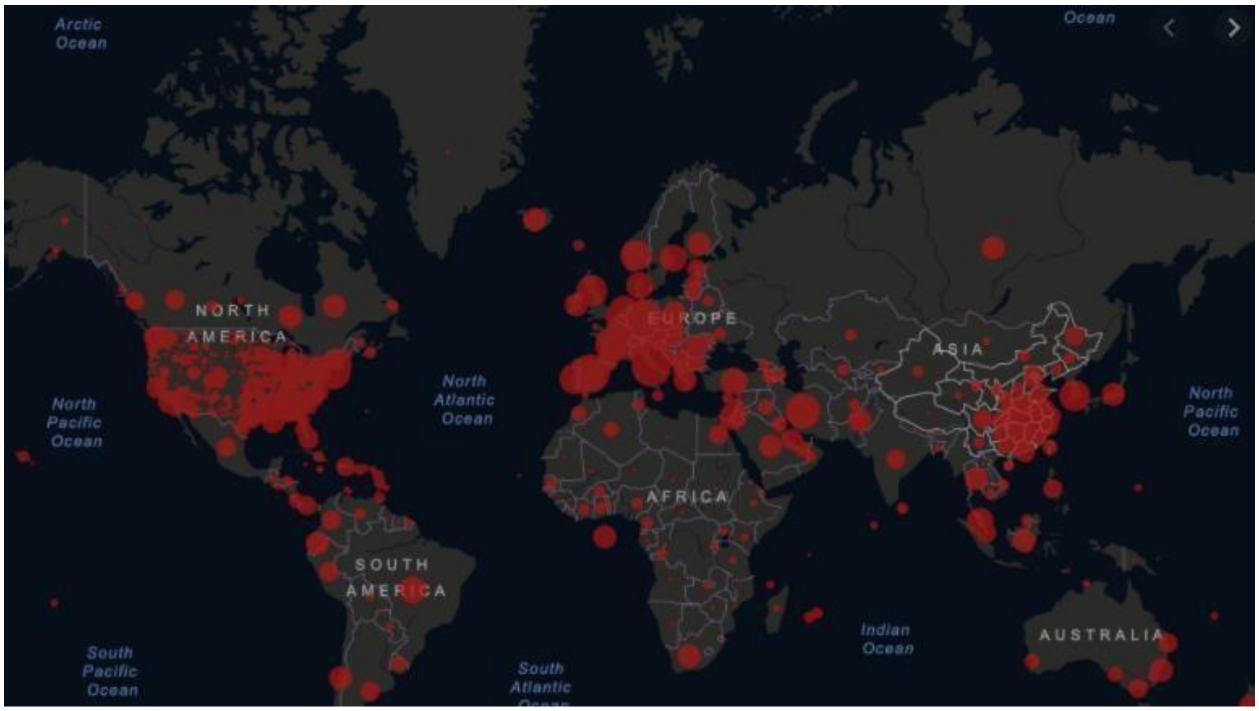
hypertension	Identify if the patient has a diagnosis of hypertension.	Categorical Value [1,2]	P
other_disease	Identify if the patient has a diagnosis of other diseases.	Categorical Value [1,2,98] 98 :missing data.	P
cardiovascular	Identify if the patient has a diagnosis of cardiovascular disease.	Categorical Value [1,2,98] 98 :missing data.	P
obesity	Identify if the patient has a diagnosis of obesity.	Categorical Value [1,2,98] 98 :missing data.	P
renal_chronic	Identify if the patient has a diagnosis of chronic kidney failure.	Categorical Value [1,2,98] 98 :missing data.	P
tobacco	Identify if the patient has a smoking habit.	Categorical Value [1,2,98] 98 :missing data.	P
contact_other_covid	Identify if the patient had contact with any other case diagnosed with SARS CoV-2	Categorical Value [1,2,99] 99 :missing data.	
covid_res	Identifies the result of the analysis of the sample reported by the laboratory of the National Network of Epidemiological Surveillance Laboratories (INDRE, LESP and LAVE). (Catalog of diagnostic results attached).	Categorical Value [1,2]	R
icu	Identify if the patient required admission to an Intensive Care Unit.	Categorical Value [1,2,97] 97 :missing data.	R

Predictive Models:

Based on data set of COVID-19 patient pre-history, healthcare provider can easily predict the which categories of patient needs useful resources based on their health history. Variable date symptoms, age, contact_other_covid and covid_res can be used to predict if patient result is positive then did patient is with contact with other and which age category is most vulnerable. Based on age and other health history such as diabetes, asthma cardiovascular it can be predicted that these patients need more useful resource and attention.

Significance of study:

This data set can be used to predict how much resource of healthcare can utilize on different patient categories based on age, health history. This can be used to alert those people who are not yet COVID positive, but they have similar health history.



Data After Cleaning

* n (number of observations): 562647
* Response Variable: 1

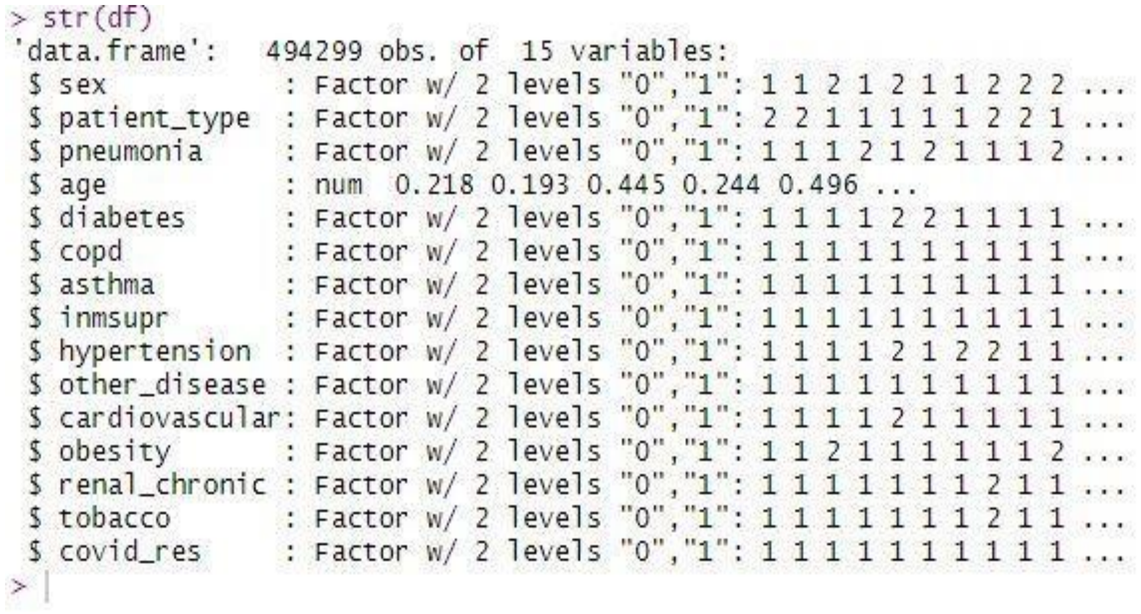
Covid Result

* Predictor Variables: 14

1. **Sex** : (Categorical Value) Identify the sex of the patient .

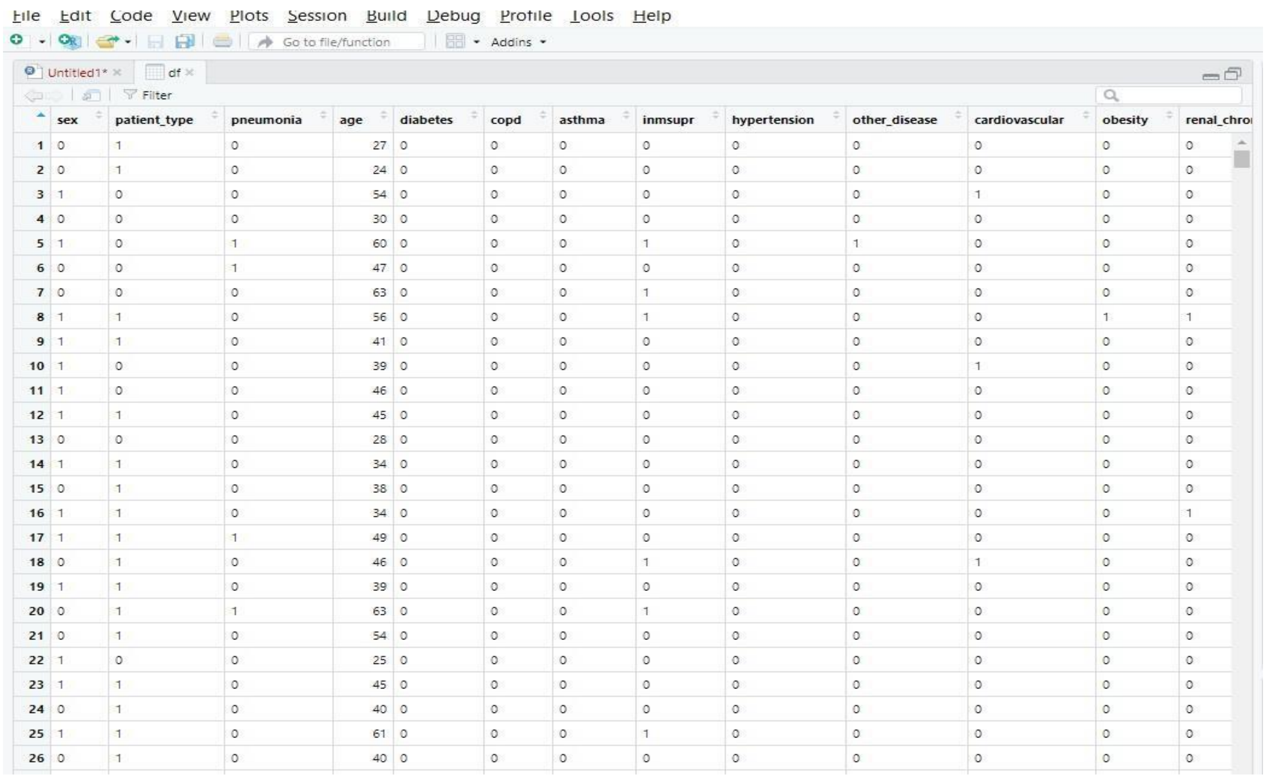
2. **patient_type** : (Categorical Value) Sentinel surveillance is carried out through the system of respiratory disease monitoring units (USMER). The USMER includes medical units of the first, second or third level of care, and third level units also participate as USMERs, which due to their characteristics contribute to broadening the epidemiological information panorama, including those with a specialty in pulmonology, infectology or pediatrics.
3. **Age**: (Numeric Value) Identify the age of the patient
4. **Pneumonia:** (Categorical Value) Identify if the patient was diagnosed with pneumonia
5. **Diabetes:** (Categorical Value) Identify if the patient has a diagnosis of diabetes
6. **Copd:** (Categorical Value) Identify if the patient has a diagnosis of COPD
7. **Asthma:** (Categorical Value) Identify if the patient has a diagnosis of asthma.
8. **Inmsupr:** (Categorical Value) Identify if the patient is immunosuppressed
9. **Hypertension:** (Categorical Value) Identify if the patient has a diagnosis of hypertension
10. **other_disease:** (Categorical Value) Identify if the patient has a diagnosis of other diseases
11. **Cardiovascular:** (Categorical Value) Identify if the patient has a diagnosis of cardiovascular disease
12. **Obesity:** (Categorical Value) Identify if the patient has a diagnosis of obesity
13. **Renal_chronic:** (Categorical Value) Identify if the patient has a diagnosis of chronic kidney failure
14. **Tobacco:** (Categorical Value) Identify if the patient has a smoking habit

* Descriptive Analysis:



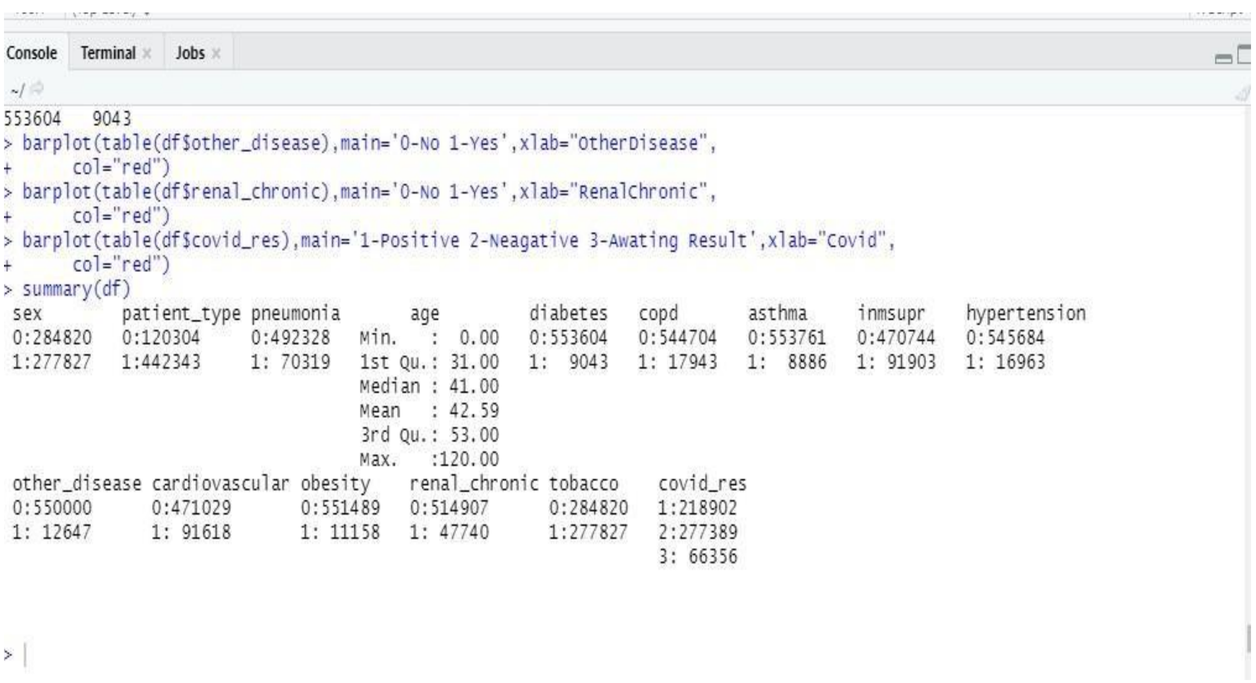
```
> str(df)
'data.frame':   494299 obs. of  15 variables:
 $ sex           : Factor w/ 2 levels "0","1": 1 1 2 1 2 1 1 2 2 2 ...
 $ patient_type  : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 2 2 1 ...
 $ pneumonia     : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 1 1 2 ...
 $ age           : num  0.218 0.193 0.445 0.244 0.496 ...
 $ diabetes      : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 1 1 ...
 $ copd          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ asthma        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ inmsupr       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ hypertension  : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 2 2 1 1 ...
 $ other_disease : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ cardiovascular: Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
 $ obesity       : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 1 1 2 ...
 $ renal_chronic : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ tobacco       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ covid_res     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
> |
```

Categorical fields

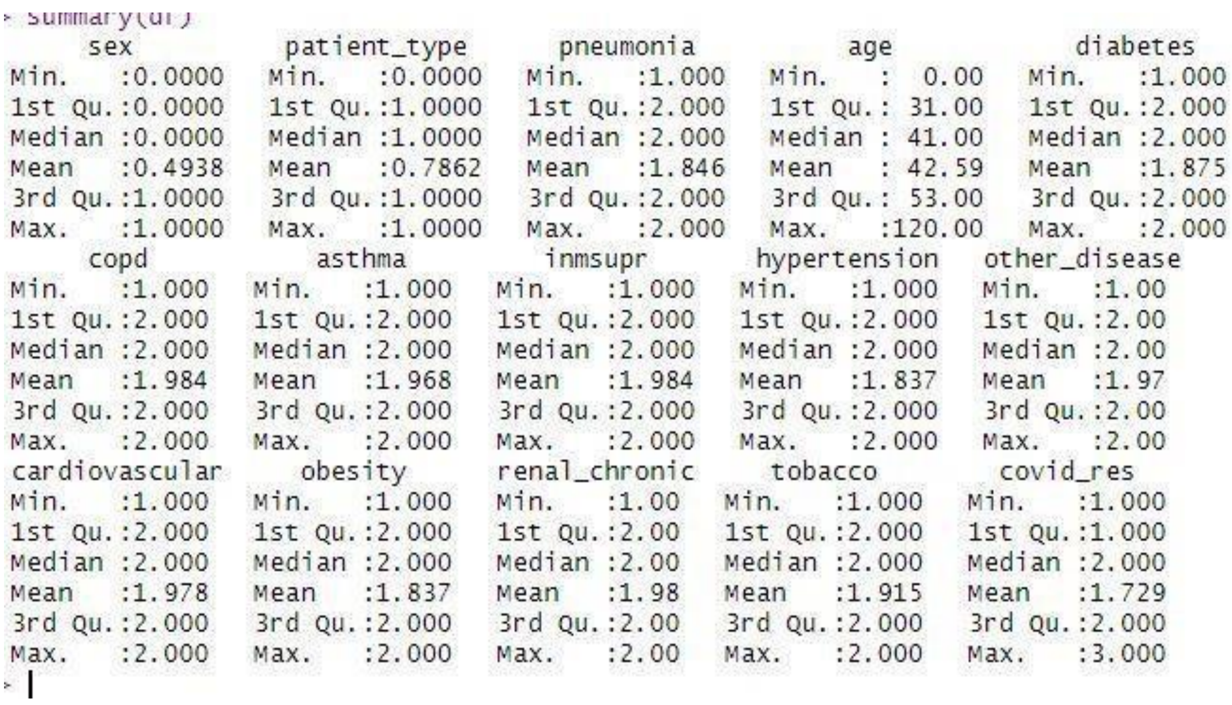

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Untitled1*
df
Filter

	sex	patient_type	pneumonia	age	diabetes	copd	asthma	inmsupr	hypertension	other_disease	cardiovascular	obesity	renal_chro
1	0	1	0	27	0	0	0	0	0	0	0	0	0
2	0	1	0	24	0	0	0	0	0	0	0	0	0
3	1	0	0	54	0	0	0	0	0	0	1	0	0
4	0	0	0	30	0	0	0	0	0	0	0	0	0
5	1	0	1	60	0	0	0	1	0	1	0	0	0
6	0	0	1	47	0	0	0	0	0	0	0	0	0
7	0	0	0	63	0	0	0	1	0	0	0	0	0
8	1	1	0	56	0	0	0	1	0	0	0	1	1
9	1	1	0	41	0	0	0	0	0	0	0	0	0
10	1	0	0	39	0	0	0	0	0	0	1	0	0
11	1	0	0	46	0	0	0	0	0	0	0	0	0
12	1	1	0	45	0	0	0	0	0	0	0	0	0
13	0	0	0	28	0	0	0	0	0	0	0	0	0
14	1	1	0	34	0	0	0	0	0	0	0	0	0
15	0	1	0	38	0	0	0	0	0	0	0	0	0
16	1	1	0	34	0	0	0	0	0	0	0	0	1
17	1	1	1	49	0	0	0	0	0	0	0	0	0
18	0	1	0	46	0	0	0	1	0	0	1	0	0
19	1	1	0	39	0	0	0	0	0	0	0	0	0
20	0	1	1	63	0	0	0	1	0	0	0	0	0
21	0	1	0	54	0	0	0	0	0	0	0	0	0
22	1	0	0	25	0	0	0	0	0	0	0	0	0
23	1	1	0	45	0	0	0	0	0	0	0	0	0
24	0	1	0	40	0	0	0	0	0	0	0	0	0
25	1	1	0	61	0	0	0	1	0	0	0	0	0
26	0	1	0	40	0	0	0	0	0	0	0	0	0

** Summary statistics obtained from R for each variable. These include mean, median, and quartiles along with some other statistics



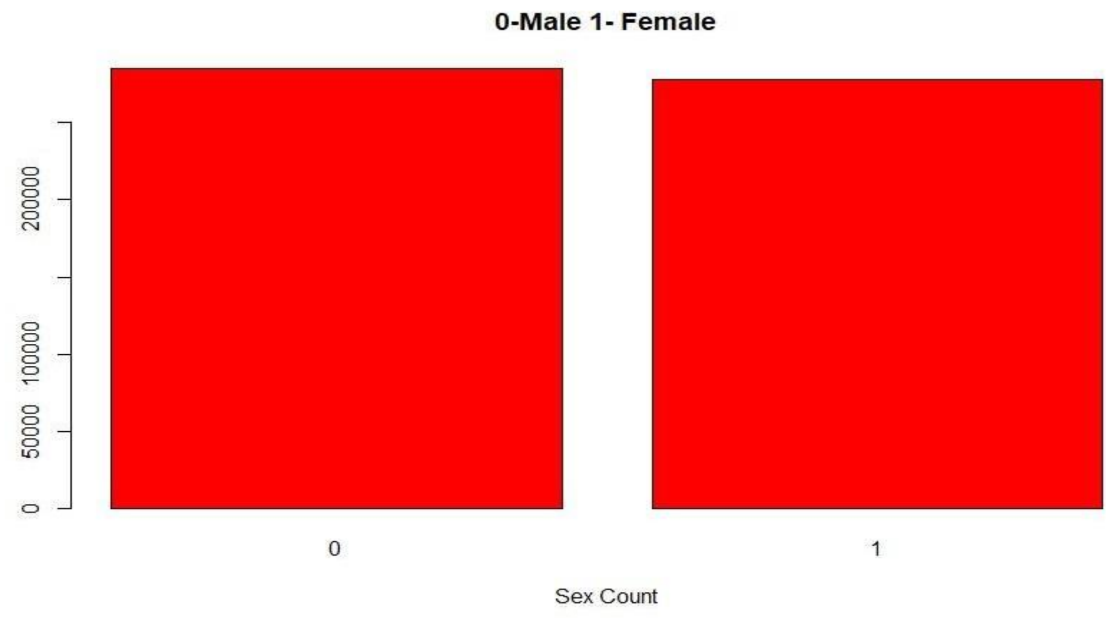
```
553604   9043
> barplot(table(df$other_disease),main='0-No 1-Yes',xlab="OtherDisease",
+       col="red")
> barplot(table(df$renal_chronic),main='0-No 1-Yes',xlab="RenalChronic",
+       col="red")
> barplot(table(df$covid_res),main='1-Positive 2-Neagative 3-Awating Result',xlab="Covid",
+       col="red")
> summary(df)
 sex      patient_type pneumonia       age          diabetes   copd       asthma     inmsupr    hypertension
 0:284820  0:120304     0:492328   Min.   :  0.00   0:553604   0:544704   0:553761   0:470744   0:545684    
 1:277827  1:442343     1: 70319   1st Qu.: 31.00   1:  9043   1: 17943   1:  8886   1: 91903   1: 16963    
                                   Median : 41.00                                                            
                                   Mean   : 42.59                                                            
                                   3rd Qu.: 53.00                                                            
                                   Max.   :120.00                                                            
 other_disease cardiovascular obesity    renal_chronic tobacco    covid_res
 0:550000      0:471029       0:551489   0:514907      0:284820   1:218902 
 1: 12647      1: 91618       1: 11158   1: 47740      1:277827   2:277389 
                                                                  3: 66356 



> |
```

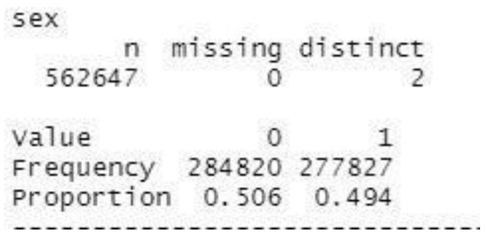


```
> summary(df)
      sex          patient_type      pneumonia          age            diabetes    
 Min.   :0.0000   Min.   :0.0000   Min.   :1.000   Min.   :  0.00   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:2.000   1st Qu.: 31.00   1st Qu.:2.000  
 Median :0.0000   Median :1.0000   Median :2.000   Median : 41.00   Median :2.000  
 Mean   :0.4938   Mean   :0.7862   Mean   :1.846   Mean   : 42.59   Mean   :1.875  
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.: 53.00   3rd Qu.:2.000  
 Max.   :1.0000   Max.   :1.0000   Max.   :2.000   Max.   :120.00   Max.   :2.000  
      copd           asthma          inmsupr       hypertension   other_disease 
 Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.00  
 1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.00  
 Median :2.000   Median :2.000   Median :2.000   Median :2.000   Median :2.00  
 Mean   :1.984   Mean   :1.968   Mean   :1.984   Mean   :1.837   Mean   :1.97  
 3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.00  
 Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :2.00  
 cardiovascular     obesity      renal_chronic     tobacco        covid_res    
 Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   :1.000   Min.   :1.000  
 1st Qu.:2.000   1st Qu.:2.000   1st Qu.:2.00   1st Qu.:2.000   1st Qu.:1.000  
 Median :2.000   Median :2.000   Median :2.00   Median :2.000   Median :2.000  
 Mean   :1.978   Mean   :1.837   Mean   :1.98   Mean   :1.915   Mean   :1.729  
 3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.00   3rd Qu.:2.000   3rd Qu.:2.000  
 Max.   :2.000   Max.   :2.000   Max.   :2.00   Max.   :2.000   Max.   :3.000  
> |
```

Histograms for quantitative variables and barcharts for the qualitative variables all produced in R

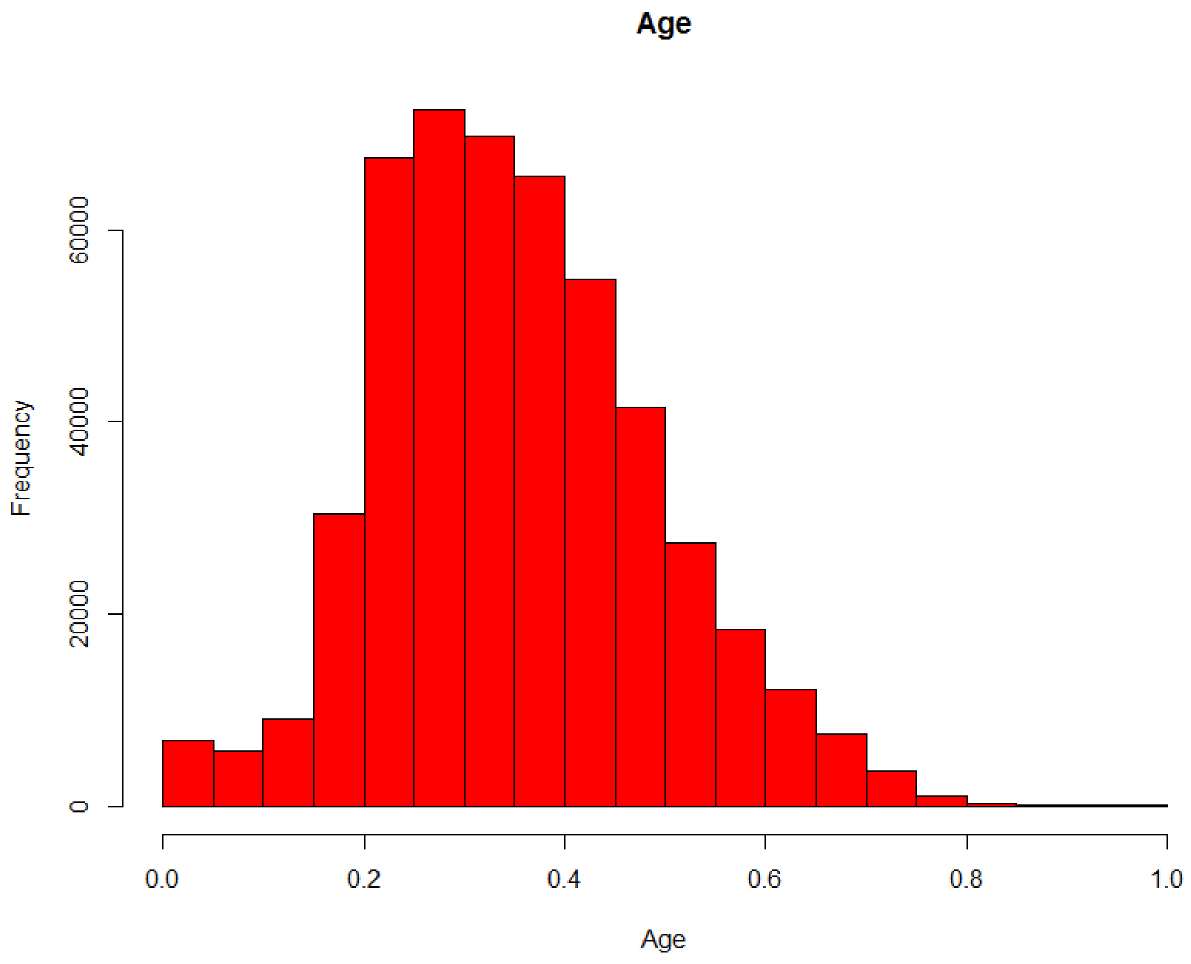
Sex

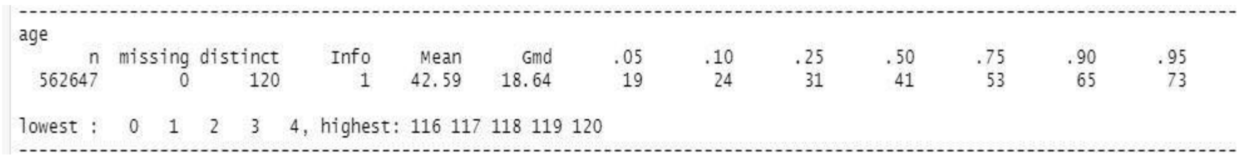




```
sex
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  284820 277827
Proportion  0.506  0.494
------------------------------
```

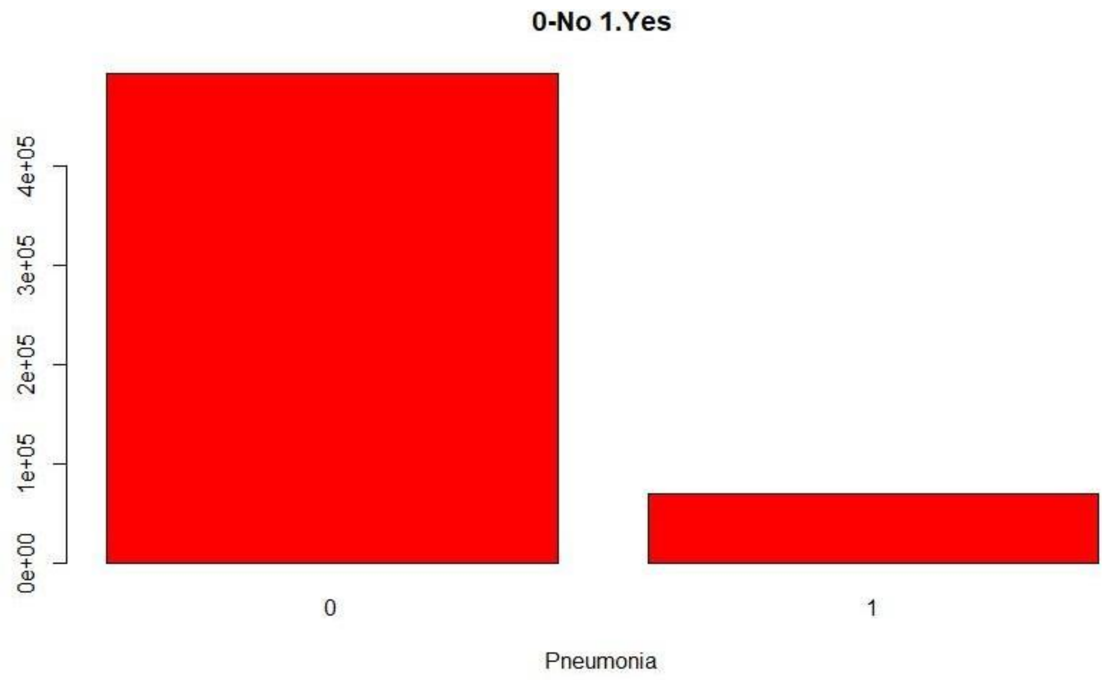
Age

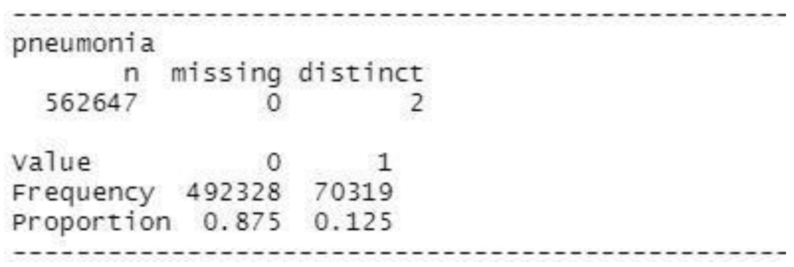




```
--------------------------------------------------------------------------------------------------------
age
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75      .90      .95
  562647        0      120        1    42.59    18.64       19       24       31       41       53       65       73

lowest :   0   1   2   3   4, highest: 116 117 118 119 120
--------------------------------------------------------------------------------------------------------
```

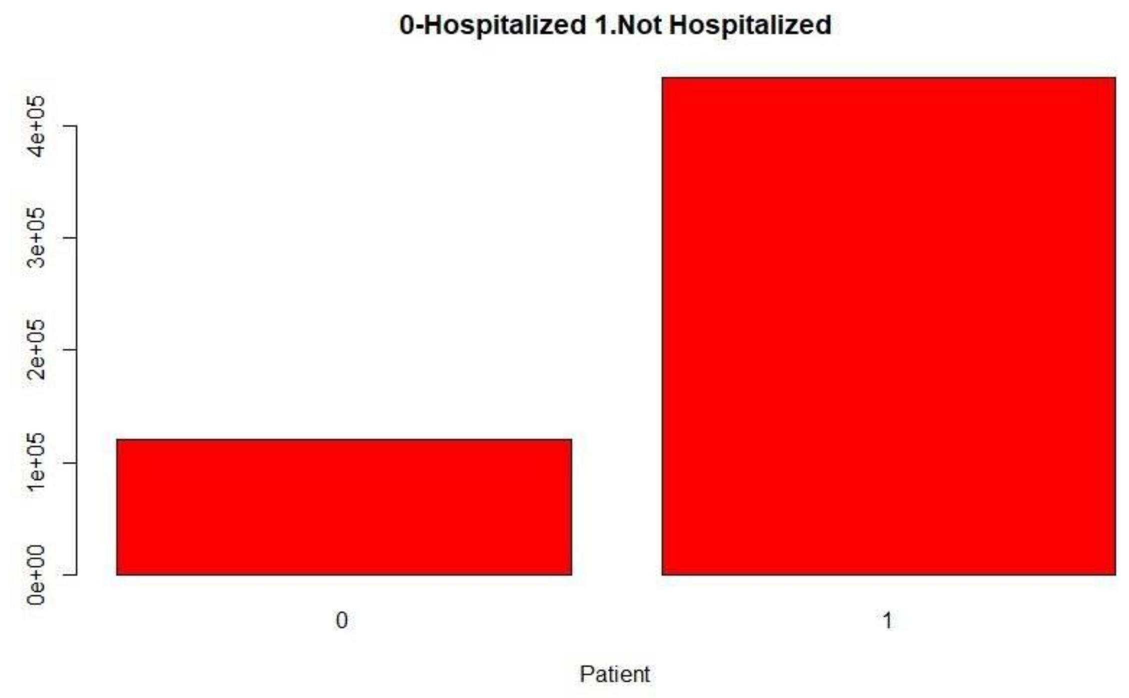
Pneumonia

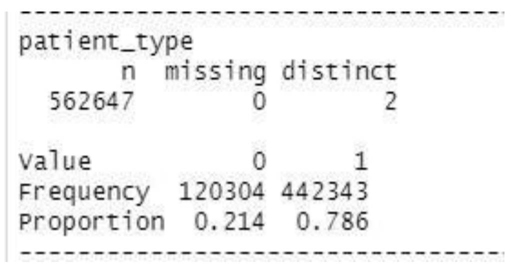




```
-----------------------------------------------
pneumonia
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  492328  70319
Proportion  0.875  0.125
-----------------------------------------------
```

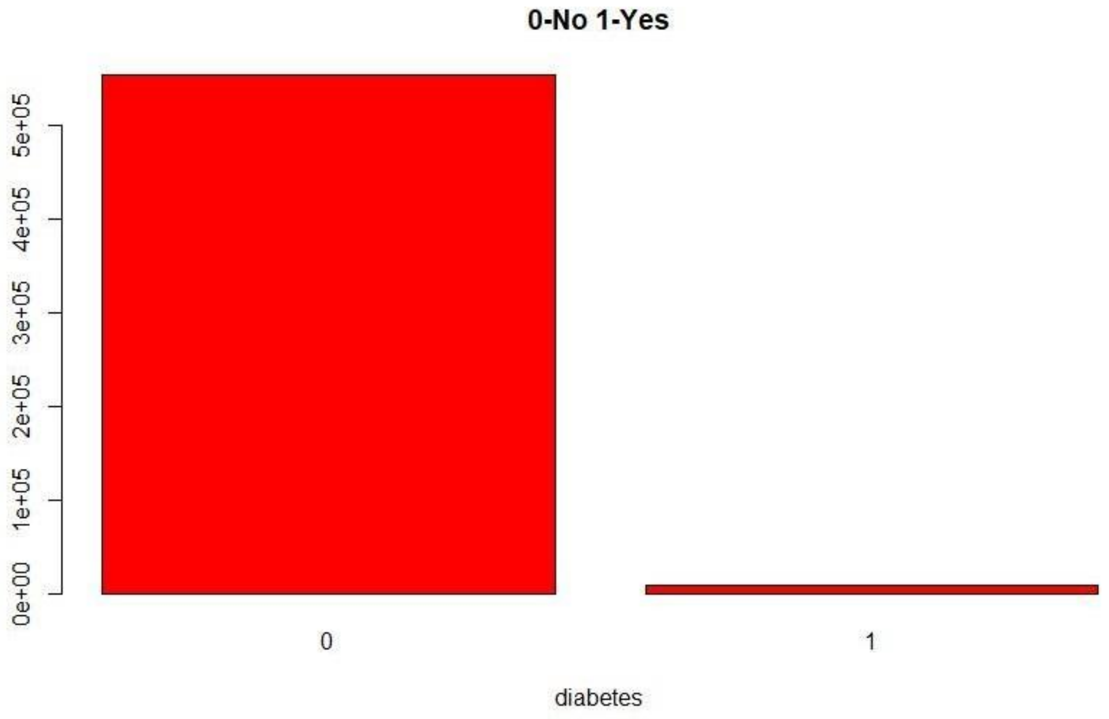
Patient

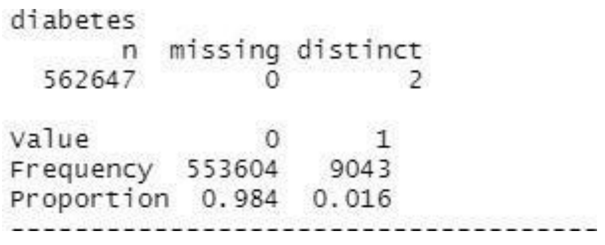




```
---------------------------------
patient_type
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  120304 442343
Proportion  0.214  0.786
---------------------------------
```

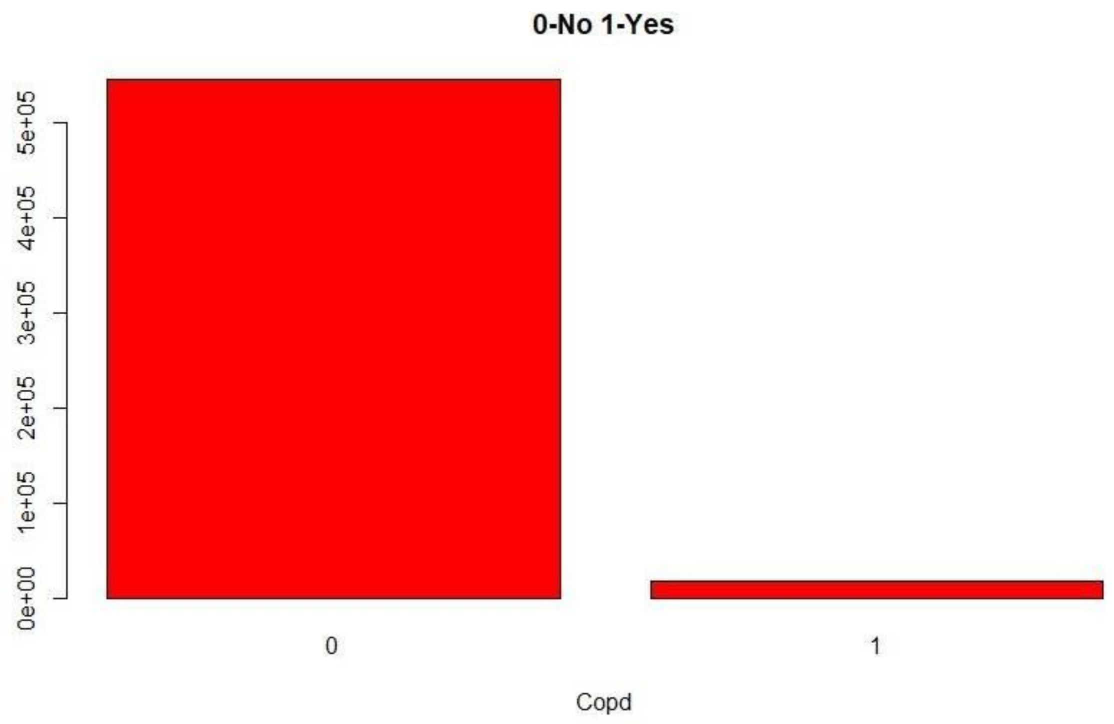
Diabetes

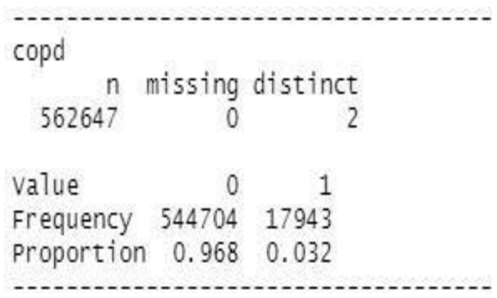




```
diabetes
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  553604   9043
Proportion  0.984  0.016
------------------------------------
```

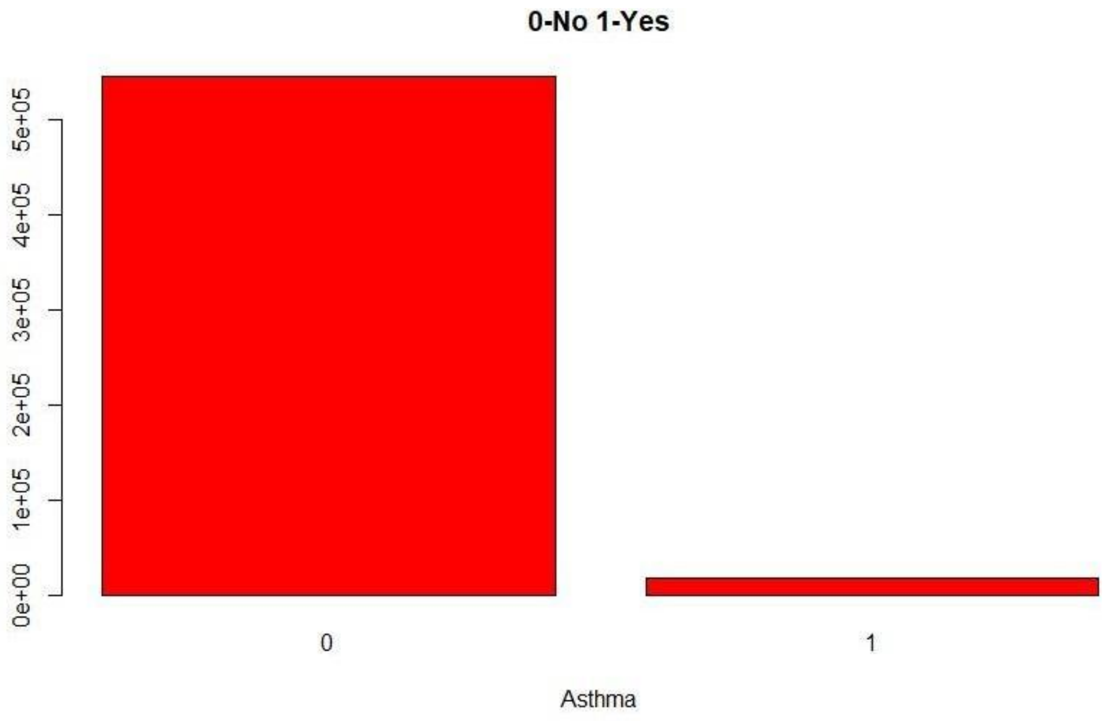
Copd:

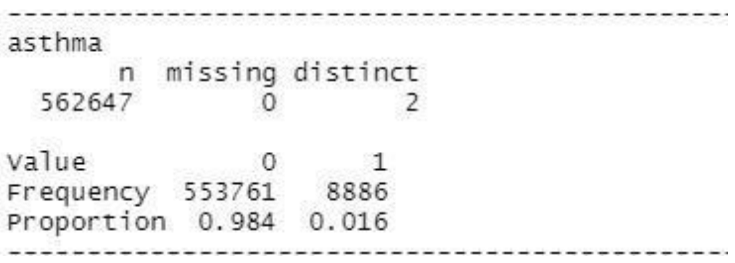




```
------------------------------------
copd
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  544704  17943
Proportion  0.968  0.032
------------------------------------
```

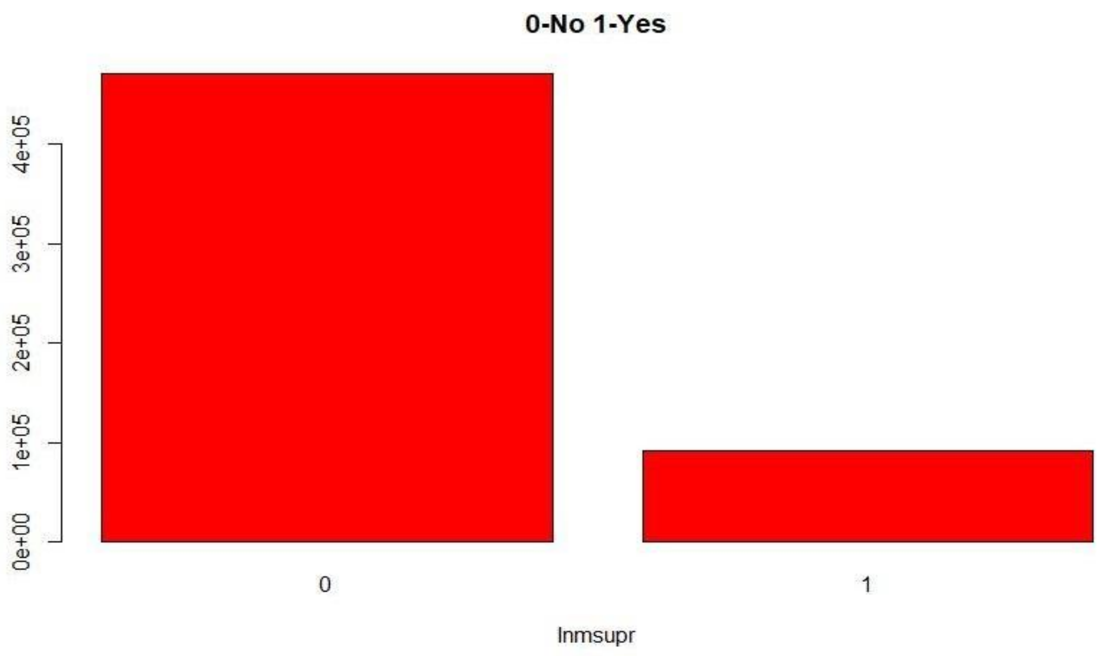
Asthma

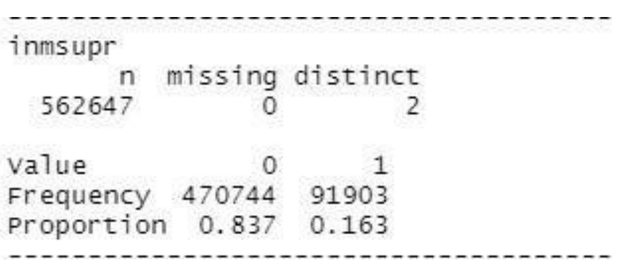




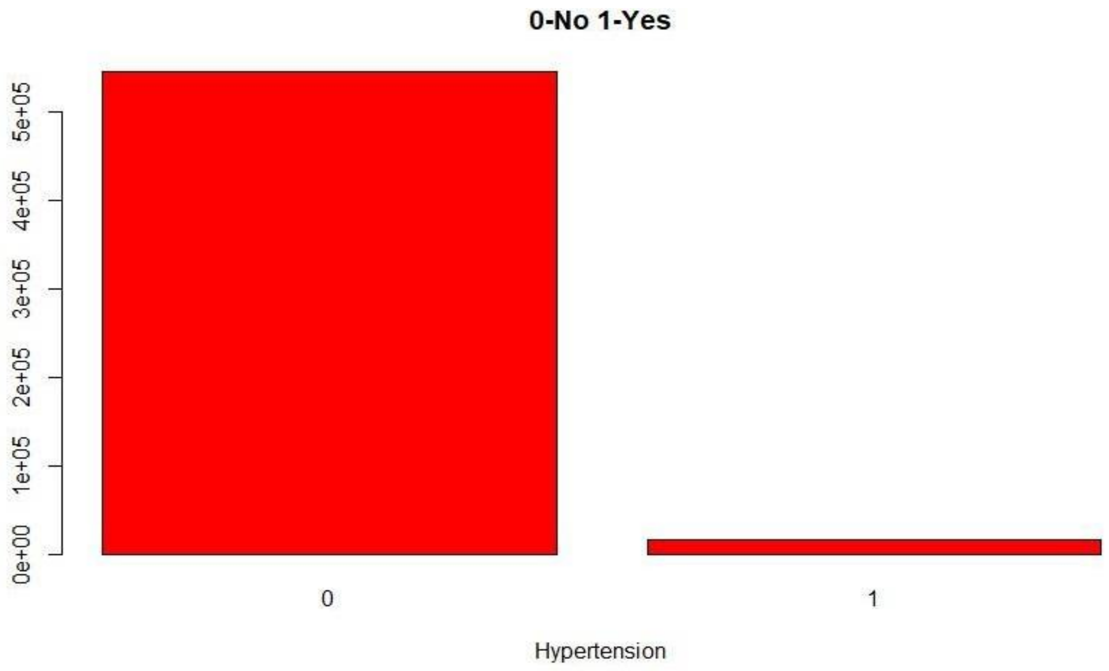
```
---------------------------------------------
asthma
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  553761   8886
Proportion  0.984  0.016
---------------------------------------------
```

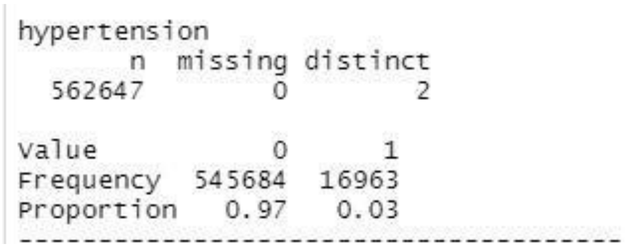
Inmsupr




```
--------------------------------------
inmsupr
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  470744  91903
Proportion  0.837  0.163
--------------------------------------
```

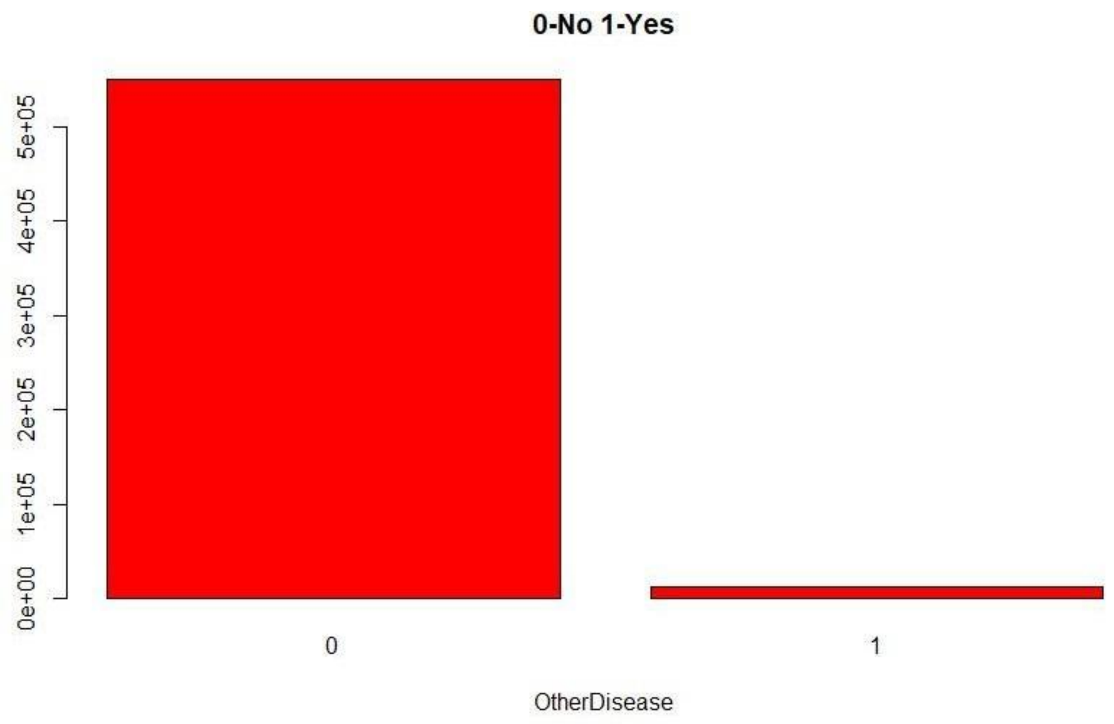
Hypertension

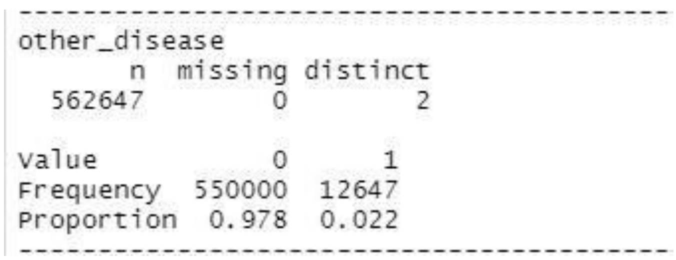




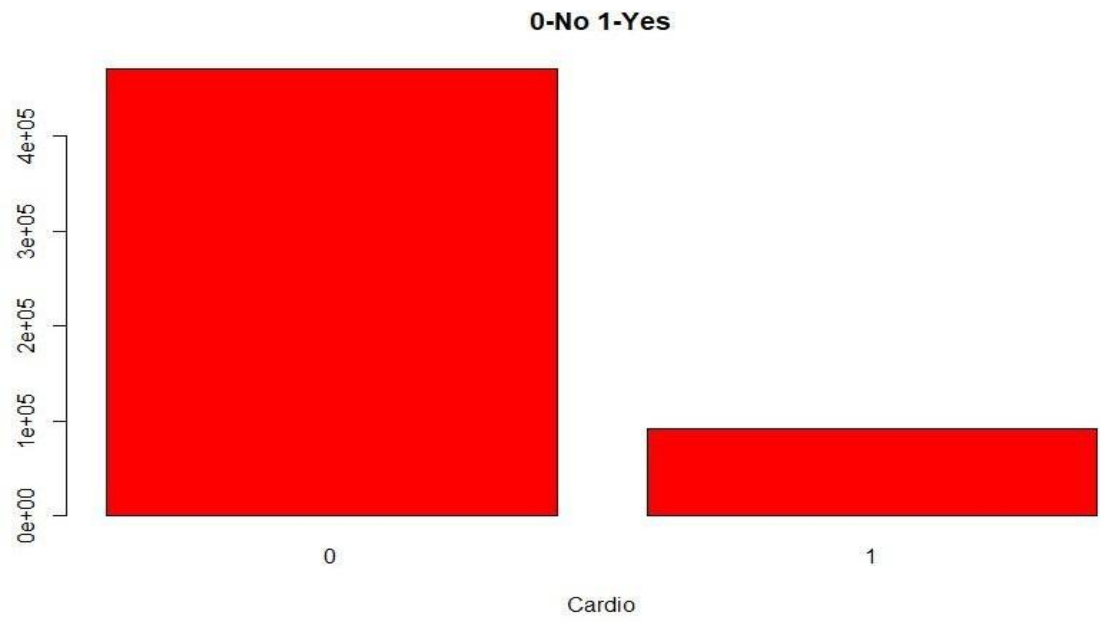
```
hypertension
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  545684  16963
Proportion   0.97   0.03
--------------------------------------
```

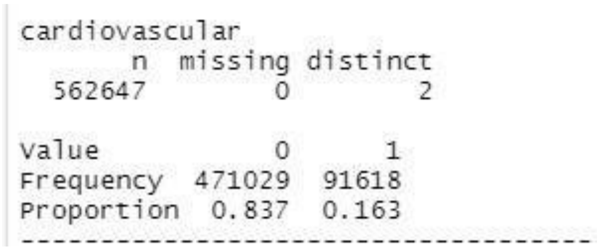
Other Diseases




```
-----------------------------------------
other_disease
       n  missing distinct
  562647        0        2

Value          0      1
Frequency  550000  12647
Proportion  0.978  0.022
-----------------------------------------
```

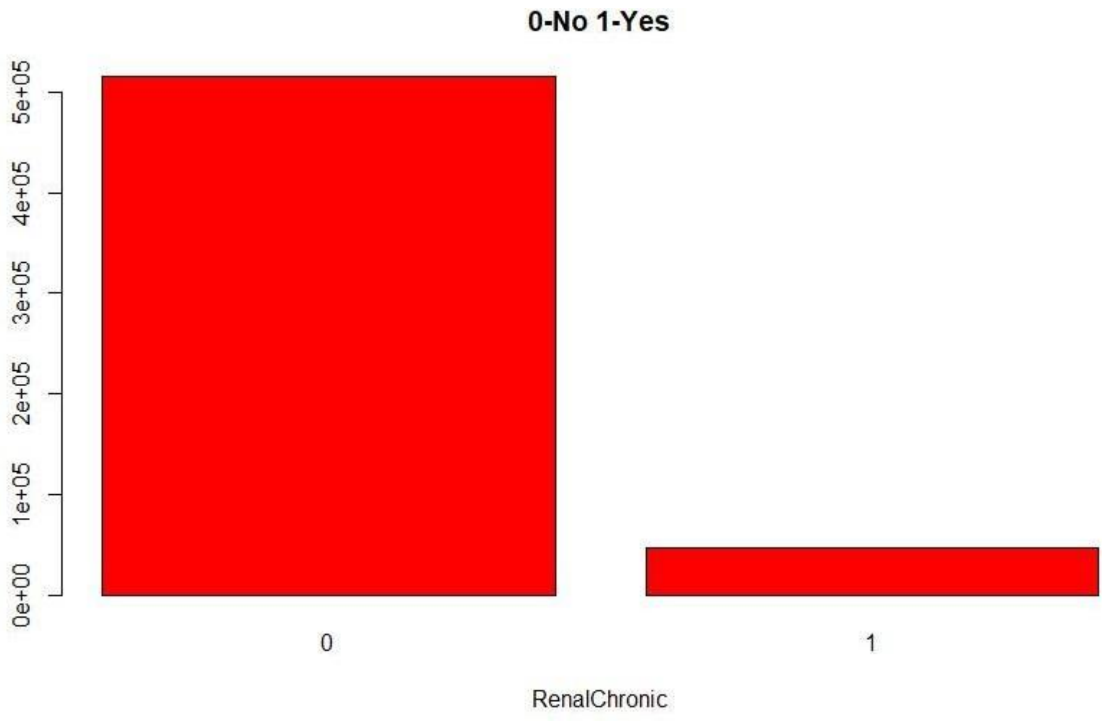
Cardiovascular

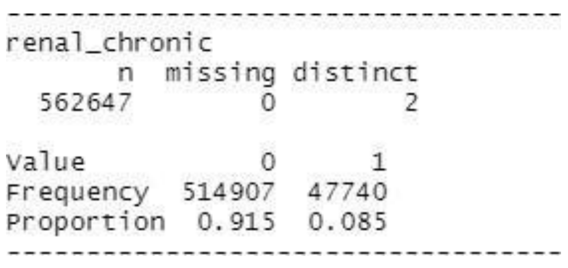




```
cardiovascular
       n  missing distinct
  562647        0        2

Value            0       1
Frequency   471029   91618
Proportion   0.837   0.163
------------------------------------
```

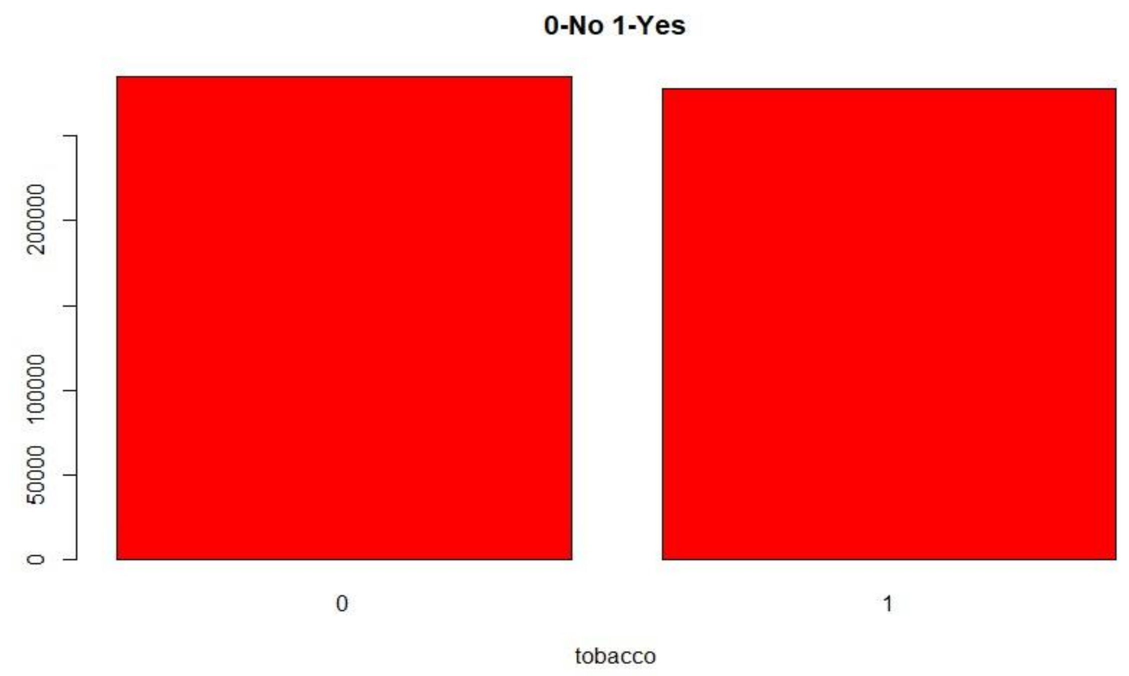
Renal Chronic

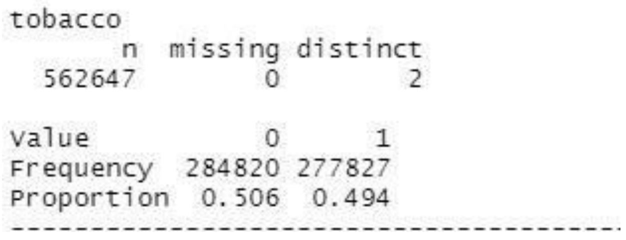




```
-----------------------------------
renal_chronic
       n  missing distinct
  562647        0        2

Value           0      1
Frequency  514907  47740
Proportion  0.915  0.085
-----------------------------------
```

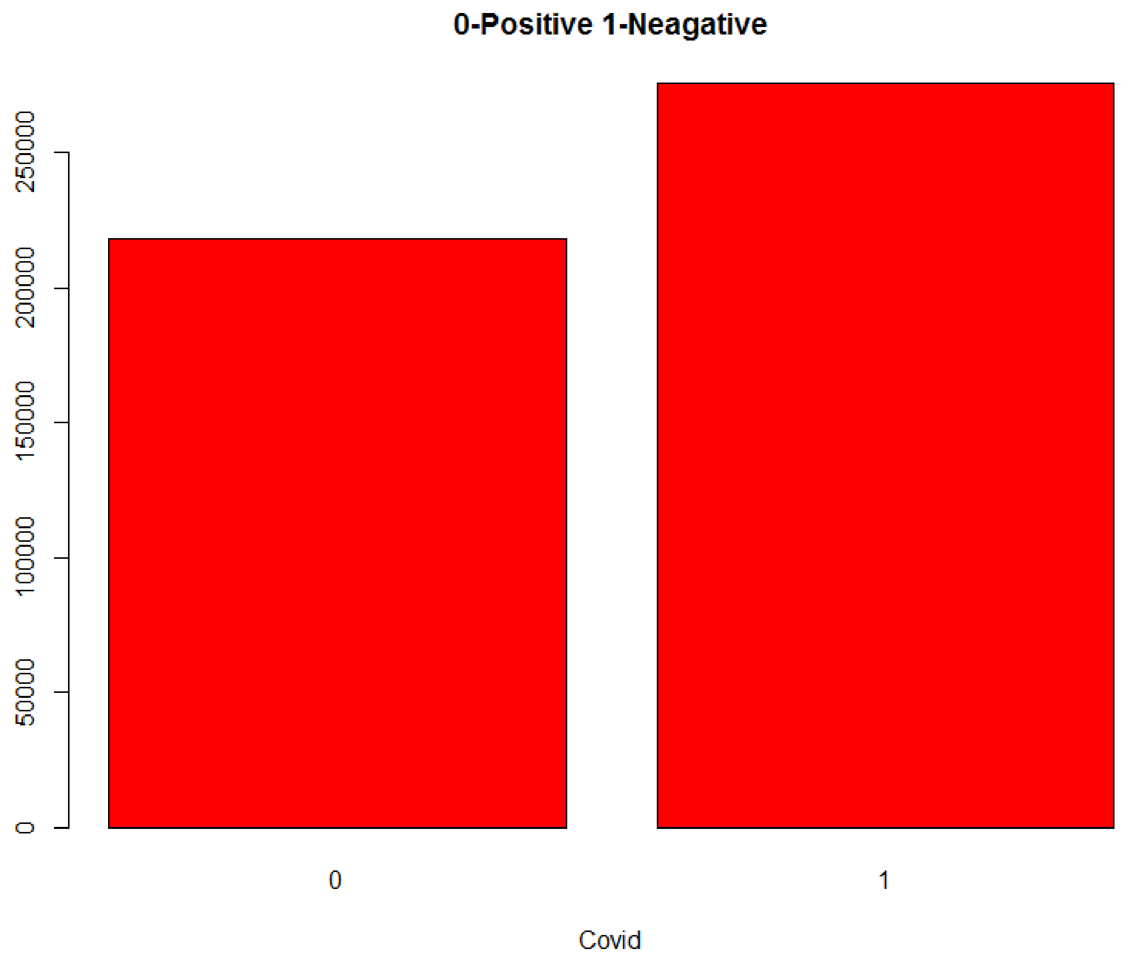
Tobacco

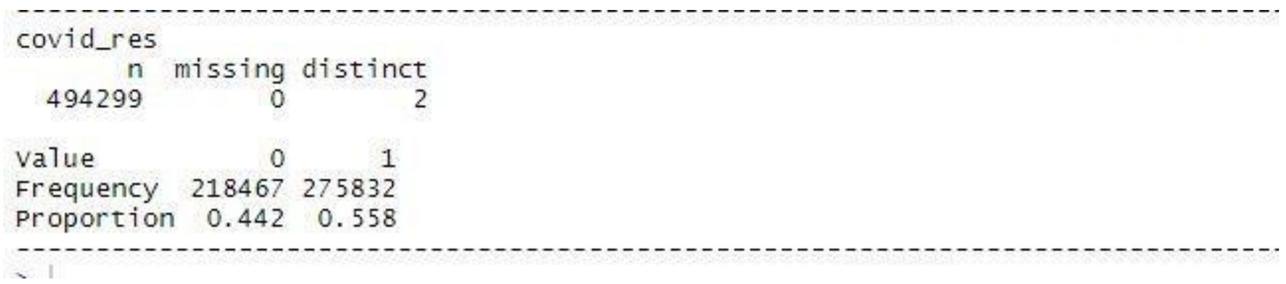




```
tobacco
       n  missing distinct 
  562647        0        2 

Value           0      1
Frequency  284820 277827
Proportion  0.506  0.494
---------------------------------------
```

Covid




```
------------------------------------------------------------------------------
covid_res
       n  missing distinct
  494299        0        2

value            0      1
Frequency   218467 275832
Proportion   0.442  0.558
------------------------------------------------------------------------------
```

* Analysis Plan: Discuss your current plan about how the rest of your analysis will proceed. What type of predictive models seem to be appropriate? What is the response variable? What are the predictors?

Currently, we have implemented data cleaning, data pre-processing and data preparation steps which would help for further analysis. We plan to use Logistic Regression Algorithm, Bayesian Model or

KNN for predictive modelling based on the accuracy and results achieved after implementing it on the training model and test data set. While implementing the algorithm for selecting one of the predictive models, we will analysis different attributes and various aspects of the data set like –

1. Does the gender of a patient relates to high chances of getting COVID-19?

2. Are the patients with previous cardiovascular disease or diabetes more likely to be affected by COVID-

19?

3. Whether tobacco consumption is related to COVID-19?

These analyses will be useful to select the predictive model. Variables that have the best performing model will be selected. At each stage, the worst-performing predictor will be removed until all the regressors left perform well. The Logistic Regression Model will be used to assess the accuracy of this model 's prediction by creating a confusion matrix on train results. Then, it is possible to verify the assumptions of the logistic regression model. The generalized linear model assumes linear relationships between continuous independent variables and the result variable logit that can be visualized between- continuous predictor and the logit values using a scatterplot.

Total No. of Predictor Variables =
14 Predictor Variables:

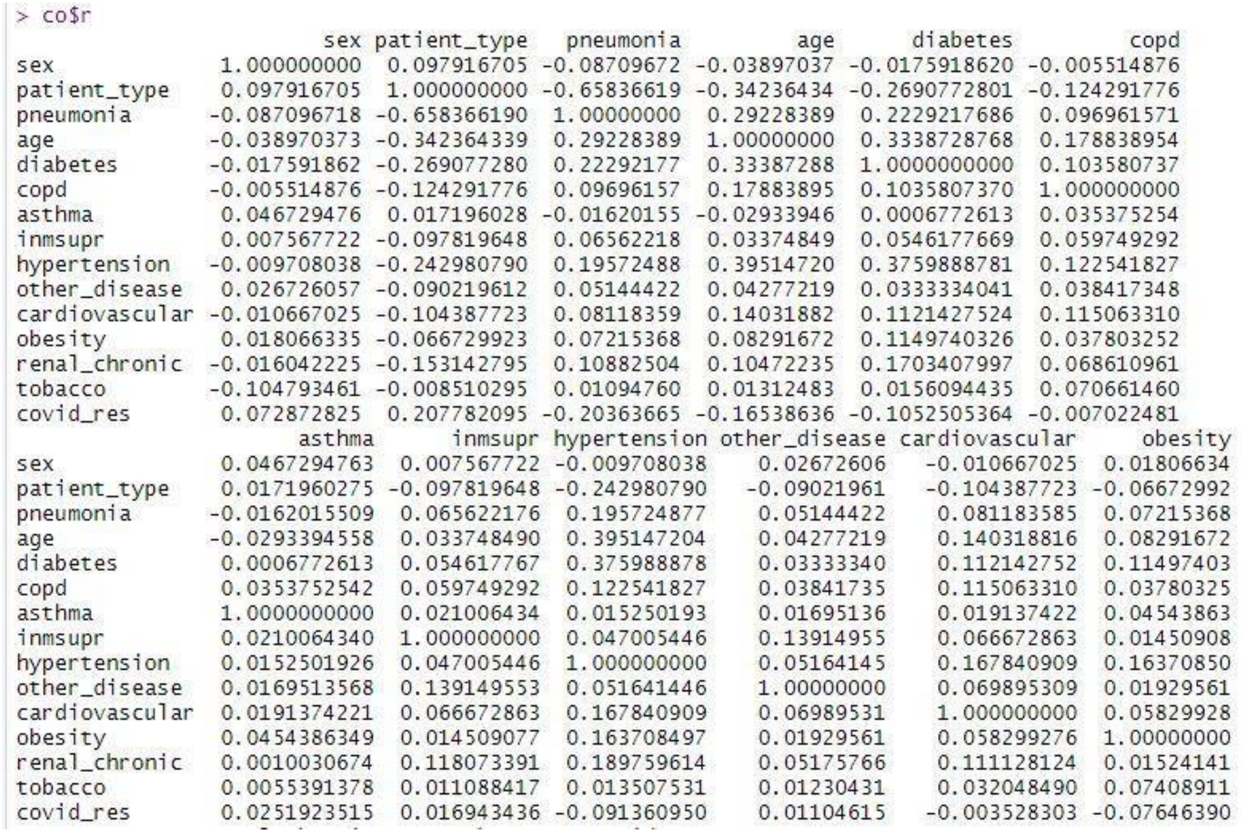
1. Sex
2. patient_type
3. Age
4. Pneumonia
5. Diabetes
6. Copd
7. Asthma
8. Inmsupr
9. Hypertension
10. other_disease
11. Cardiovascular
12. Obesity
13. Renal_chronic

14. Tobacco

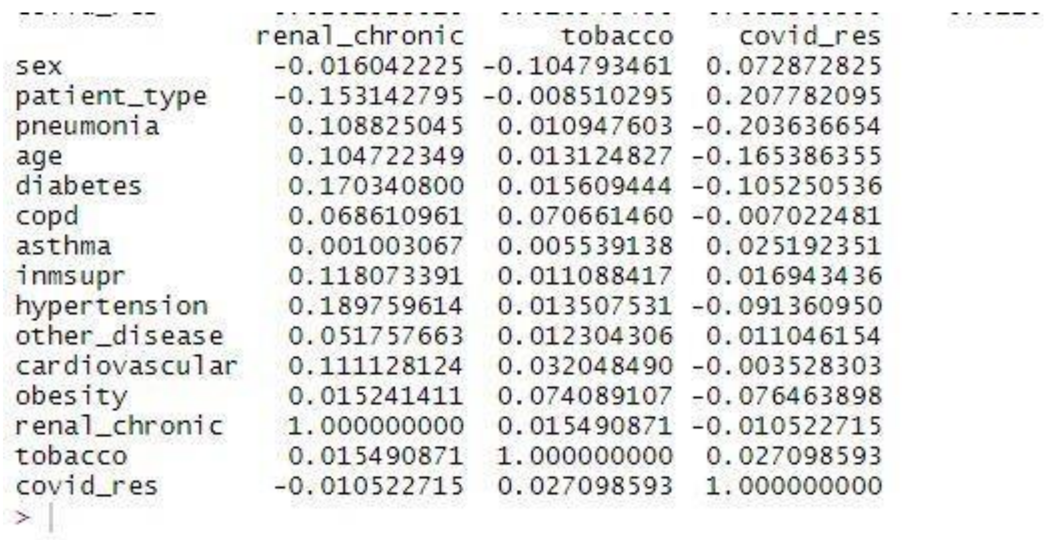
Total No. of Response Variables
= 1 Response Variables:

1. COVID Result

Correlation Matrix:



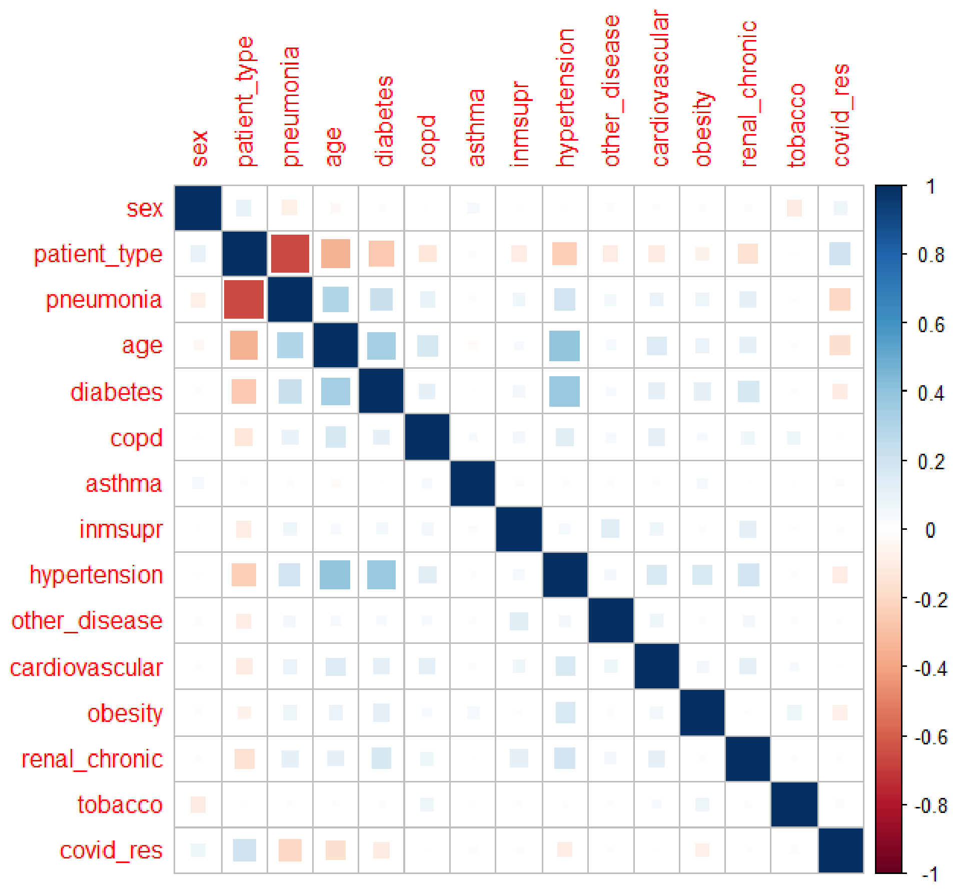
```
> co$r
                      sex patient_type   pneumonia         age      diabetes         copd
sex            1.000000000  0.097916705 -0.08709672 -0.03897037 -0.0175918620 -0.005514876
patient_type   0.097916705  1.000000000 -0.65836619 -0.34236434 -0.2690772801 -0.124291776
pneumonia     -0.087096718 -0.658366190  1.00000000  0.29228389  0.2229217686  0.096961571
age           -0.038970373 -0.342364339  0.29228389  1.00000000  0.3338728768  0.178838954
diabetes      -0.017591862 -0.269077280  0.22292177  0.33387288  1.0000000000  0.103580737
copd          -0.005514876 -0.124291776  0.09696157  0.17883895  0.1035807370  1.000000000
asthma         0.046729476  0.017196028 -0.01620155 -0.02933946  0.0006772613  0.035375254
inmsupr        0.007567722 -0.097819648  0.06562218  0.03374849  0.0546177669  0.059749292
hypertension  -0.009708038 -0.242980790  0.19572488  0.39514720  0.3759888781  0.122541827
other_disease  0.026726057 -0.090219612  0.05144422  0.04277219  0.0333334041  0.038417348
cardiovascular -0.010667025 -0.104387723  0.08118359  0.14031882  0.1121427524  0.115063310
obesity        0.018066335 -0.066729923  0.07215368  0.08291672  0.1149740326  0.037803252
renal_chronic -0.016042225 -0.153142795  0.10882504  0.10472235  0.1703407997  0.068610961
tobacco       -0.104793461 -0.008510295  0.01094760  0.01312483  0.0156094435  0.070661460
covid_res      0.072872825  0.207782095 -0.20363665 -0.16538636 -0.1052505364 -0.007022481
                     asthma      inmsupr hypertension other_disease cardiovascular     obesity
sex            0.0467294763  0.007567722 -0.009708038    0.02672606   -0.010667025  0.01806634
patient_type   0.0171960275 -0.097819648 -0.242980790   -0.09021961   -0.104387723 -0.06672992
pneumonia     -0.0162015509  0.065622176  0.195724877    0.05144422    0.081183585  0.07215368
age           -0.0293394558  0.033748490  0.395147204    0.04277219    0.140318816  0.08291672
diabetes       0.0006772613  0.054617767  0.375988878    0.03333340    0.112142752  0.11497403
copd           0.0353752542  0.059749292  0.122541827    0.03841735    0.115063310  0.03780325
asthma         1.0000000000  0.021006434  0.015250193    0.01695136    0.019137422  0.04543863
inmsupr        0.0210064340  1.000000000  0.047005446    0.13914955    0.066672863  0.01450908
hypertension   0.0152501926  0.047005446  1.000000000    0.05164145    0.167840909  0.16370850
other_disease  0.0169513568  0.139149553  0.051641446    1.00000000    0.069895309  0.01929561
cardiovascular 0.0191374221  0.066672863  0.167840909    0.06989531    1.000000000  0.05829928
obesity        0.0454386349  0.014509077  0.163708497    0.01929561    0.058299276  1.00000000
renal_chronic  0.0010030674  0.118073391  0.189759614    0.05175766    0.111128124  0.01524141
tobacco        0.0055391378  0.011088417  0.013507531    0.01230431    0.032048490  0.07408911
covid_res      0.0251923515  0.016943436 -0.091360950    0.01104615   -0.003528303 -0.07646390
```



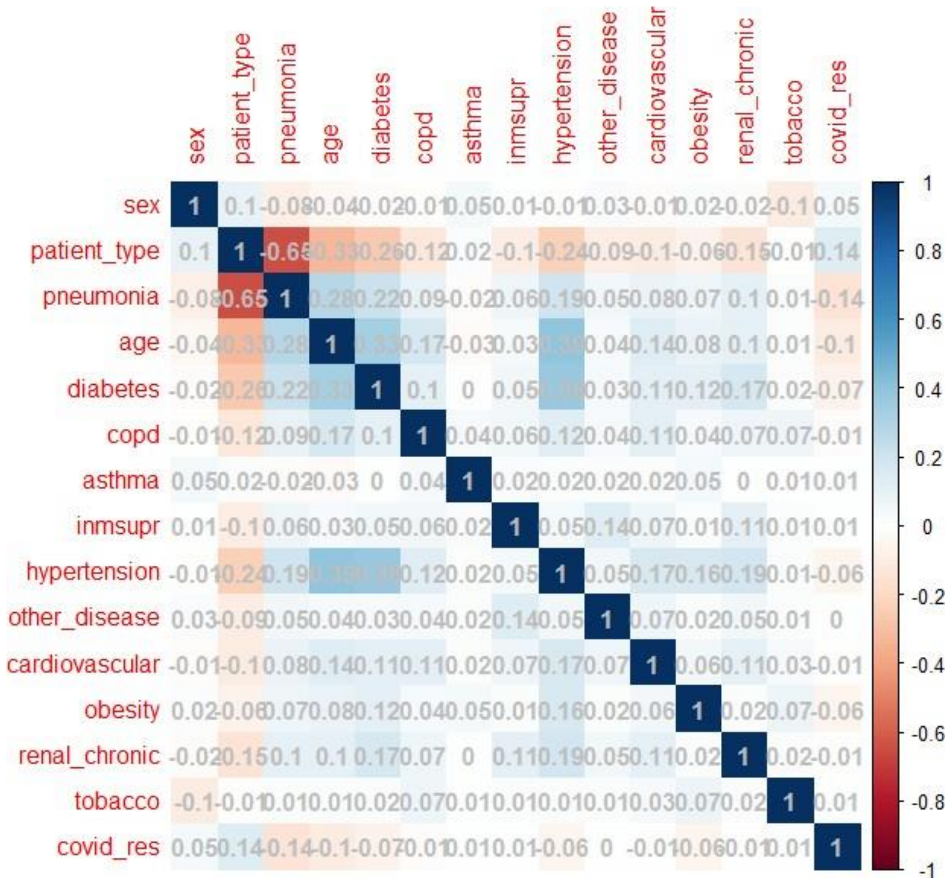
```
              renal_chronic      tobacco    covid_res
sex            -0.016042225 -0.104793461  0.072872825
patient_type   -0.153142795 -0.008510295  0.207782095
pneumonia       0.108825045  0.010947603 -0.203636654
age             0.104722349  0.013124827 -0.165386355
diabetes        0.170340800  0.015609444 -0.105250536
copd            0.068610961  0.070661460 -0.007022481
asthma          0.001003067  0.005539138  0.025192351
inmsupr         0.118073391  0.011088417  0.016943436
hypertension    0.189759614  0.013507531 -0.091360950
other_disease   0.051757663  0.012304306  0.011046154
cardiovascular  0.111128124  0.032048490 -0.003528303
obesity         0.015241411  0.074089107 -0.076463898
renal_chronic   1.000000000  0.015490871 -0.010522715
tobacco         0.015490871  1.000000000  0.027098593
covid_res      -0.010522715  0.027098593  1.000000000
> |
```

Above shown data is the correlation matrix for set of variables which determines if there is relationship between the variables. The positive number indicates positive relationship and negative number indicates negative relationship. For example, from above data (tobacco and renal chronic) are positively correlated and (tobacco and sex) are negatively correlated. Higher the positive number stronger is the correlation and Higher the negative number indicates weak correlation.

Correlation Plot



The above image shows the correlation plot. Blue square indicates positive correlation and red square indicates negative correlation. Dark blue color indicates strong correlation and Dark red color indicates weak correlations. For example: (age and hypertension) are positively correlated and (age and patient type) are negatively correlated.



Positive and Negative Correlations with Positive and Negative coefficient value.

K nearest neighbors (KNN)

k-nearest neighbors' classification for test set from training set. K-nearest neighbor classifier is one of the simplest to use, and hence, is widely used for classifying dynamic datasets.

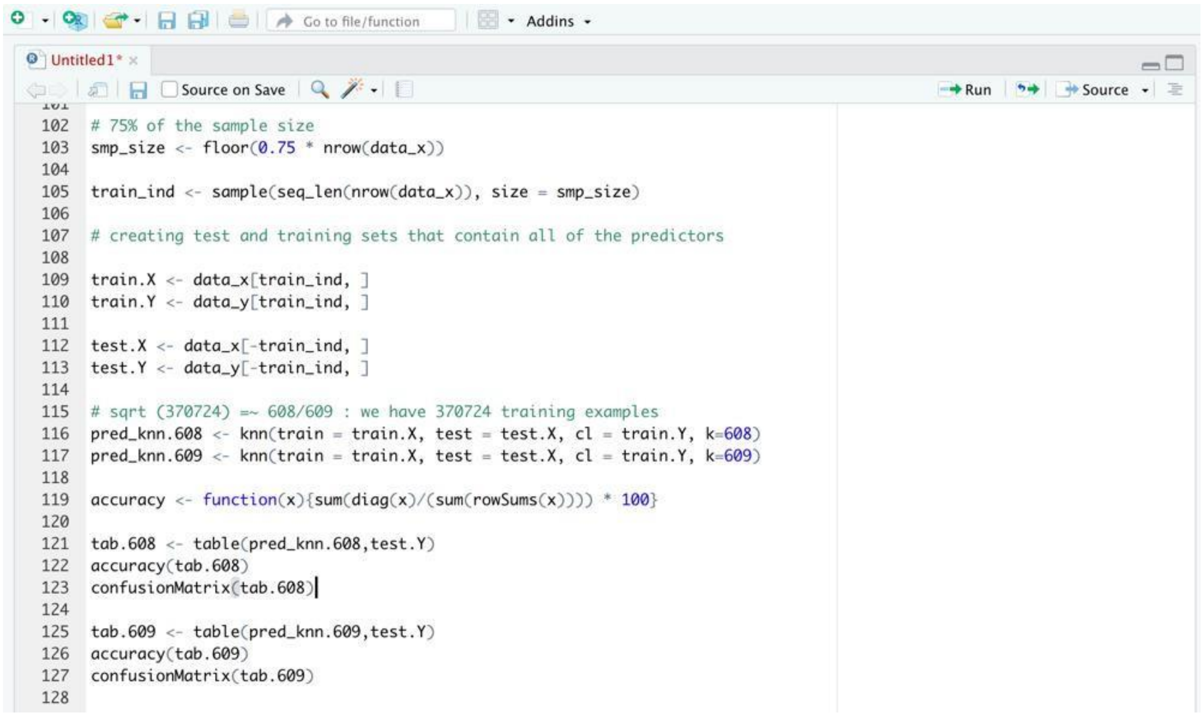
For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random.
To perform k-nearest neighbors for classification, we will use the knn() function from the class package.

Here, knn() takes four arguments:

- Train the predictors for the train set.
- Test, the predictors for the test set. knn() will output results (classifications) for these cases.
- cl, the true class labels for the train set.
- k, the number of neighbors to consider.

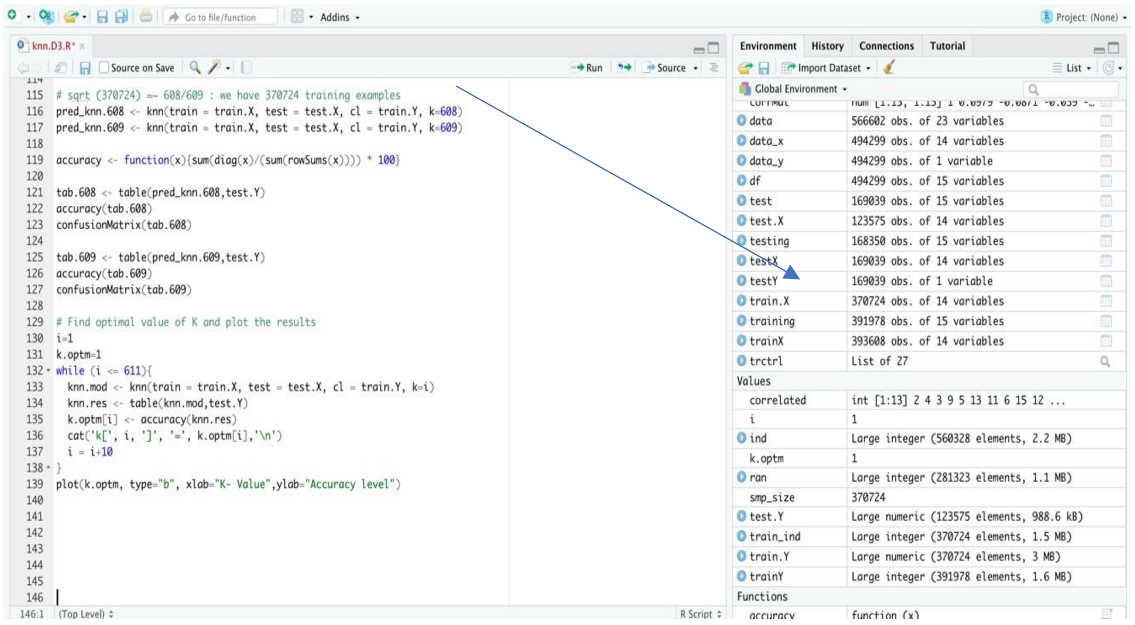
This is how we partition overall test data set into 75% training and 25%testing

```
smp_size <- floor(0.75 * nrow(data_x))

train_ind <- sample(seq_len(nrow(data_x)), size = smp_size)

# creating test and training sets that contain all of the predictors

train.X <- data_x[train_ind, ]
train.Y <- data_y[train_ind, ]

test.X <- data_x[-train_ind, ]
test.Y <- data_y[-train_ind, ]
```



```r
# 75% of the sample size
smp_size <- floor(0.75 * nrow(data_x))

train_ind <- sample(seq_len(nrow(data_x)), size = smp_size)

# creating test and training sets that contain all of the predictors

train.X <- data_x[train_ind, ]
train.Y <- data_y[train_ind, ]

test.X <- data_x[-train_ind, ]
test.Y <- data_y[-train_ind, ]

# sqrt (370724) =~ 608/609 : we have 370724 training examples
pred_knn.608 <- knn(train = train.X, test = test.X, cl = train.Y, k=608)
pred_knn.609 <- knn(train = train.X, test = test.X, cl = train.Y, k=609)

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

tab.608 <- table(pred_knn.608,test.Y)
accuracy(tab.608)
confusionMatrix(tab.608)

tab.609 <- table(pred_knn.609,test.Y)
accuracy(tab.609)
confusionMatrix(tab.609)

```

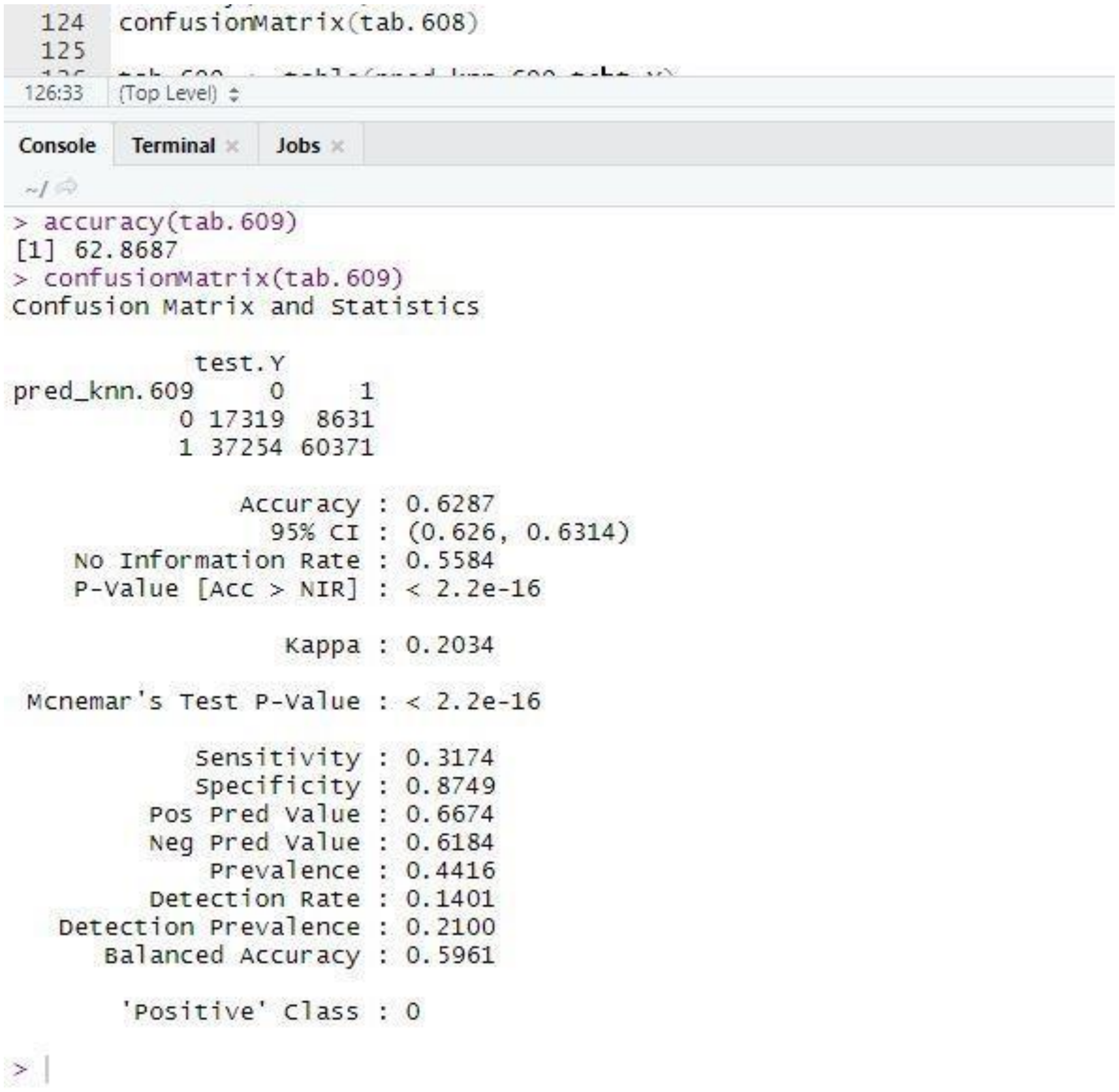
This is where we apply the knn model for our data set.



```r
# sqrt (370724) =~ 608/609 : we have 370724 training examples
pred_knn.608 <- knn(train = train.X, test = test.X, cl = train.Y, k=608)
pred_knn.609 <- knn(train = train.X, test = test.X, cl = train.Y, k=609)

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

tab.608 <- table(pred_knn.608,test.Y)
accuracy(tab.608)
confusionMatrix(tab.608)

tab.609 <- table(pred_knn.609,test.Y)
accuracy(tab.609)
confusionMatrix(tab.609)

# Find optimal value of K and plot the results
i=1
k.optm=1
while (i <= 611){
  knn.mod <- knn(train = train.X, test = test.X, cl = train.Y, k=i)
  knn.res <- table(knn.mod,test.Y)
  k.optm[i] <- accuracy(knn.res)
  cat('k[', i, ']', '=', k.optm[i],'\n')
  i = i+10
}
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```

corrMat	num [1:13, 1:13] 1 0.0979 -0.0871 -0.059 -…
data	566602 obs. of 23 variables
data_x	494299 obs. of 14 variables
data_y	494299 obs. of 1 variable
df	494299 obs. of 15 variables
test	169039 obs. of 15 variables
test.X	123575 obs. of 14 variables
testing	168350 obs. of 15 variables
testX	169039 obs. of 14 variables
testY	169039 obs. of 1 variable
train.X	370724 obs. of 14 variables
training	391978 obs. of 15 variables
trainX	393608 obs. of 14 variables
trctrl	List of 27
Values	
correlated	int [1:13] 2 4 3 9 5 13 11 6 15 12 ...
i	1
ind	Large integer (560328 elements, 2.2 MB)
k.optm	1
ran	Large integer (281323 elements, 1.1 MB)
smp_size	370724
test.Y	Large numeric (123575 elements, 988.6 kB)
train_ind	Large integer (370724 elements, 1.5 MB)
train.Y	Large numeric (370724 elements, 3 MB)
trainY	Large integer (391978 elements, 1.6 MB)
Functions	
accuracy	function (x)

After applying knn model to date set accuracy and confusion matrix to given data set Confusion matrix for model when K=609 and accuracy of model is 62.87%.



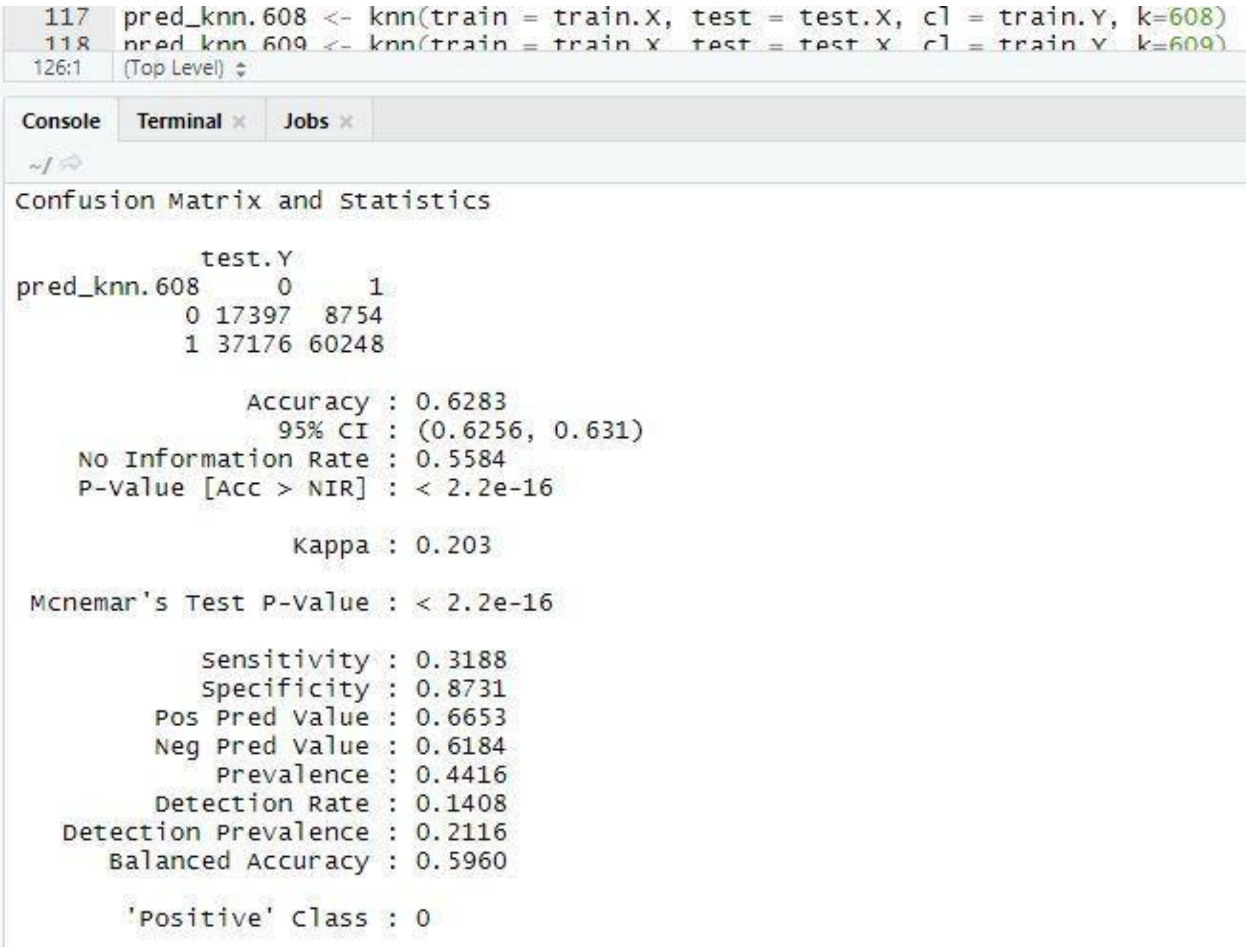
```
confusionMatrix(tab.608)

126:33   (Top Level) ÷
```

Console Terminal × Jobs ×

~/

```
> accuracy(tab.609)
[1] 62.8687
> confusionMatrix(tab.609)
Confusion Matrix and Statistics

             test.Y
pred_knn.609     0     1
           0 17319  8631
           1 37254 60371

               Accuracy : 0.6287
                 95% CI : (0.626, 0.6314)
    No Information Rate : 0.5584
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2034

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.3174
            Specificity : 0.8749
         Pos Pred Value : 0.6674
         Neg Pred Value : 0.6184
             Prevalence : 0.4416
         Detection Rate : 0.1401
   Detection Prevalence : 0.2100
      Balanced Accuracy : 0.5961

       'Positive' Class : 0

> |
```

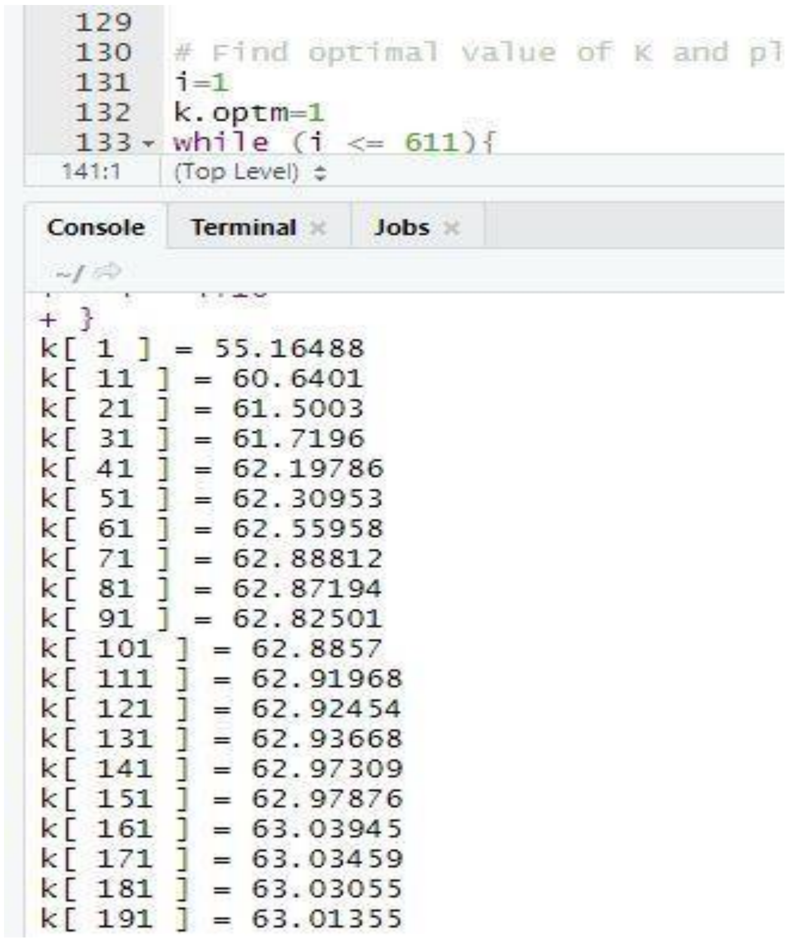
After applying knn model to date set accuracy and confusion matrix to given data set Confusion matrix for model when K=609 and accuracy of model is 62.87%.



```
pred_knn.608 <- knn(train = train.X, test = test.X, cl = train.Y, k=608)
pred_knn.609 <- knn(train = train.X, test = test.X, cl = train.Y, k=609)
126:1  (Top Level)
```

```
Confusion Matrix and Statistics

            test.Y
pred_knn.608     0     1
           0 17397  8754
           1 37176 60248

               Accuracy : 0.6283
                 95% CI : (0.6256, 0.631)
    No Information Rate : 0.5584
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.203

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.3188
            Specificity : 0.8731
         Pos Pred Value : 0.6653
         Neg Pred Value : 0.6184
             Prevalence : 0.4416
         Detection Rate : 0.1408
   Detection Prevalence : 0.2116
      Balanced Accuracy : 0.5960

       'Positive' Class : 0
```

Similarly, we observed the accuracy of model is 62.83% for k= 608



```
# Find optimal value of K and pl
i=1
k.optm=1
while (i <= 611){
```

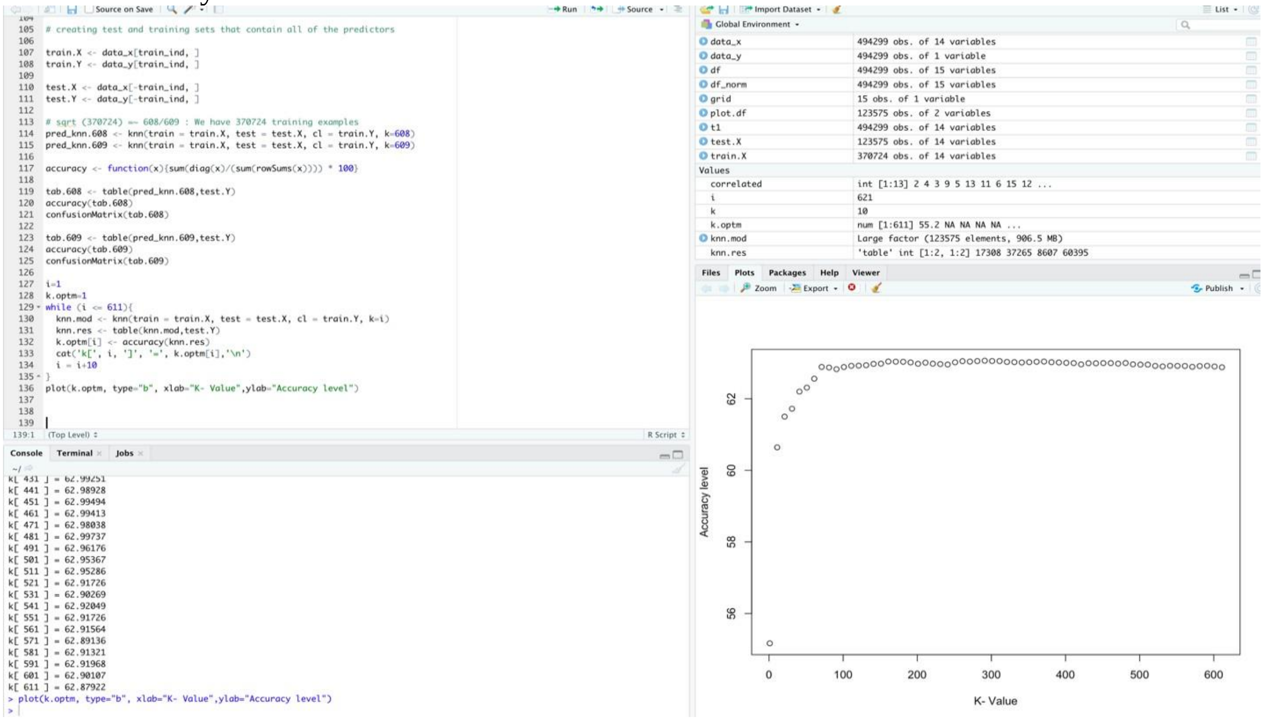
141:1 (Top Level)

Console Terminal Jobs

```
+ }
k[ 1 ] = 55.16488
k[ 11 ] = 60.6401
k[ 21 ] = 61.5003
k[ 31 ] = 61.7196
k[ 41 ] = 62.19786
k[ 51 ] = 62.30953
k[ 61 ] = 62.55958
k[ 71 ] = 62.88812
k[ 81 ] = 62.87194
k[ 91 ] = 62.82501
k[ 101 ] = 62.8857
k[ 111 ] = 62.91968
k[ 121 ] = 62.92454
k[ 131 ] = 62.93668
k[ 141 ] = 62.97309
k[ 151 ] = 62.97876
k[ 161 ] = 63.03945
k[ 171 ] = 63.03459
k[ 181 ] = 63.03055
k[ 191 ] = 63.01355
```

We also try find optimal value for k [1…611]

The maximum accuracy observed by our model is ~63% following graph shows the accuracy.
X axis: k values
Y axis: accuracy



```r
# creating test and training sets that contain all of the predictors

train.X <- data_x[train_ind, ]
train.Y <- data_y[train_ind, ]

test.X <- data_x[-train_ind, ]
test.Y <- data_y[-train_ind, ]

# sqrt (370724) ~= 608/609 : We have 370724 training examples
pred_knn.608 <- knn(train = train.X, test = test.X, cl = train.Y, k=608)
pred_knn.609 <- knn(train = train.X, test = test.X, cl = train.Y, k=609)

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

tab.608 <- table(pred_knn.608,test.Y)
accuracy(tab.608)
confusionMatrix(tab.608)

tab.609 <- table(pred_knn.609,test.Y)
accuracy(tab.609)
confusionMatrix(tab.609)

i=1
k.optm=1
while (i <= 611){
  knn.mod <- knn(train = train.X, test = test.X, cl = train.Y, k=i)
  knn.res <- table(knn.mod,test.Y)
  k.optm[i] <- accuracy(knn.res)
  cat('k[', i, ']', '=', k.optm[i],'\n')
  i = i+10
}
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```

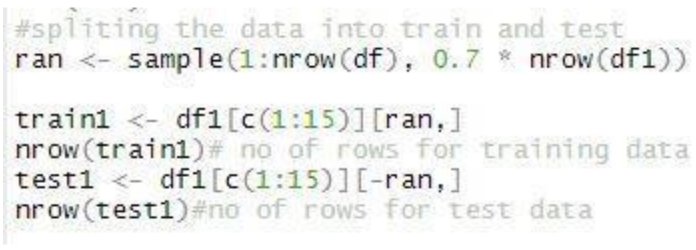
```
data_x      494299 obs. of 14 variables
data_y      494299 obs. of 1 variable
df          494299 obs. of 15 variables
df_norm     494299 obs. of 15 variables
grid        15 obs. of 1 variable
plot.df     123575 obs. of 2 variables
t1          494299 obs. of 14 variables
test.X      123575 obs. of 14 variables
train.X     370724 obs. of 14 variables
Values
  correlated  int [1:13] 2 4 3 9 5 13 11 6 15 12 ...
  i           621
  k           10
  k.optm      num [1:611] 55.2 NA NA NA NA ...
knn.mod     Large factor (123575 elements, 906.5 MB)
  knn.res     'table' int [1:2, 1:2] 17308 37265 8607 60395
```

```
~/
k[ 431 ] = 62.99251
k[ 441 ] = 62.98928
k[ 451 ] = 62.99494
k[ 461 ] = 62.99413
k[ 471 ] = 62.98038
k[ 481 ] = 62.99737
k[ 491 ] = 62.96176
k[ 501 ] = 62.95367
k[ 511 ] = 62.95286
k[ 521 ] = 62.91726
k[ 531 ] = 62.90269
k[ 541 ] = 62.92049
k[ 551 ] = 62.91726
k[ 561 ] = 62.91564
k[ 571 ] = 62.89136
k[ 581 ] = 62.91321
k[ 591 ] = 62.91968
k[ 601 ] = 62.90107
k[ 611 ] = 62.87922
> plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
>
```

The maximum accuracy observed by our model is ~63% following graph shows the accuracy.
X axis: k values
Y axis: accuracy

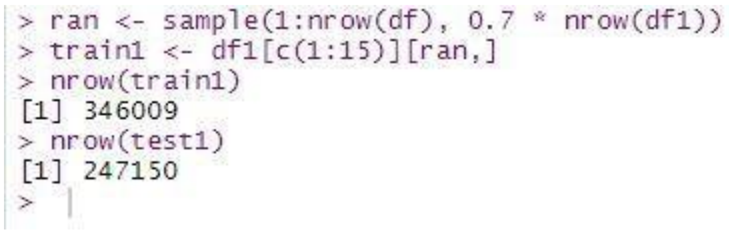
Naive Bayes

For Naïve Bayes, library e1071 is used. To spilt the data sample () function is used which randomly selects the data based on probability.



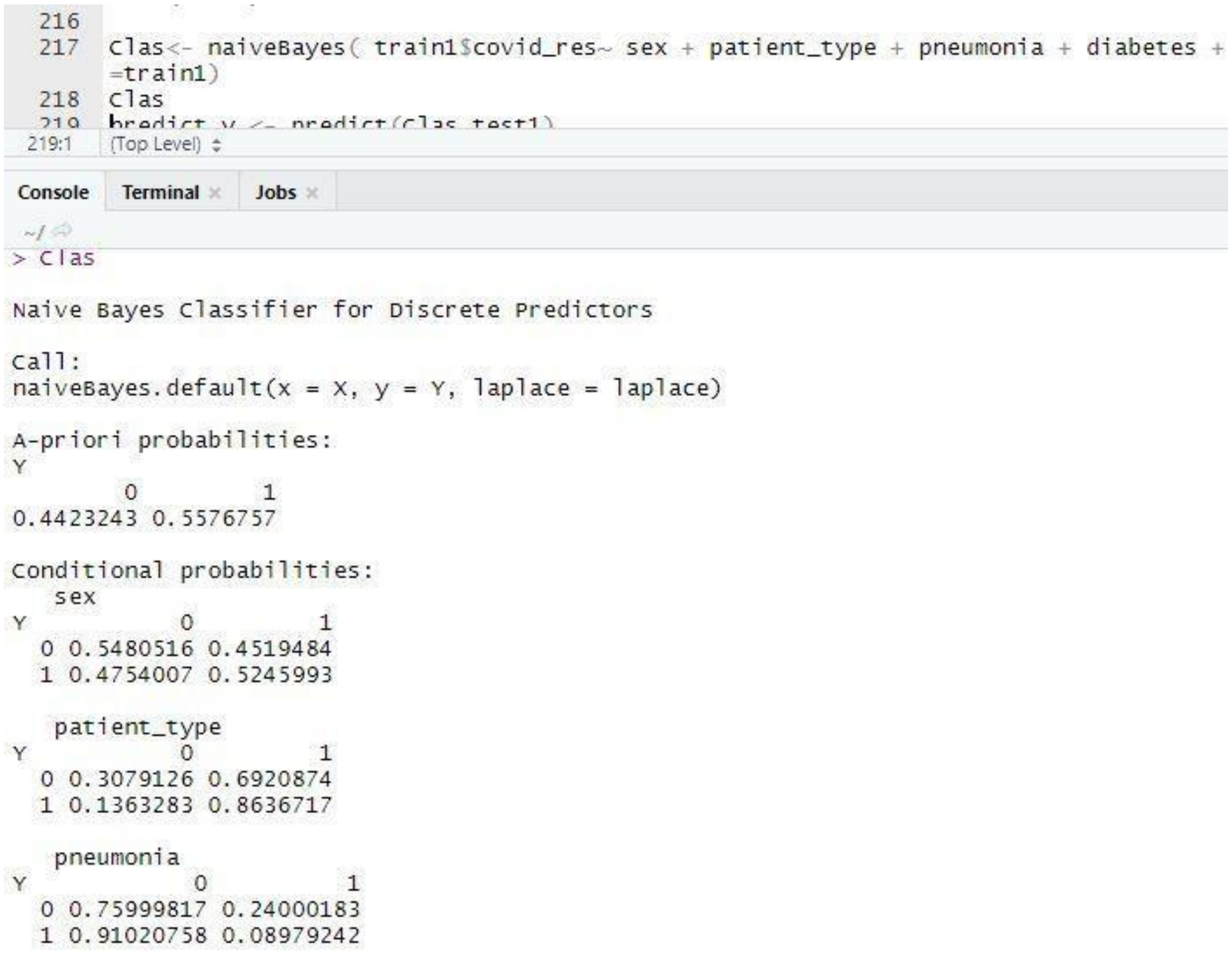
```r
#spliting the data into train and test
ran <- sample(1:nrow(df), 0.7 * nrow(df1))

train1 <- df1[c(1:15)][ran,]
nrow(train1)# no of rows for training data
test1 <- df1[c(1:15)][-ran,]
nrow(test1)#no of rows for test data
```

Rows count for train and test.



```
> ran <- sample(1:nrow(df), 0.7 * nrow(df1))
> train1 <- df1[c(1:15)][ran,]
> nrow(train1)
[1] 346009
> nrow(test1)
[1] 247150
> |
```

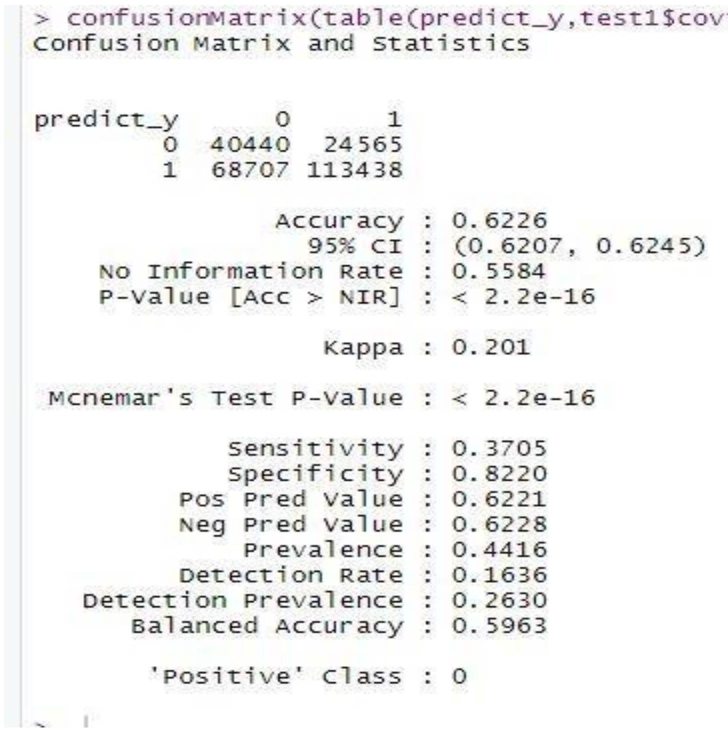
Then we have generated the naïve model using the function naiveBayes() and trained on response variable covid_res. Condition Probability is generated which tells about the likelihood of covid_res for each of the predictors.



```
Clas<- naiveBayes( train1$covid_res~ sex + patient_type + pneumonia + diabetes +
=train1)
Clas
predict_y <- predict(Clas,test1)
```

```
> Clas

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.4423243 0.5576757 

Conditional probabilities:
   sex
Y           0         1
  0 0.5480516 0.4519484
  1 0.4754007 0.5245993

   patient_type
Y           0         1
  0 0.3079126 0.6920874
  1 0.1363283 0.8636717

   pneumonia
Y            0          1
  0 0.75999817 0.24000183
  1 0.91020758 0.08979242
```

After the model is trained, we passed the test data for the prediction of covid_res. For this predict () function is used. And then Confusion Matrix is generated using the prediction. To create this confustionMatrix() function is used.



```
> confusionMatrix(table(predict_y,test1$covid_res))
Confusion Matrix and Statistics


predict_y      0      1
        0  40440  24565
        1  68707 113438

               Accuracy : 0.6226
                 95% CI : (0.6207, 0.6245)
    No Information Rate : 0.5584
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.201

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.3705
            Specificity : 0.8220
         Pos Pred Value : 0.6221
         Neg Pred Value : 0.6228
             Prevalence : 0.4416
         Detection Rate : 0.1636
   Detection Prevalence : 0.2630
      Balanced Accuracy : 0.5963

       'Positive' Class : 0

```

The confusion matrix gives the information about true positive, true negative , false positive and false negative. After applying the confusion matric, the 40440 elements are correctly classified who are covid positive whereas 113438 elements in the dataset are covid negative. Other elements are incorrectly classified. After applying Naïve Bayes model, the accuracy of the model is 62.26%.

Logistic regression Model:

a] Correlation matrix:

Correlation is a common metric in finance, and it is useful to know how to calculate it in R.

The **cor**() **function** will calculate the correlation between two vectors, or will create a correlation matrix when given a matrix.

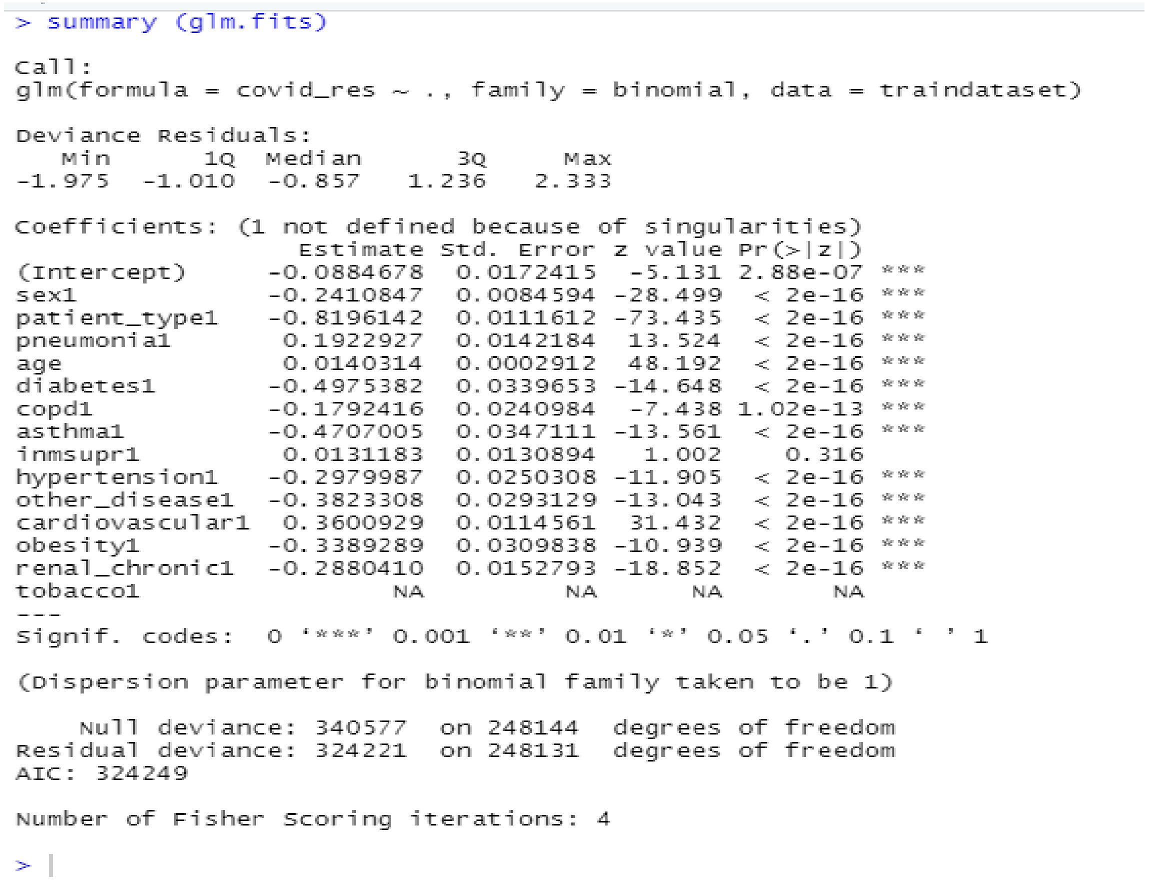
In our dataset we have only one numeric value (Age), so can not apply Correlation Matrix on our dataset.

b] Logistic regression:

We will try to implement Logistic regression model to predict the response variable Covid Result using columns from 1 to 14.

Segment <- sample(1:nrow(df), 0.5 * nrow(df))
traindataset <- df[c(1:15)][Segment,]
testdataset <- df[c(1:14)][-Segment,]

Using the above codes to divide the data into two equal halves for training and research. glm is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution. In order to say R to run a logistic regression, we need to define the argument **family = binomial.**
glm.fits=glm(covid_res ~ ., data = traindataset ,family =binomial)
summary (glm.fits)



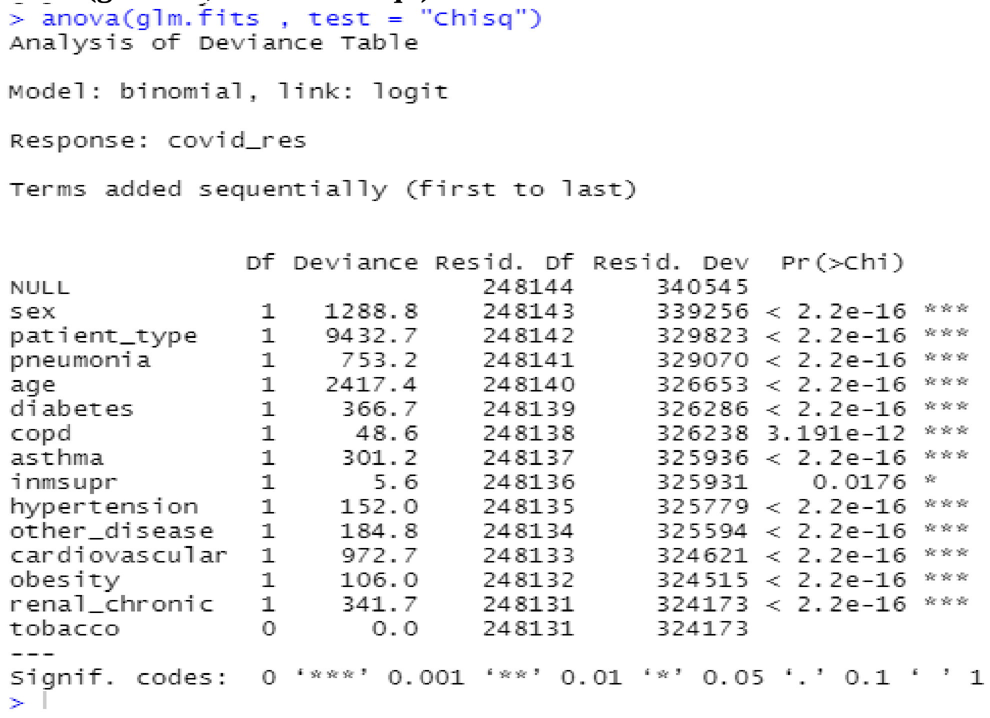
```
> summary (glm.fits)

Call:
glm(formula = covid_res ~ ., family = binomial, data = traindataset)

Deviance Residuals: 
   Min      1Q  Median      3Q     Max  
-1.975  -1.010  -0.857   1.236   2.333  

Coefficients: (1 not defined because of singularities)
                  Estimate Std. Error z value Pr(>|z|)    
(Intercept)     -0.0884678  0.0172415  -5.131 2.88e-07 ***
sex1            -0.2410847  0.0084594 -28.499  < 2e-16 ***
patient_type1   -0.8196142  0.0111612 -73.435  < 2e-16 ***
pneumonia1       0.1922927  0.0142184  13.524  < 2e-16 ***
age              0.0140314  0.0002912  48.192  < 2e-16 ***
diabetes1       -0.4975382  0.0339653 -14.648  < 2e-16 ***
copd1           -0.1792416  0.0240984  -7.438 1.02e-13 ***
asthma1         -0.4707005  0.0347111 -13.561  < 2e-16 ***
inmsupr1         0.0131183  0.0130894   1.002    0.316    
hypertension1   -0.2979987  0.0250308 -11.905  < 2e-16 ***
other_disease1  -0.3823308  0.0293129 -13.043  < 2e-16 ***
cardiovascular1  0.3600929  0.0114561  31.432  < 2e-16 ***
obesity1        -0.3389289  0.0309838 -10.939  < 2e-16 ***
renal_chronic1  -0.2880410  0.0152793 -18.852  < 2e-16 ***
tobacco1                NA         NA      NA       NA    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 340577  on 248144  degrees of freedom
Residual deviance: 324221  on 248131  degrees of freedom
AIC: 324249

Number of Fisher Scoring iterations: 4

> |
```

Tobaco doesn’t have any statistical significance value in this model.Here we can observe that we have a total of 14 coefficient numbers and P-value is very large 0.0130894 for Inmsupr1 means that Inmsupr1 statistical significance is not relevant to predict Covid19, similarly gluc2 statistical significance is also not essential since p-value is high, so when either forward selection or backward exclusion can be used to pick the right attributes to build better prediction.

ANOVA:

The one-way analysis of variance (ANOVA), also known as one-factor ANOVA, is an extension of independent two-samples t-test for comparing means in a situation where there are more than two groups. In one-way ANOVA, the data is organized into several groups base on one single grouping variable (also called factor variable). This tutorial describes the basic principle of the one-way ANOVA test and provides practical anova test examples in R software.

anova(glm.fits , test = "Chisq")

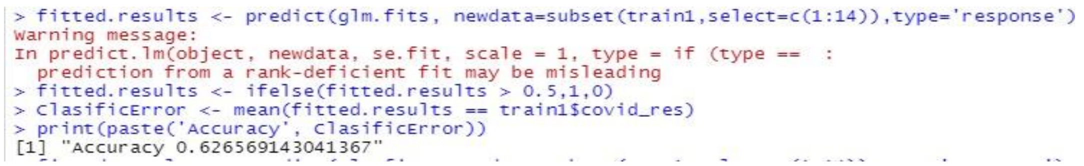


```
> anova(glm.fits , test = "Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: covid_res

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)    
NULL                          248144     340545              
sex             1   1288.8    248143     339256 < 2.2e-16 ***
patient_type    1   9432.7    248142     329823 < 2.2e-16 ***
pneumonia       1    753.2    248141     329070 < 2.2e-16 ***
age             1   2417.4    248140     326653 < 2.2e-16 ***
diabetes        1    366.7    248139     326286 < 2.2e-16 ***
copd            1     48.6    248138     326238 3.191e-12 ***
asthma          1    301.2    248137     325936 < 2.2e-16 ***
inmsupr         1      5.6    248136     325931    0.0176 *  
hypertension    1    152.0    248135     325779 < 2.2e-16 ***
other_disease   1    184.8    248134     325594 < 2.2e-16 ***
cardiovascular  1    972.7    248133     324621 < 2.2e-16 ***
obesity         1    106.0    248132     324515 < 2.2e-16 ***
renal_chronic   1    341.7    248131     324173 < 2.2e-16 ***
tobacco         0      0.0    248131     324173              
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> 
```

the predict() function to suit the model into the test data and predict the response variable, since we first generated an object in the above code block and allocated a response variable value to the object, which we can then use if another comment affects the likelihood of class.

1. If the probability (P)> 0.5, class 1 indicates that patient has Covid19.
2. If the probability (P)<0.5, class '0' means patient does not Covid19.

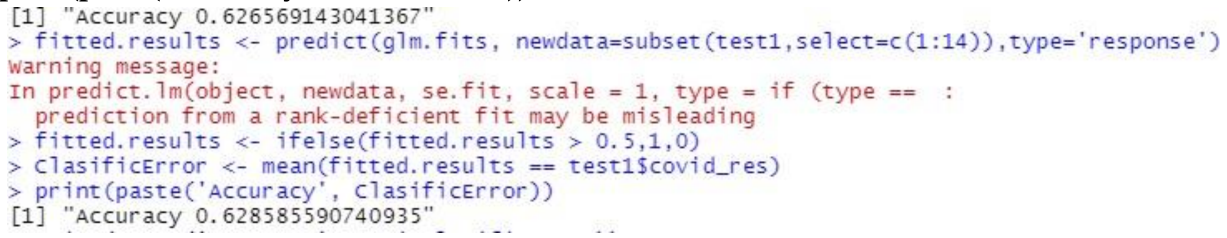
fitted.results <- predict(glm.fits, newdata=subset(train1,select=c(1:14)),type='response') fitted.results <- ifelse(fitted.results > 0.5,1,0)
ClasificError <- mean(fitted.results == train1$covid_res)
print(paste('Accuracy', ClasificError))



```
> fitted.results <- predict(glm.fits, newdata=subset(train1,select=c(1:14)),type='response')
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> fitted.results <- ifelse(fitted.results > 0.5,1,0)
> ClasificError <- mean(fitted.results == train1$covid_res)
> print(paste('Accuracy', ClasificError))
[1] "Accuracy 0.626569143041367"
```

fitted.results <- predict(glm.fits, newdata=subset(test1,select=c(1:14)),type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)
ClasificError <- mean(fitted.results == test1$covid_res)

print(paste('Accuracy', ClasificError))



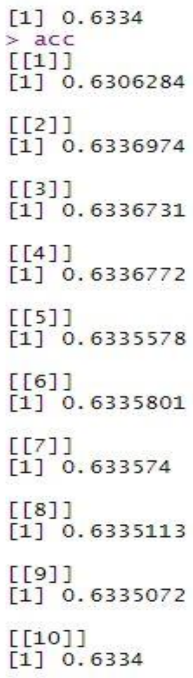
```
[1] "Accuracy 0.626569143041367"
> fitted.results <- predict(glm.fits, newdata=subset(test1,select=c(1:14)),type='response')
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> fitted.results <- ifelse(fitted.results > 0.5,1,0)
> ClasificError <- mean(fitted.results == test1$covid_res)
> print(paste('Accuracy', ClasificError))
[1] "Accuracy 0.628585590740935"
```

This is equal to the model's accuracy since we will run the model on both halves first on test data and obtained 62.85 percent accuracy and then on training dataset and obtained 62.85 percent accuracy, indicating that there is no variation and we will be able to generate more accuracy. Also, we are implementing K-fold cross validation to minimize i.

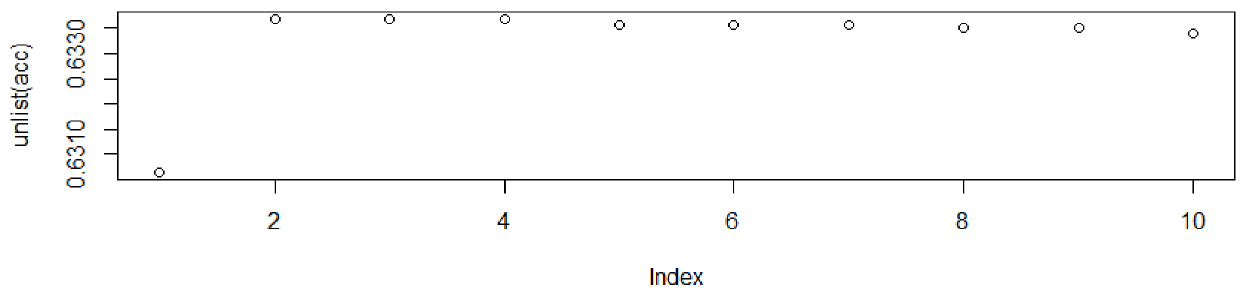
Resampling

Resampling is the method that consists of drawing repeated samples from the original data samples. The process of Resampling is a nonparametric method of statistical inference. Resampling involves selecting randomized cases with replacement from the original data sample in such a manner that each number of the sample drawn has several cases like the original data sample. Due to replacement, the illustrated number of samples used by the method of Resampling consists of repetitive cases.

logistic regression with -whole dataset approach

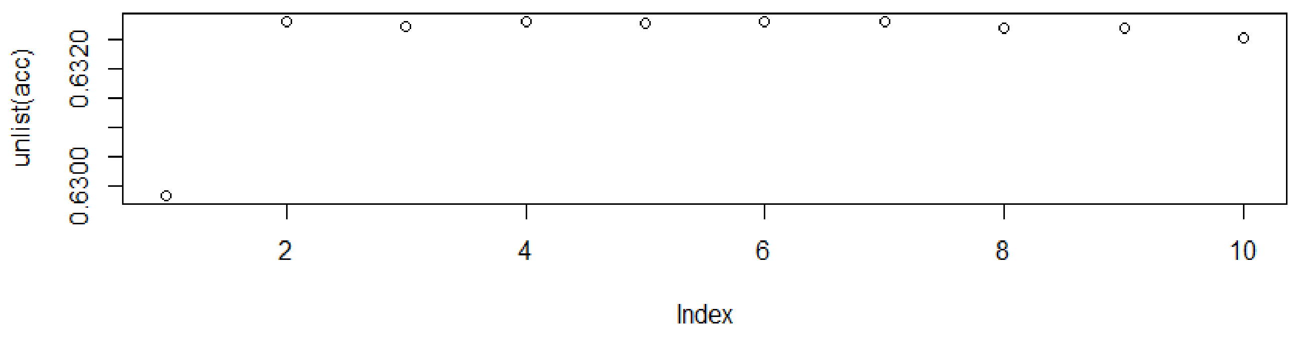


```
[1] 0.6334
> acc
[[1]]
[1] 0.6306284

[[2]]
[1] 0.6336974

[[3]]
[1] 0.6336731

[[4]]
[1] 0.6336772

[[5]]
[1] 0.6335578

[[6]]
[1] 0.6335801

[[7]]
[1] 0.633574

[[8]]
[1] 0.6335113

[[9]]
[1] 0.6335072

[[10]]
[1] 0.6334
```



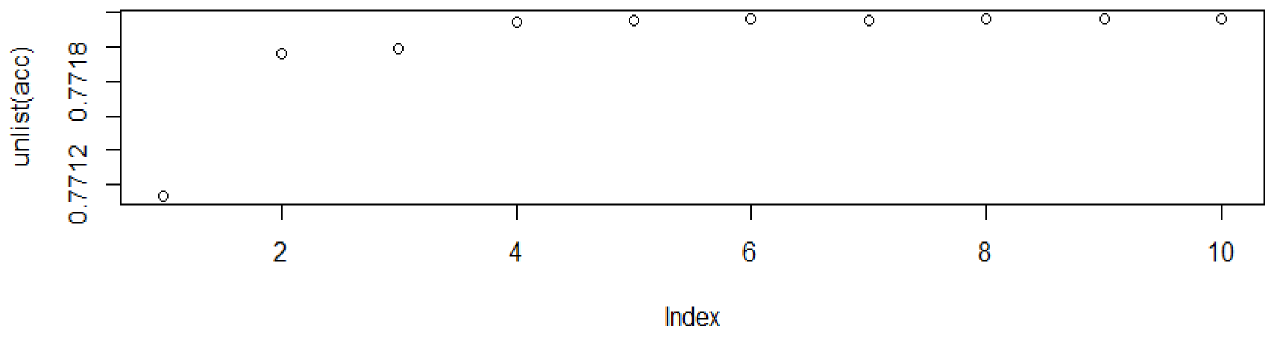
In this cross-validation approach, we are using whole date set for training and testing dataset. Above graph is output for this method, where we are using polynomial degree to check the accuracy. In above result we can clearly observe that initially degree of polynomial is increased and then remain constant.

logistic regression with - Validation set approach:
Following step are Used:-
1.Randomly divide the available set of observations into two parts, a training set and a validation set or hold-out set.
2. Fit the model on the training set.
3. Use the resulting fitted model to predict the responses for the observations in the validation set.
4. The resulting validation set error rate is typically assessed using the MSE in the case of a quantitative response. This provides an estimate of the test error rate.



For this technique we divided the data in to two halves test data and training data. These are the result we received, polynomial degree is increased substantially in the beginning and then remain constant for rest of the time.

logistic regression with - K- fold
This method randomly divides a set of observations into *k* groups, for *folds*, of approximately equal size. Each fold contains a non-overlapping (with the subsequent folds) validation set and training set. The approach could be thought of as a hybrid of both the LOOCV and the validation approach. In fact LOOCV is a special case of *k*-folds where k = n. The advantage of this is computational speed.



For this technique we divided the data randomly data. These are the result we got, polynomial degree is increased considerably in the beginning and then remain constant for rest of the time.

KNN

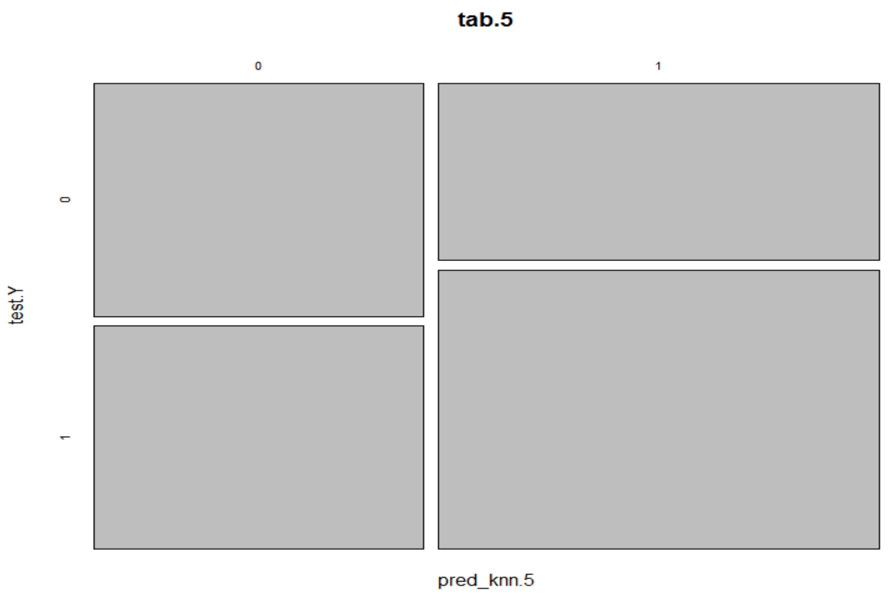
leave-one-out cross validation

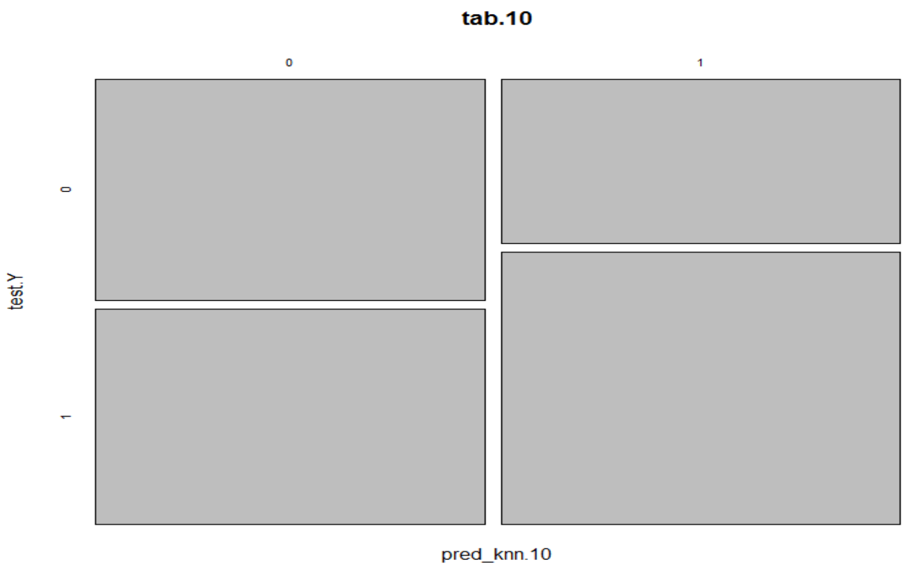
In this for each row of the training set train, the k nearest (in Euclidean distance) other training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the k th nearest vector, all candidates are included in the vote.

Training and Testing dataset :

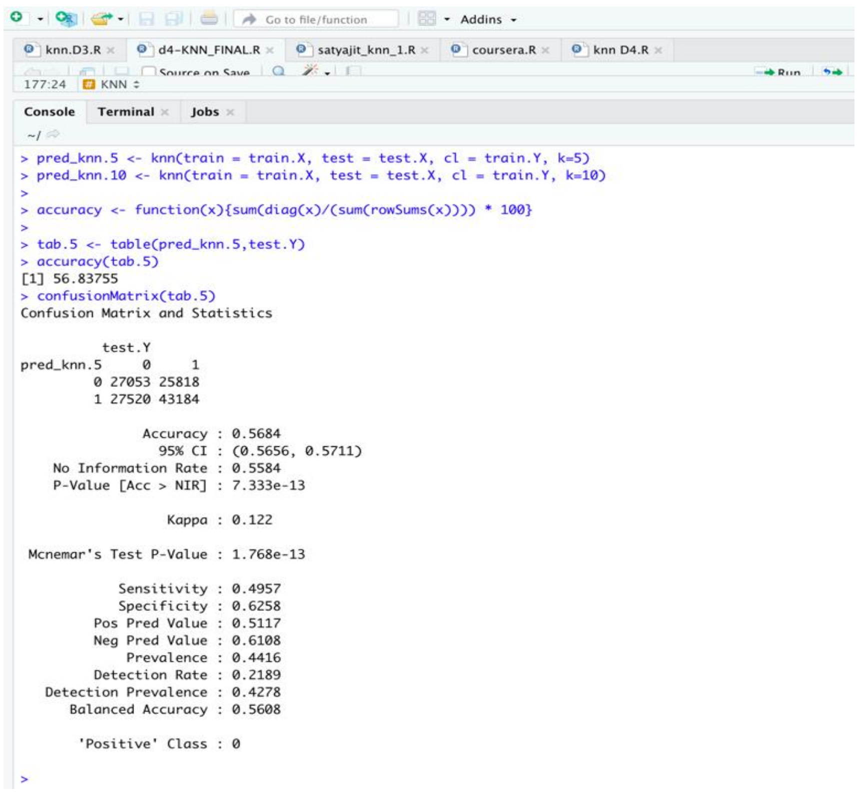
```
data_y <- df %>% select(covid_res) # storing response variable covid_res in data_y
data_x <- df %>% select(-covid_res) # storing all the predictors variable in data_x

# 75% of the sample size
smp_size <- floor(0.75 * nrow(data_x))
train_ind <- sample(seq_len(nrow(data_x)), size = smp_size)
# creating test and training sets that contain all of the predictors
train.X <- data_x[train_ind, ]
train.Y <- data_y[train_ind, ]

test.X <- data_x[-train_ind, ]
test.Y <- data_y[-train_ind, ]

pred_knn.5 <- knn(train = train.X, test = test.X, cl = train.Y, k=5)
pred_knn.10 <- knn(train = train.X, test = test.X, cl = train.Y, k=10)

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

tab.5 <- table(pred_knn.5,test.Y)
accuracy(tab.5)
confusionMatrix(tab.5)
```

```
tab.10 <- table(pred_knn.10,test.Y)
accuracy(tab.10)
confusionMatrix(tab.10)
```





output:



```
> pred_knn.5 <- knn(train = train.X, test = test.X, cl = train.Y, k=5)
> pred_knn.10 <- knn(train = train.X, test = test.X, cl = train.Y, k=10)
>
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
>
> tab.5 <- table(pred_knn.5,test.Y)
> accuracy(tab.5)
[1] 56.83755
> confusionMatrix(tab.5)
Confusion Matrix and Statistics

          test.Y
pred_knn.5     0     1
         0 27053 25818
         1 27520 43184

               Accuracy : 0.5684
                 95% CI : (0.5656, 0.5711)
    No Information Rate : 0.5584
    P-Value [Acc > NIR] : 7.333e-13

                  Kappa : 0.122

 Mcnemar's Test P-Value : 1.768e-13

            Sensitivity : 0.4957
            Specificity : 0.6258
         Pos Pred Value : 0.5117
         Neg Pred Value : 0.6108
             Prevalence : 0.4416
         Detection Rate : 0.2189
   Detection Prevalence : 0.4278
      Balanced Accuracy : 0.5608

       'Positive' Class : 0

>
```

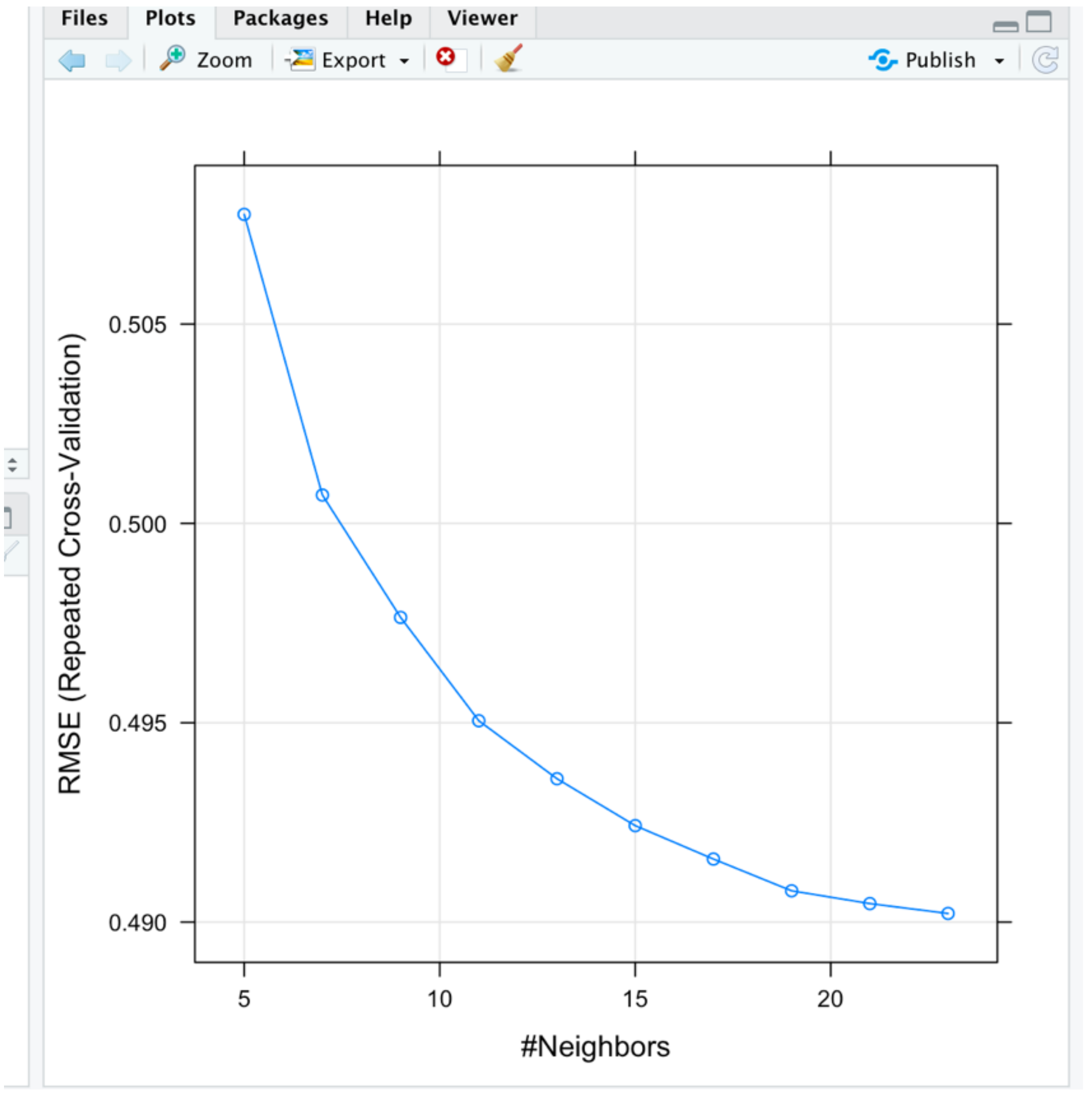
k-fold cross validation

It’s easy to follow and implement. Below are the steps for it:

- Randomly split your entire dataset into k”folds”
- For each k-fold in your dataset, build model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for *kth* fold
- Record the error on each of the predictions
- Repeat this until each of the k-folds has served as the test set
- The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model

```
#plotting cross-validated prediction accuracy
qplot(knn_results.k, knn_results.Accuracy,geom = "line",
    xlab = "k", ylab = "Accuracy")
input.data <-df[sample(nrow(df), 20000), ]
smp_size <- floor(0.25 * nrow(input.data ))
train_ind <- sample(seq_len(nrow(input.data)), size = smp_size)
input.data.train <- df[train_ind, ]
input.data.test <- df[-train_ind, ]
Fold 10:
trCtrl.knn <- trainControl(method = "repeatedcv",
                    number = 10,  #10-fold CV
                    repeats = 3,
                        classProbs = TRUE,
                        savePredictions = TRUE)
model.knn <- train(covid_res ~ .,
                   data=input.data.train,
                   method="knn",
                   trControl = trCtrl.knn,
                   preProcess = c("center","scale"),
                   tuneLength=10)
model.knn_predict <- predict(model.knn, newdata = input.data.test)
#To print knn model
print(model.knn)
#To plot model knn
plot(model.knn)
```

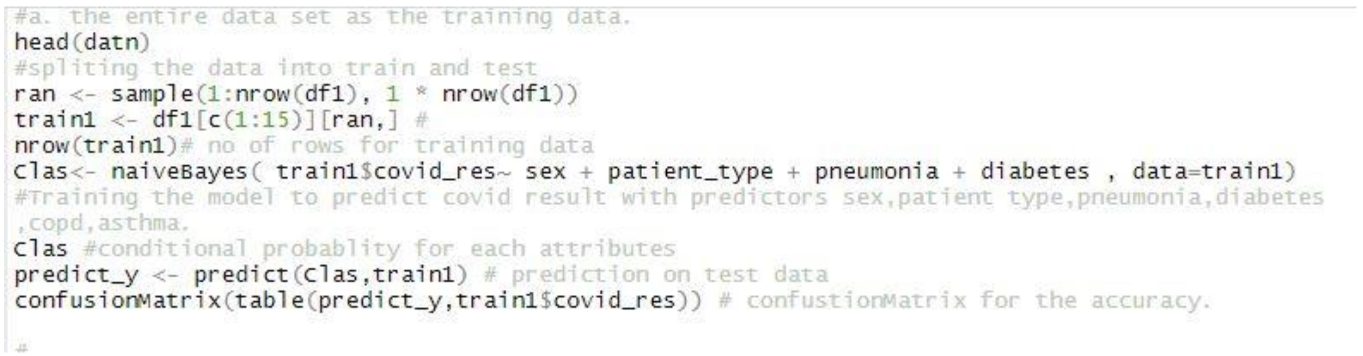
output:



The RMSE decreases as the k polynomial increases.

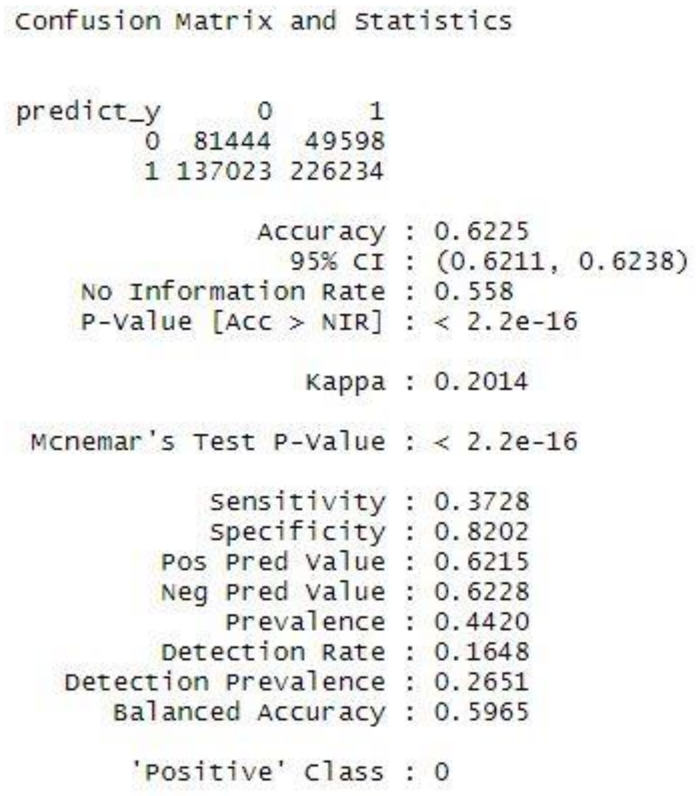
Naïve Bayes

a) The Entire data set as the training data



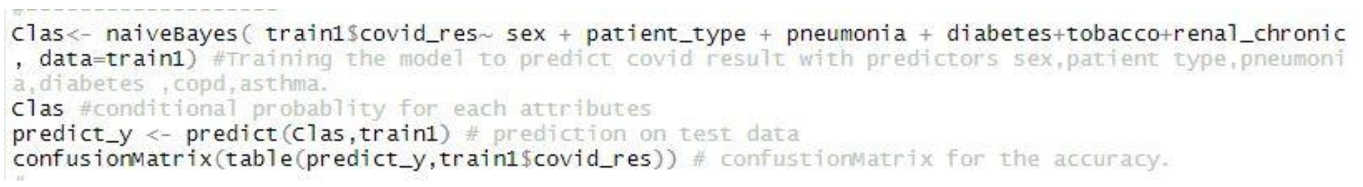
```
#a. the entire data set as the training data.
head(datn)
#spliting the data into train and test
ran <- sample(1:nrow(df1), 1 * nrow(df1))
train1 <- df1[c(1:15)][ran,] #
nrow(train1)# no of rows for training data
Clas<- naiveBayes( train1$covid_res~ sex + patient_type + pneumonia + diabetes , data=train1)
#Training the model to predict covid result with predictors sex,patient type,pneumonia,diabetes
,copd,asthma.
Clas #conditional probablity for each attributes
predict_y <- predict(Clas,train1) # prediction on test data
confusionMatrix(table(predict_y,train1$covid_res)) # confustionMatrix for the accuracy.

#
```

Taken entire dataset as training data and passed 4 parameters for the prediction.



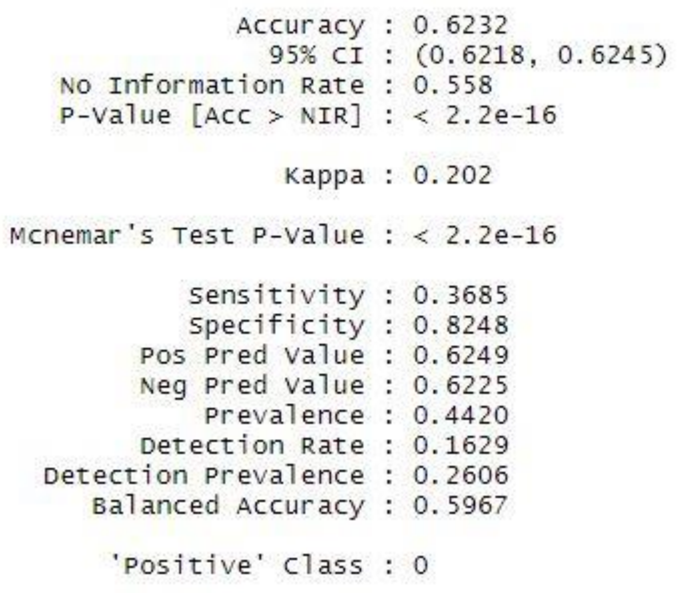
```
Confusion Matrix and Statistics


predict_y      0      1
        0  81444  49598
        1 137023 226234

               Accuracy : 0.6225
                 95% CI : (0.6211, 0.6238)
    No Information Rate : 0.558
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2014

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.3728
            Specificity : 0.8202
         Pos Pred Value : 0.6215
         Neg Pred Value : 0.6228
             Prevalence : 0.4420
         Detection Rate : 0.1648
   Detection Prevalence : 0.2651
      Balanced Accuracy : 0.5965

       'Positive' Class : 0
```

Taking entire data as training data and passed 4 parameters for prediction, we get the accuracy of 62.25%. Then we increased the number of parameters on entire training data.



```
#--------------------
Clas<- naiveBayes( train1$covid_res~ sex + patient_type + pneumonia + diabetes+tobacco+renal_chronic
, data=train1) #Training the model to predict covid result with predictors sex,patient type,pneumoni
a,diabetes ,copd,asthma.
Clas #conditional probablity for each attributes
predict_y <- predict(Clas,train1) # prediction on test data
confusionMatrix(table(predict_y,train1$covid_res)) # confustionMatrix for the accuracy.
#
```

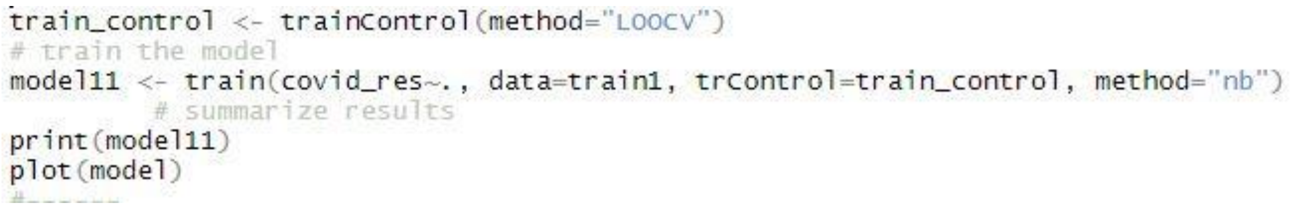
Passed 6 parameters for the prediction:



```
               Accuracy : 0.6232
                 95% CI : (0.6218, 0.6245)
    No Information Rate : 0.558
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.202

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.3685
            Specificity : 0.8248
         Pos Pred Value : 0.6249
         Neg Pred Value : 0.6225
             Prevalence : 0.4420
         Detection Rate : 0.1629
   Detection Prevalence : 0.2606
      Balanced Accuracy : 0.5967

       'Positive' Class : 0
```

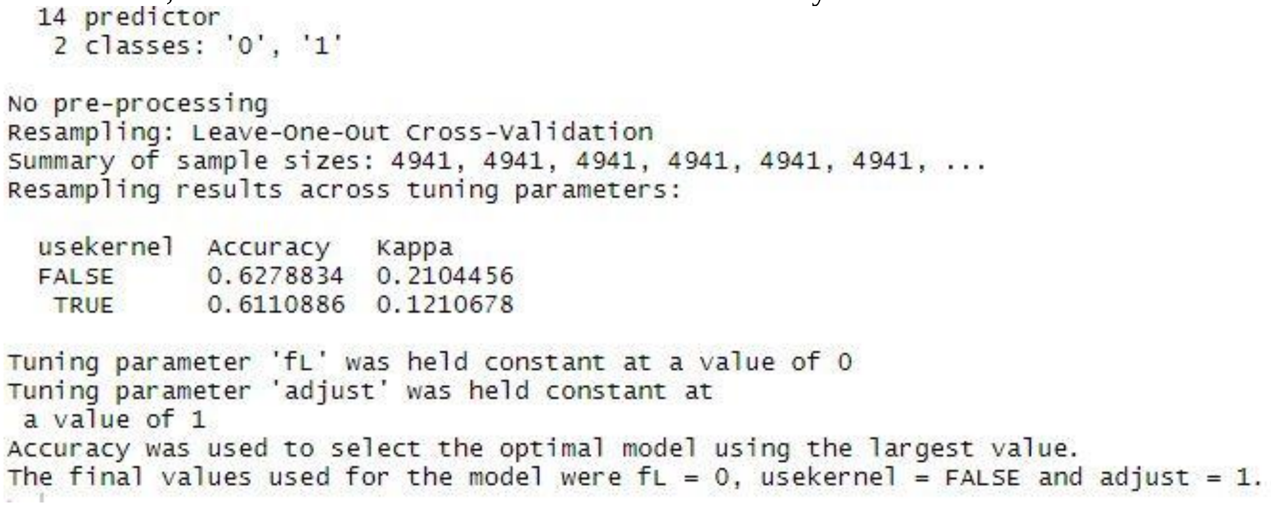
After increasing the parameters on entire data, the accuracy slightly increased from 62.25 to 62.32

c) leave-one-out cross validation

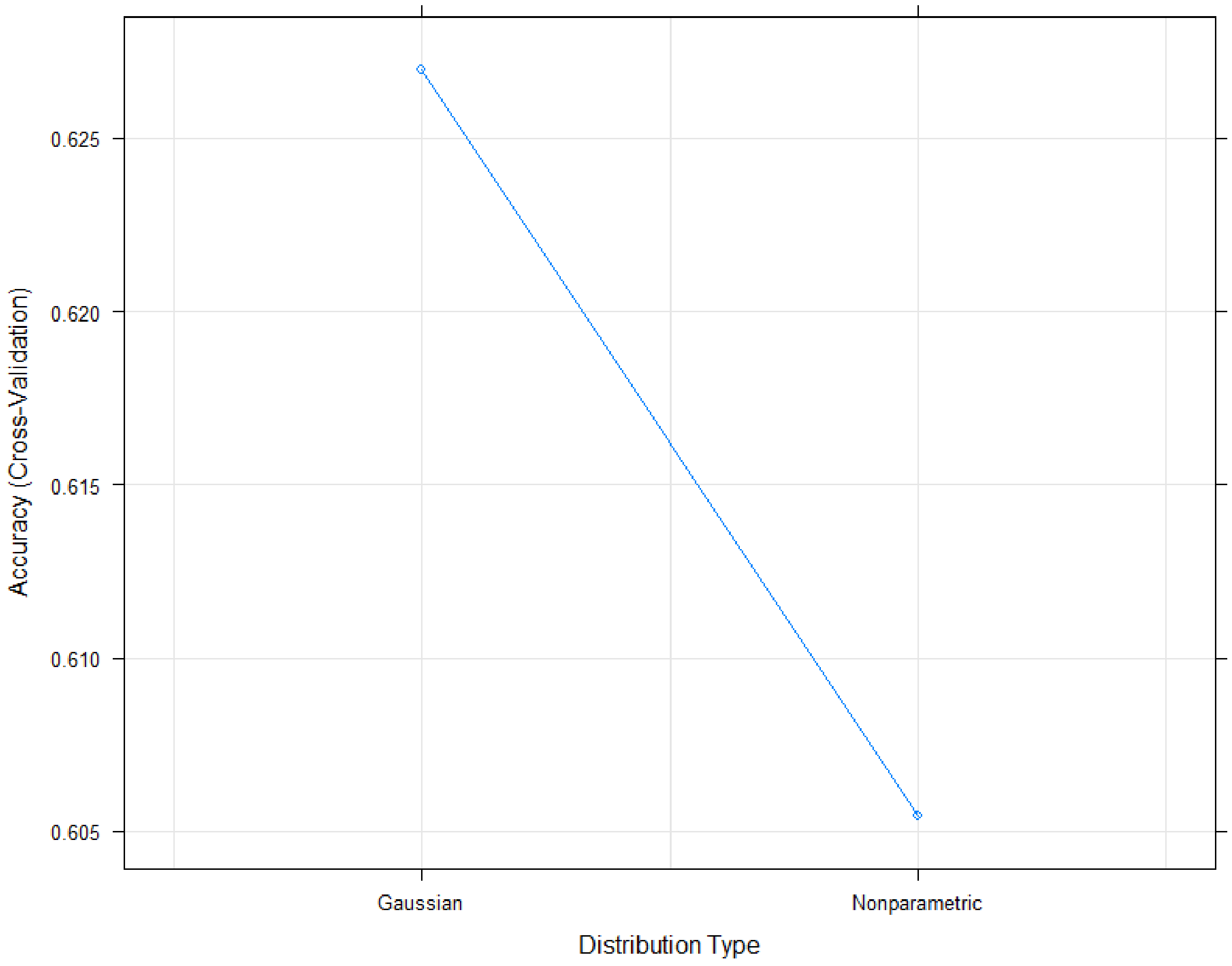


```
train_control <- trainControl(method="LOOCV")
# train the model
model11 <- train(covid_res~., data=train1, trControl=train_control, method="nb")
         # summarize results
print(model11)
plot(model)
#------
```

For LOOCV, we used method LOOCV with to make Naïve Bayes model.

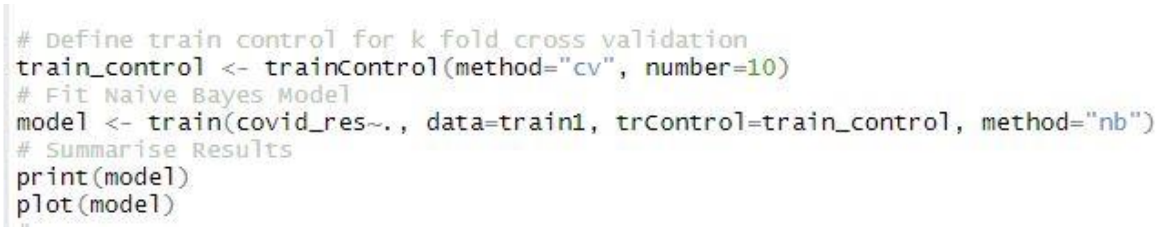


```
  14 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 4941, 4941, 4941, 4941, 4941, 4941, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.6278834  0.2104456
   TRUE      0.6110886  0.1210678

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at
 a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 1.
```



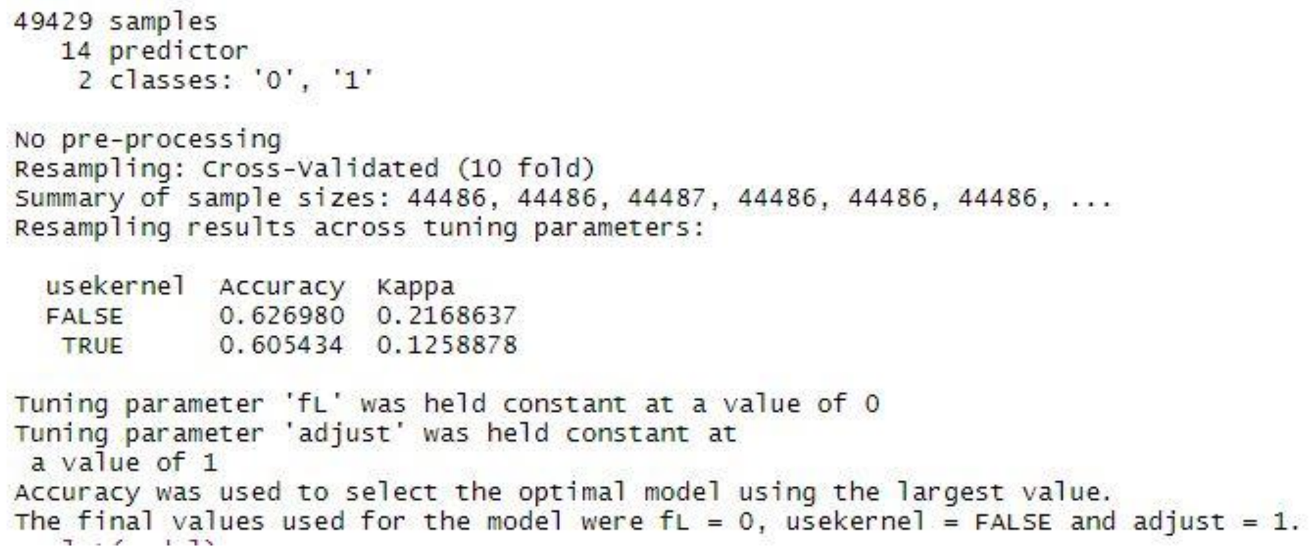
d) 10-fold cross validation

Naïve Bayes

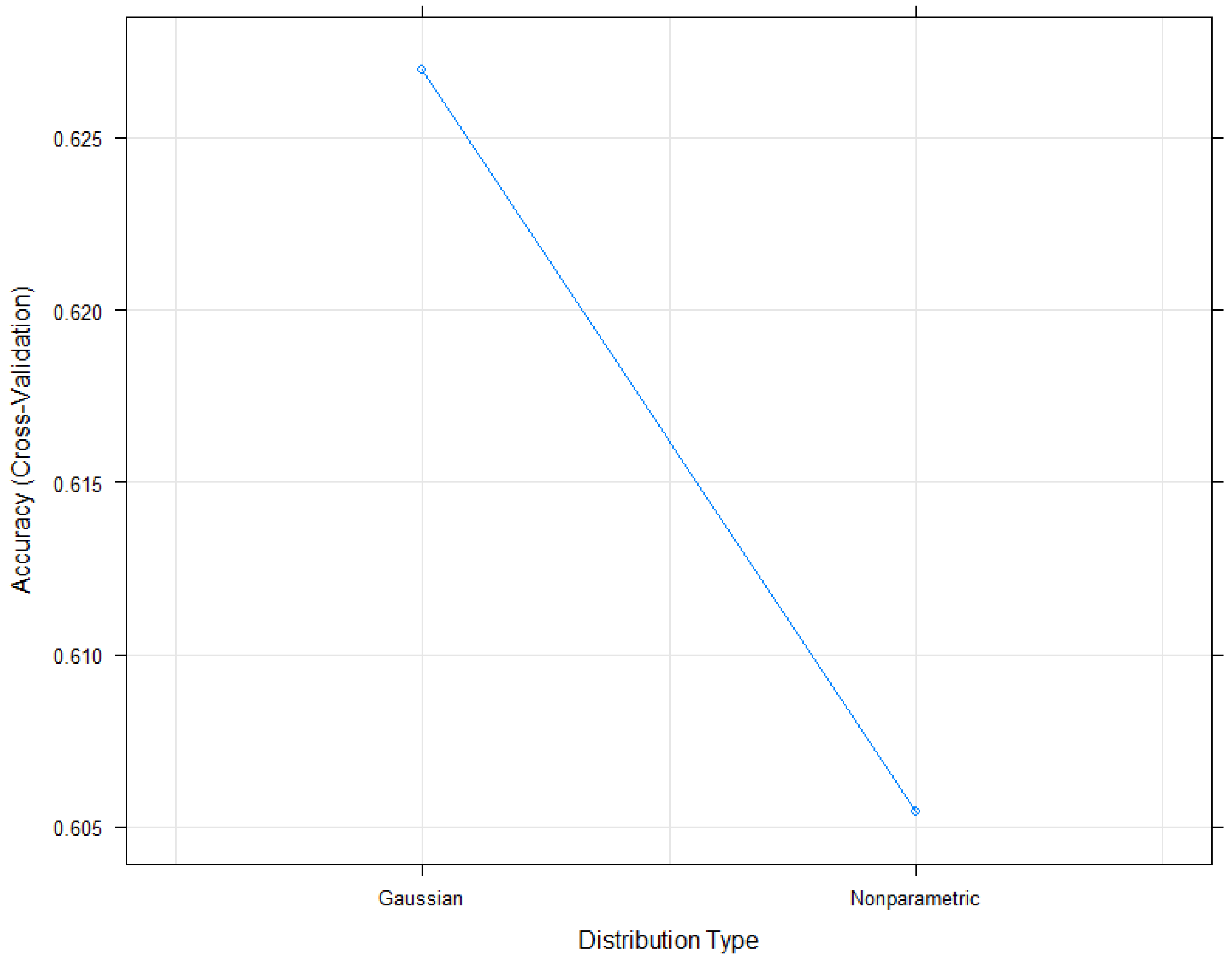


```
# Define train control for k fold cross validation
train_control <- trainControl(method="cv", number=10)
# Fit Naive Bayes Model
model <- train(covid_res~., data=train1, trControl=train_control, method="nb")
# Summarise Results
print(model)
plot(model)
```

Method used cv and number=10 for 10-fold cross validation.



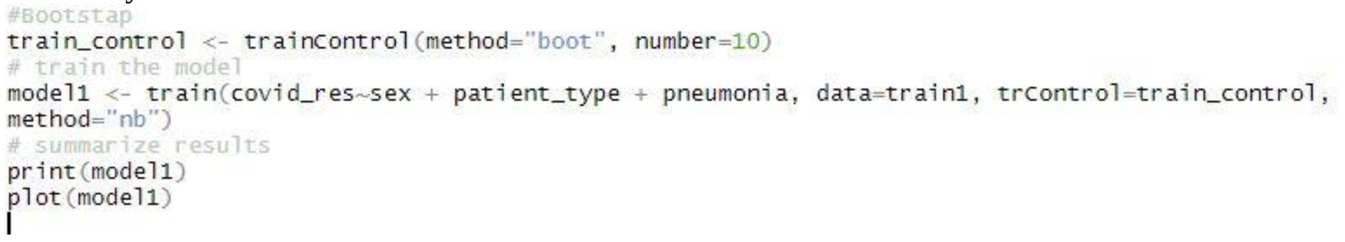
```
49429 samples
   14 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 44486, 44486, 44487, 44486, 44486, 44486, ...
Resampling results across tuning parameters:

  usekernel  Accuracy  Kappa
  FALSE      0.626980  0.2168637
   TRUE      0.605434  0.1258878

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at
 a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 1.
```



The Accuracy increases from 60.2 to 62.75

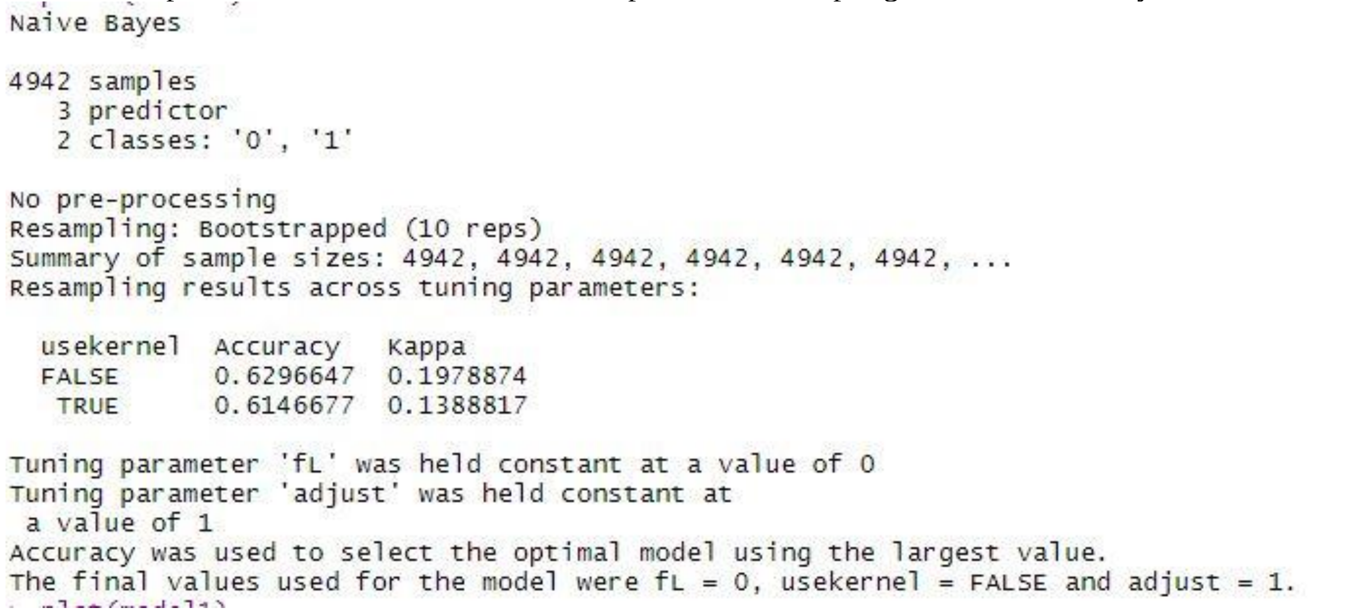
2.) Bootstrap

Navie Bayes



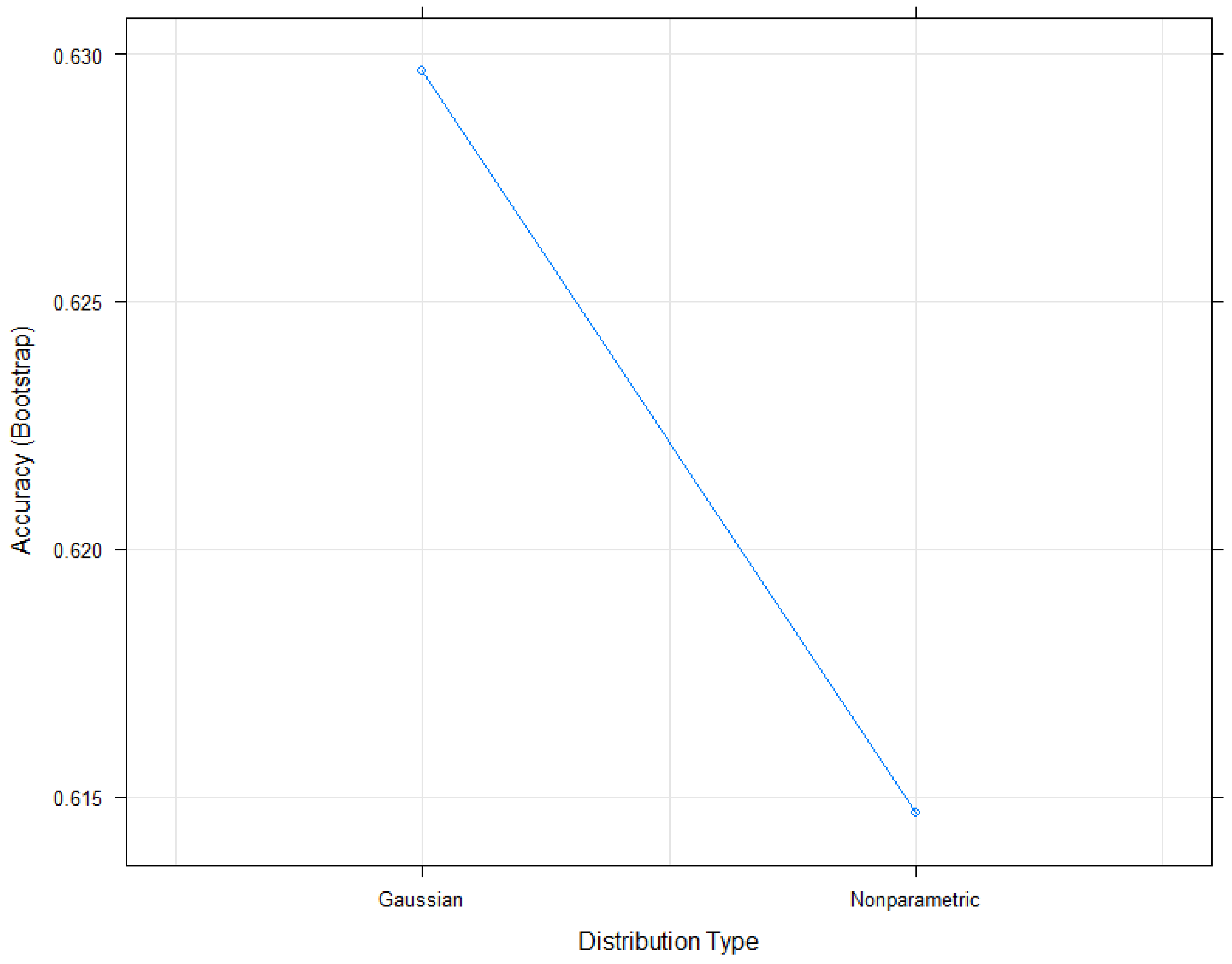
```
#Bootstap
train_control <- trainControl(method="boot", number=10)
# train the model
model1 <- train(covid_res~sex + patient_type + pneumonia, data=train1, trControl=train_control,
method="nb")
# summarize results
print(model1)
plot(model1)
```

For Bootstrap, we used method boot for bootstrap with 10 resampling to make Naïve Bayes model.



```
Naive Bayes

4942 samples
   3 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (10 reps)
Summary of sample sizes: 4942, 4942, 4942, 4942, 4942, 4942, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.6296647  0.1978874
   TRUE      0.6146677  0.1388817

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at
 a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 1.
```

After applying bootstrap to Naïve Bayes, the TRUE accuracy increased from 60.54 to 61.46 and FALSE accuracy increased from 62.69 to 62.96



Conclusion:
Resampling is the method that consists of drawing repeated samples from the original data samples. The process of Resampling is a nonparametric method of statistical inference. Resampling involves selecting randomized cases with replacement from the original data sample in such a manner that each number of the sample drawn has several cases like the original data sample. Resampling methods like K cross fold , LOOCV, Bootstrapping , Validation are used to build the Naïve Bayes Model , KNN and Logistic Regression.

References

1. https://rstudio-pubs- static.s3.amazonaws.com/21668_28239bbc1ff34bc99f062f3241ca3a97.html

2. http://www2.stat.duke.edu/~rcs46/lectures_2017/05-resample/05-cv.pdf