R-MINI PROJECT

PROJECT TOPIC: **Prediction and Analysis on Diabetes Dataset**

Project Description:

The system will allow us to predict if the patient has diabetes on the basis of certain diagnostic measures available in the dataset. The different steps involved in EDA include: 1.Data Collection, 2.Data Cleaning and 3.Data Visualization. This project first conducts Exploratory Data Analysis (EDA) and data visualization on the diabetes dataset and then predicts the diabetes.

Exploratory Data Analysis (EDA)

1. Descriptive statistics

Attribute type, Class distribution, Mean, Standard Deviation, Median, Quartile, Skewness, and Correlation.

2. Data Visualization

Histogram plot

Density plot

Box and Whisker plot

Bar plot

Missing data map

Pairwise correlation plot

Prediction on Diabetes

We compare the performance for the following classifiers:

1. Logistic Regression

   Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for given set of features (or inputs), X.

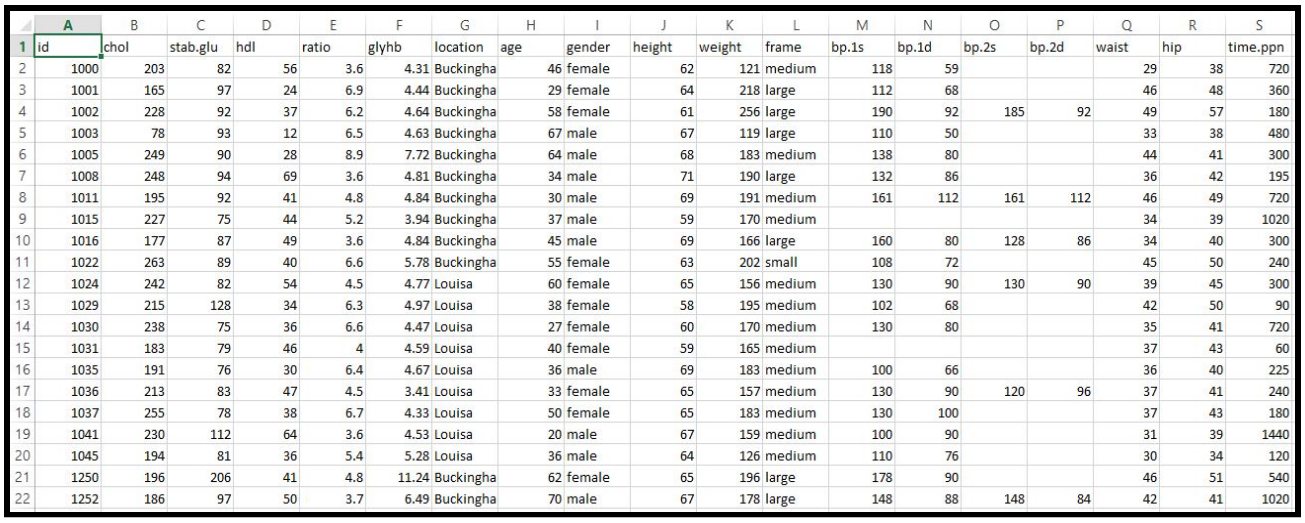
2. Support Vector Machine (SVM)

   “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well

3. Random Forest

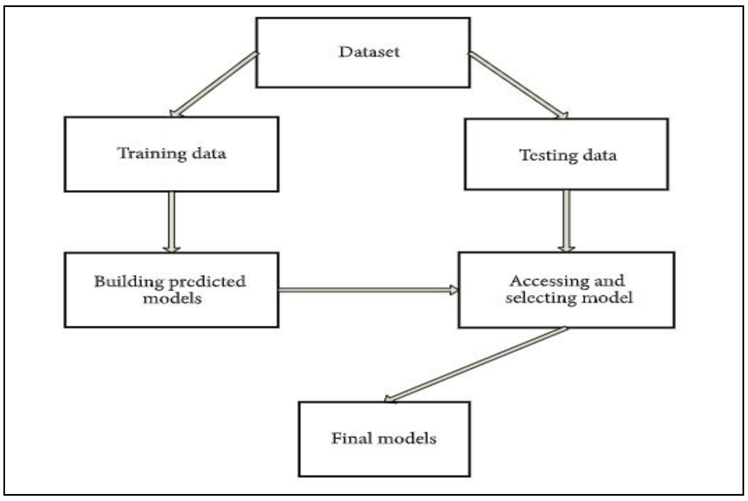
   The random forest algorithm works by aggregating the predictions made by multiple decision trees of varying depth. Every decision tree in the forest is trained on a subset of the dataset called the bootstrapped dataset.

Dataset:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	id	chol	stab.glu	hdl	ratio	glyhb	location	age	gender	height	weight	frame	bp.1s	bp.1d	bp.2s	bp.2d	waist	hip	time.ppn
2	1000	203	82	56	3.6	4.31	Buckingha	46	female	62	121	medium	118	59			29	38	720
3	1001	165	97	24	6.9	4.44	Buckingha	29	female	64	218	large	112	68			46	48	360
4	1002	228	92	37	6.2	4.64	Buckingha	58	female	61	256	large	190	92	185	92	49	57	180
5	1003	78	93	12	6.5	4.63	Buckingha	67	male	67	119	large	110	50			33	38	480
6	1005	249	90	28	8.9	7.72	Buckingha	64	male	68	183	medium	138	80			44	41	300
7	1008	248	94	69	3.6	4.81	Buckingha	34	male	71	190	large	132	86			36	42	195
8	1011	195	92	41	4.8	4.84	Buckingha	30	male	69	191	medium	161	112	161	112	46	49	720
9	1015	227	75	44	5.2	3.94	Buckingha	37	male	59	170	medium					34	39	1020
10	1016	177	87	49	3.6	4.84	Buckingha	45	male	69	166	large	160	80	128	86	34	40	300
11	1022	263	89	40	6.6	5.78	Buckingha	55	female	63	202	small	108	72			45	50	240
12	1024	242	82	54	4.5	4.77	Louisa	60	female	65	156	medium	130	90	130	90	39	45	300
13	1029	215	128	34	6.3	4.97	Louisa	38	female	58	195	medium	102	68			42	50	90
14	1030	238	75	36	6.6	4.47	Louisa	27	female	60	170	medium	130	80			35	41	720
15	1031	183	79	46	4	4.59	Louisa	40	female	59	165	medium					37	43	60
16	1035	191	76	30	6.4	4.67	Louisa	36	male	69	183	medium	100	66			36	40	225
17	1036	213	83	47	4.5	3.41	Louisa	33	female	65	157	medium	130	90	120	96	37	41	240
18	1037	255	78	38	6.7	4.33	Louisa	50	female	65	183	medium	130	100			37	43	180
19	1041	230	112	64	3.6	4.53	Louisa	20	male	67	159	medium	100	90			31	39	1440
20	1045	194	81	36	5.4	5.28	Louisa	36	male	64	126	medium	110	76			30	34	120
21	1250	196	206	41	4.8	11.24	Buckingha	62	female	65	196	large	178	90			46	51	540
22	1252	186	97	50	3.7	6.49	Buckingha	70	male	67	178	large	148	88	148	84	42	41	1020

Our research dataset is divided into two parts; two-thirds of the data is used as a training set, and one-third of the dataset is defined as a testing set to evaluate the performance of several classifiers. All classifiers were fitted to the same training and testing data. The specific process is: .



Libraries:

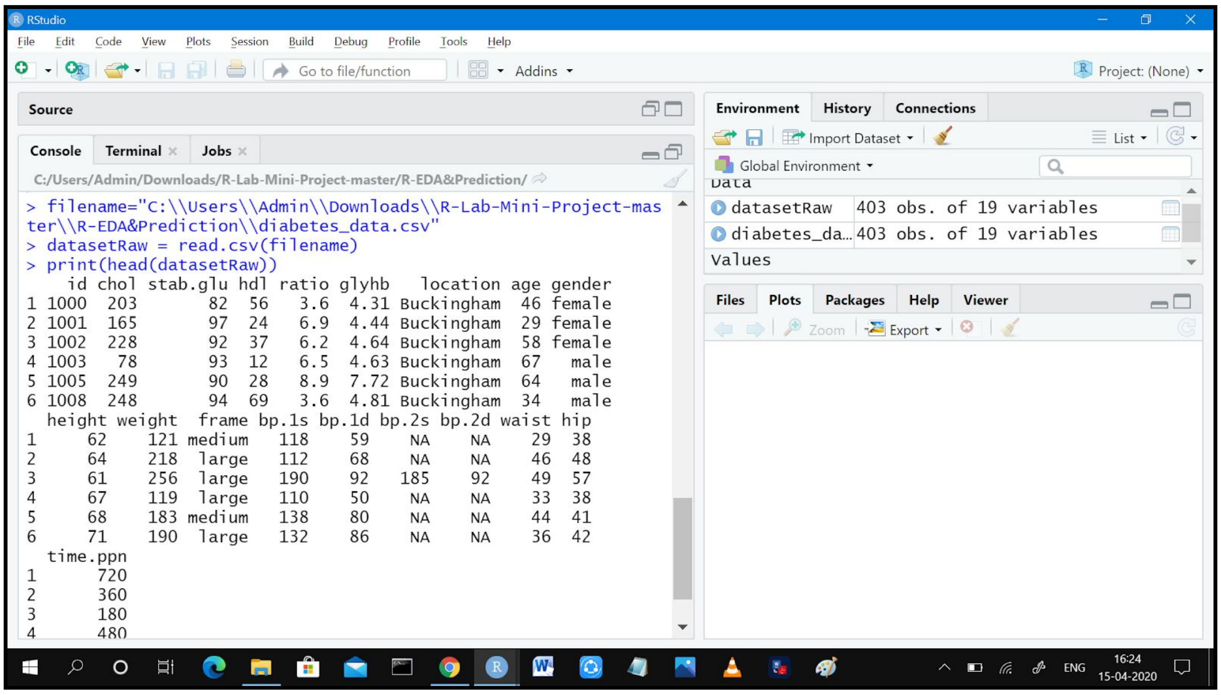
- **Randomforest**
  Random Forest implements Breiman's random forest algorithm (based on Breiman's and Cutler's original FORTRAN code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points.

- **Caret**
  The caret package (short for Classification And REgression Training) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:
  - data splitting
  - pre-processing
  - feature selection
  - model tuning using resampling
  - variable importance estimation
  - as well as other functionality.

Output:

Step1: Load the data

```
> filename="C:\\Users\\Admin\\Desktop\\\\ R-Lab-Mini-Project \\diabetes_data.csv"
> datasetRaw = read.csv(filename)
> print(head(datasetRaw))
```



C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/

```
> filename="C:\\Users\\Admin\\Downloads\\R-Lab-Mini-Project-mas
ter\\R-EDA&Prediction\\diabetes_data.csv"
> datasetRaw = read.csv(filename)
> print(head(datasetRaw))
    id chol stab.glu hdl ratio glyhb   location age gender
1 1000  203       82  56   3.6  4.31 Buckingham  46 female
2 1001  165       97  24   6.9  4.44 Buckingham  29 female
3 1002  228       92  37   6.2  4.64 Buckingham  58 female
4 1003   78       93  12   6.5  4.63 Buckingham  67   male
5 1005  249       90  28   8.9  7.72 Buckingham  64   male
6 1008  248       94  69   3.6  4.81 Buckingham  34   male
  height weight  frame bp.1s bp.1d bp.2s bp.2d waist hip
1     62    121 medium   118    59    NA    NA    29  38
2     64    218  large   112    68    NA    NA    46  48
3     61    256  large   190    92   185    92    49  57
4     67    119  large   110    50    NA    NA    33  38
5     68    183 medium   138    80    NA    NA    44  41
6     71    190  large   132    86    NA    NA    36  42
  time.ppn
1      720
2      360
3      180
4      480
```

```
Data
datasetRaw      403 obs. of 19 variables
diabetes_da…    403 obs. of 19 variables
Values
```

Step 2: Clean the data

```
> numColumns = dim(datasetRaw)[2]
> vector_NAs = rep(0, numColumns)
> for (i in 1:numColumns) {
+   vector_NAs[i] = sum(is.na(datasetRaw[,i]))
+   }
> print("The missing values in each column:")
> print(vector_NAs)
```

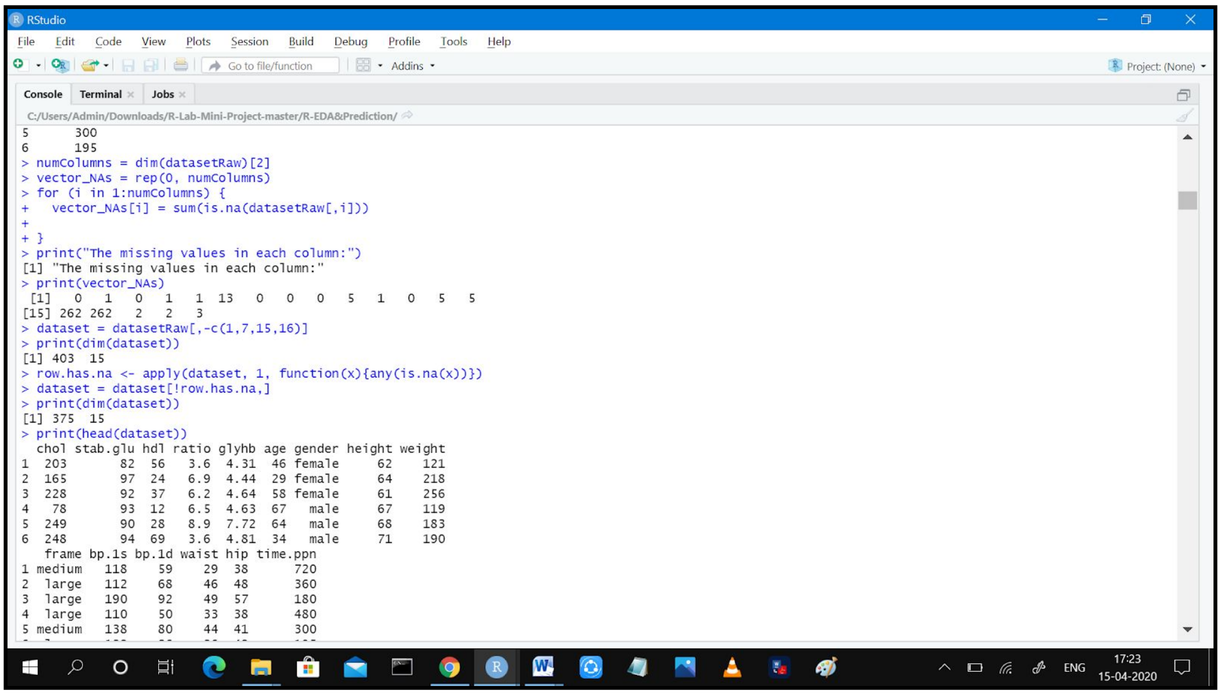
# deletes columns 15 and 16 due to many missing values

# deletes column 1 (id), column 7 (location) because they contain no useful information

```
> print(dim(dataset))
```

remove the row with missing values

```
> row.has.na <- apply(dataset, 1, function(x){any(is.na(x))})
> dataset = dataset[!row.has.na,]
> print(dim(dataset))
> print(head(dataset))
```



```
5       300
6       195
> numColumns = dim(datasetRaw)[2]
> vector_NAs = rep(0, numColumns)
> for (i in 1:numColumns) {
+   vector_NAs[i] = sum(is.na(datasetRaw[,i]))
+ 
+ }
> print("The missing values in each column:")
[1] "The missing values in each column:"
> print(vector_NAs)
 [1]   0   1   0   1   1  13   0   0   0   5   1   0   5   5
[15] 262 262   2   2   3
> dataset = datasetRaw[,-c(1,7,15,16)]
> print(dim(dataset))
[1] 403  15
> row.has.na <- apply(dataset, 1, function(x){any(is.na(x))})
> dataset = dataset[!row.has.na,]
> print(dim(dataset))
[1] 375  15
> print(head(dataset))
  chol stab.glu hdl ratio glyhb age gender height weight
1  203       82  56   3.6  4.31  46 female     62    121
2  165       97  24   6.9  4.44  29 female     64    218
3  228       92  37   6.2  4.64  58 female     61    256
4   78       93  12   6.5  4.63  67   male     67    119
5  249       90  28   8.9  7.72  64   male     68    183
6  248       94  69   3.6  4.81  34   male     71    190
   frame bp.1s bp.1d waist hip time.ppn
1 medium   118    59    29  38      720
2  large   112    68    46  48      360
3  large   190    92    49  57      180
4  large   110    50    33  38      480
5 medium   138    80    44  41      300
```

# encodes the class label (column 5): Glycosylated haemoglobin > 7.0 is taken as a positive diagnosis of diabetes.

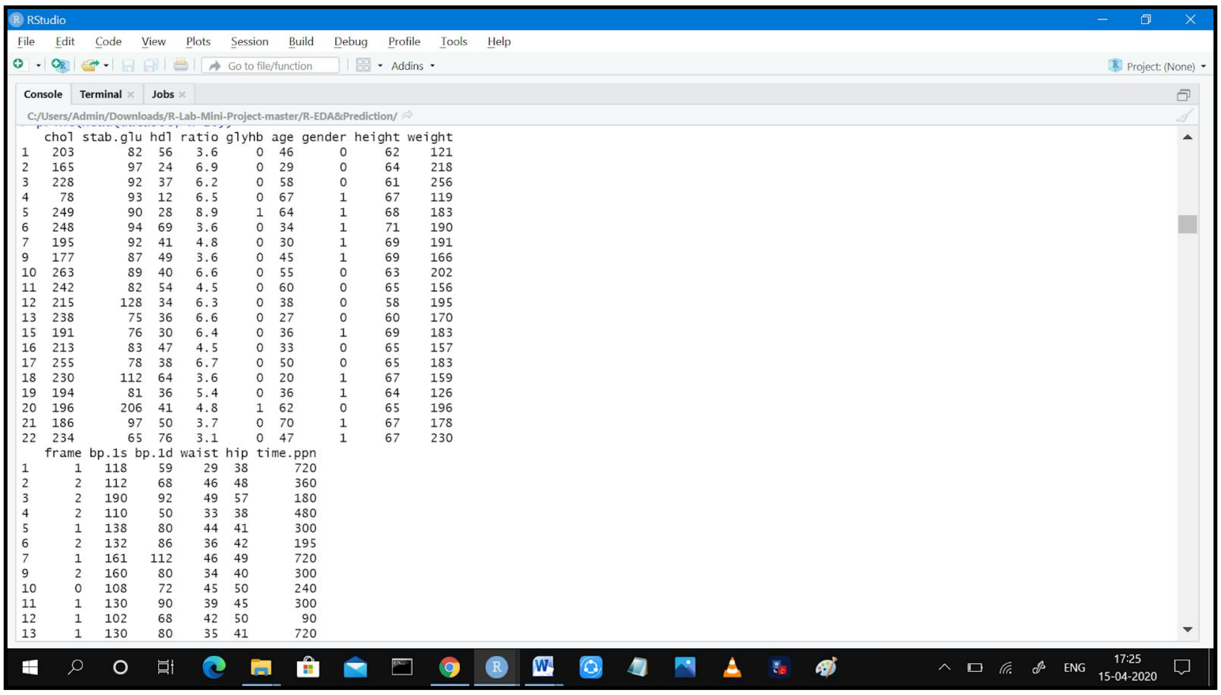
```
> dataset [,5] = if else(dataset[,5] >= 7.0, 1, 0)
> dataset [,5] = factor(dataset[,5])
```

# encode the categorical data (column-7 gender)

```
> dataset[,7] = if else(dataset[,7] == "female", 0, 1)
> dataset[,7] = factor(dataset[,7])
```

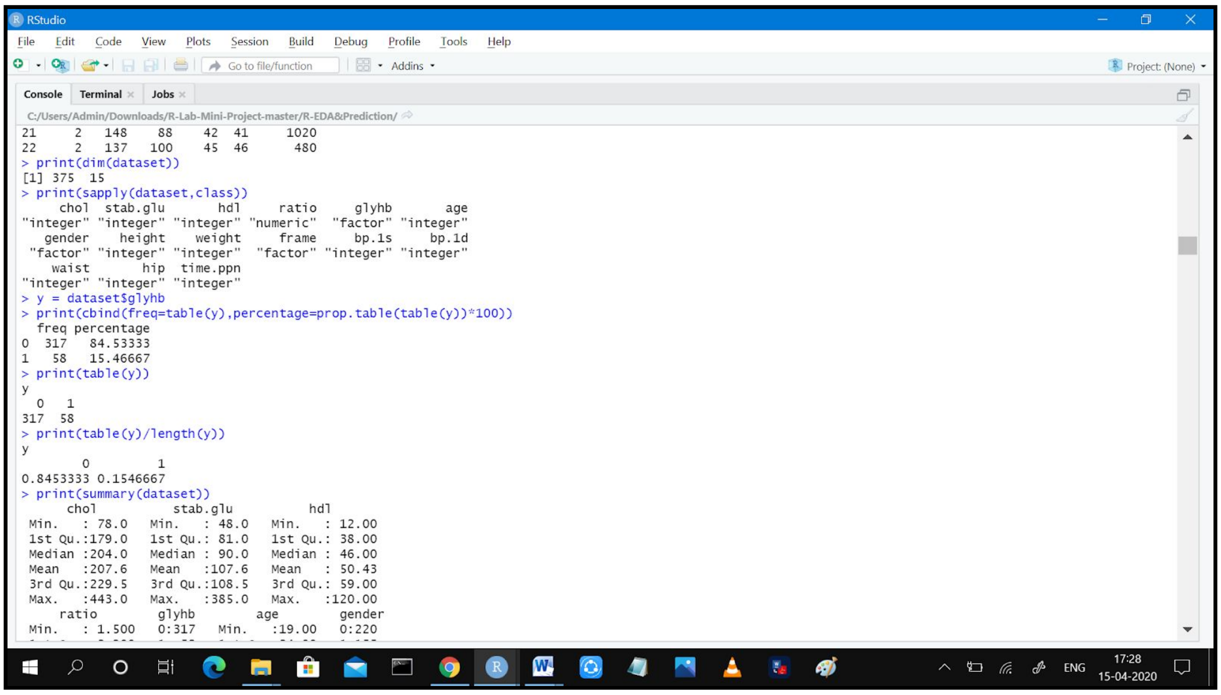
# encode the categorical data (column-10 frame)

```
> dataset[,10] = if else(dataset[,10] == "small", 0, if else(dataset[,10] == "medium", 1,2) )
> dataset[,10] = factor(dataset[,10])
# Descriptive statistics %%%%%%
# display the first 20 rows
> print(head(dataset, n=20))
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
Jobs
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
ENG
17:25
15-04-2020

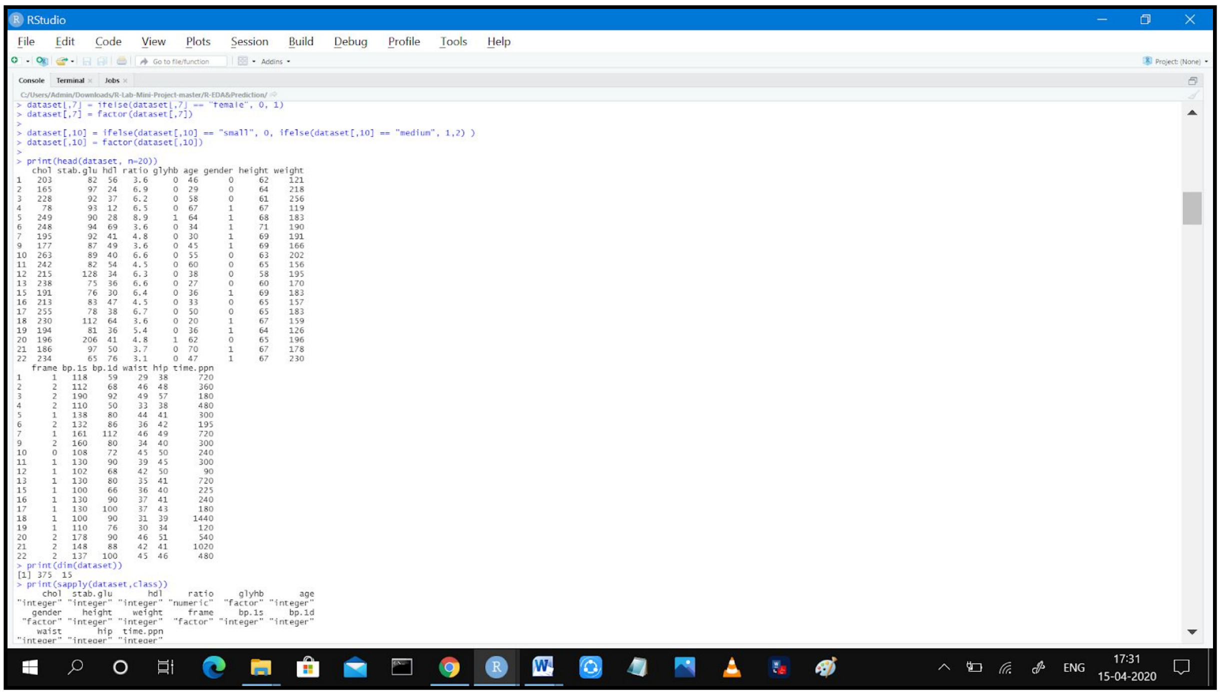
```
   chol stab.glu hdl ratio glyhb age gender height weight
1   203       82  56   3.6     0  46      0     62    121
2   165       97  24   6.9     0  29      0     64    218
3   228       92  37   6.2     0  58      0     61    256
4    78       93  12   6.5     0  67      1     67    119
5   249       90  28   8.9     1  64      1     68    183
6   248       94  69   3.6     0  34      1     71    190
7   195       92  41   4.8     0  30      1     69    191
9   177       87  49   3.6     0  45      1     69    166
10  263       89  40   6.6     0  55      0     63    202
11  242       82  54   4.5     0  60      0     65    156
12  215      128  34   6.3     0  38      0     58    195
13  238       75  36   6.6     0  27      0     60    170
15  191       76  30   6.4     0  36      1     69    183
16  213       83  47   4.5     0  33      0     65    157
17  255       78  38   6.7     0  50      0     65    183
18  230      112  64   3.6     0  20      1     67    159
19  194       81  36   5.4     0  36      1     64    126
20  196      206  41   4.8     1  62      0     65    196
21  186       97  50   3.7     0  70      1     67    178
22  234       65  76   3.1     0  47      1     67    230
   frame bp.1s bp.1d waist hip time.ppn
1      1   118    59    29  38      720
2      2   112    68    46  48      360
3      2   190    92    49  57      180
4      2   110    50    33  38      480
5      1   138    80    44  41      300
6      2   132    86    36  42      195
7      1   161   112    46  49      720
9      2   160    80    34  40      300
10     0   108    72    45  50      240
11     1   130    90    39  45      300
12     1   102    68    42  50       90
13     1   130    80    35  41      720
```

```
# display the dimensions of the dataset
> print(dim(dataset))
 [1] 375  15
> print(head(dataset))
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
Jobs
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
ENG
17:28
15-04-2020

```
21      2   148    88    42  41     1020
22      2   137   100    45  46      480
> print(dim(dataset))
[1] 375  15
> print(sapply(dataset,class))
     chol  stab.glu       hdl     ratio     glyhb       age
"integer" "integer" "integer" "numeric"  "factor" "integer"
   gender    height    weight     frame     bp.1s     bp.1d
 "factor" "integer" "integer"  "factor" "integer" "integer"
    waist       hip  time.ppn
"integer" "integer" "integer"
> y = dataset$glyhb
> print(cbind(freq=table(y),percentage=prop.table(table(y))*100))
  freq percentage
0  317   84.53333
1   58   15.46667
> print(table(y))
y
  0   1
317  58
> print(table(y)/length(y))
y
        0         1
0.8453333 0.1546667
> print(summary(dataset))
      chol          stab.glu          hdl
 Min.   : 78.0   Min.   : 48.0   Min.   : 12.00
 1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00
 Median :204.0   Median : 90.0   Median : 46.00
 Mean   :207.6   Mean   :107.6   Mean   : 50.43
 3rd Qu.:229.5   3rd Qu.:108.5   3rd Qu.: 59.00
 Max.   :443.0   Max.   :385.0   Max.   :120.00
     ratio         glyhb        age          gender
 Min.   : 1.500   0:317   Min.   :19.00   0:220
```

```
> # encode the class label (column 5): Glycosylated hemoglobin  > 7.0 is taken as a positive
diagnosis of diabetes.
> dataset[,5] = ifelse(dataset[,5] >= 7.0, 1, 0)
> dataset[,5] = factor(dataset[,5])

> # encode the categorical data (column-7 gender)
> dataset[,7] = ifelse(dataset[,7] == "female", 0, 1)
> dataset[,7] = factor(dataset[,7])

> # encode the categorical data (column-10 frame)
> dataset[,10] = ifelse(dataset[,10] == "small", 0, ifelse(dataset[,10] == "medium", 1,2) )
> dataset[,10] = factor(dataset[,10])

> # Descriptive statistics %%%%%%%

> # display the first 20 rows
> print(head(dataset, n=20))
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Console Terminal Jobs
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
17:31
15-04-2020

```
> dataset[,7] = ifelse(dataset[,7] == "female", 0, 1)
> dataset[,7] = factor(dataset[,7])
>
> dataset[,10] = ifelse(dataset[,10] == "small", 0, ifelse(dataset[,10] == "medium", 1,2) )
> dataset[,10] = factor(dataset[,10])
>
> print(head(dataset, n=20))
   chol stab.glu hdl ratio glyhb age gender height weight
1   203       82  56   3.6     0  46      0     62    121
2   165       97  24   6.9     0  29      0     64    218
3   228       92  37   6.2     0  58      0     61    256
4    78       93  12   6.5     0  67      1     67    119
5   249       90  28   8.9     1  64      1     68    183
6   248       94  69   3.6     0  34      1     71    190
7   195       92  41   4.8     0  30      1     69    191
9   177       87  49   3.6     0  45      1     69    166
10  263       89  40   6.6     0  55      0     63    202
11  242       82  54   4.5     0  60      0     65    156
12  215      128  34   6.3     0  38      0     58    195
13  238       75  36   6.6     0  27      0     60    170
15  191       76  30   6.4     0  36      1     69    183
16  213       83  47   4.5     0  33      0     65    157
17  255       78  38   6.7     0  50      0     65    183
18  230      112  64   3.6     0  20      1     67    159
19  194       81  36   5.4     0  36      1     64    126
20  196      206  41   4.8     1  62      0     65    196
21  186       97  50   3.7     0  70      1     67    178
22  234       65  76   3.1     0  47      1     67    230
   frame bp.1s bp.1d waist hip time.ppn
1      1   118    59    29  38      720
2      2   112    68    46  48      360
3      2   190    92    49  57      180
4      2   110    50    33  38      480
5      1   138    80    44  41      300
6      2   132    86    36  42      195
7      1   161   112    46  49      720
9      2   160    80    34  40      300
10     0   108    72    45  50      240
11     1   130    90    39  45      300
12     1   102    68    42  50       90
13     1   130    80    35  41      720
15     1   100    66    36  40      225
16     1   130    90    37  41      240
17     1   130   100    37  43      180
18     1   100    90    31  39     1440
19     1   110    76    30  34      120
20     2   178    90    46  51      540
21     2   148    88    42  41     1020
22     2   137   100    45  46      480
> print(dim(dataset))
[1] 375  15
> print(sapply(dataset,class))
     chol  stab.glu       hdl     ratio     glyhb       age 
"integer" "integer" "integer" "numeric"  "factor" "integer" 
   gender    height    weight     frame     bp.1s     bp.1d 
 "factor" "integer" "integer"  "factor" "integer" "integer" 
    waist       hip  time.ppn 
"integer" "integer" "integer" 
```

```
> # display the dimensions of the dataset
> print(dim(dataset))
```

```
[1] 375  15
```

```
> # list types for each attribute
> print(sapply(dataset, class))
```

```
    chol  stab.glu    hdl    ratio    glyhb     age   gender   height   weight    frame    bp.1s
"integer" "integer" "integer" "numeric"  "factor" "integer"  "factor" "integer" "integer"  "factor"
"integer"
   bp.1d     waist      hip  time.ppn
"integer" "integer" "integer" "integer"
```

```
> # distribution of the class labels
> y = dataset$glyhb
> print(cbind(freq=table(y),percentage=prop.table(table(y))*100))
```

```
freq percentage
0  317   84.53333
1   58   15.46667
```

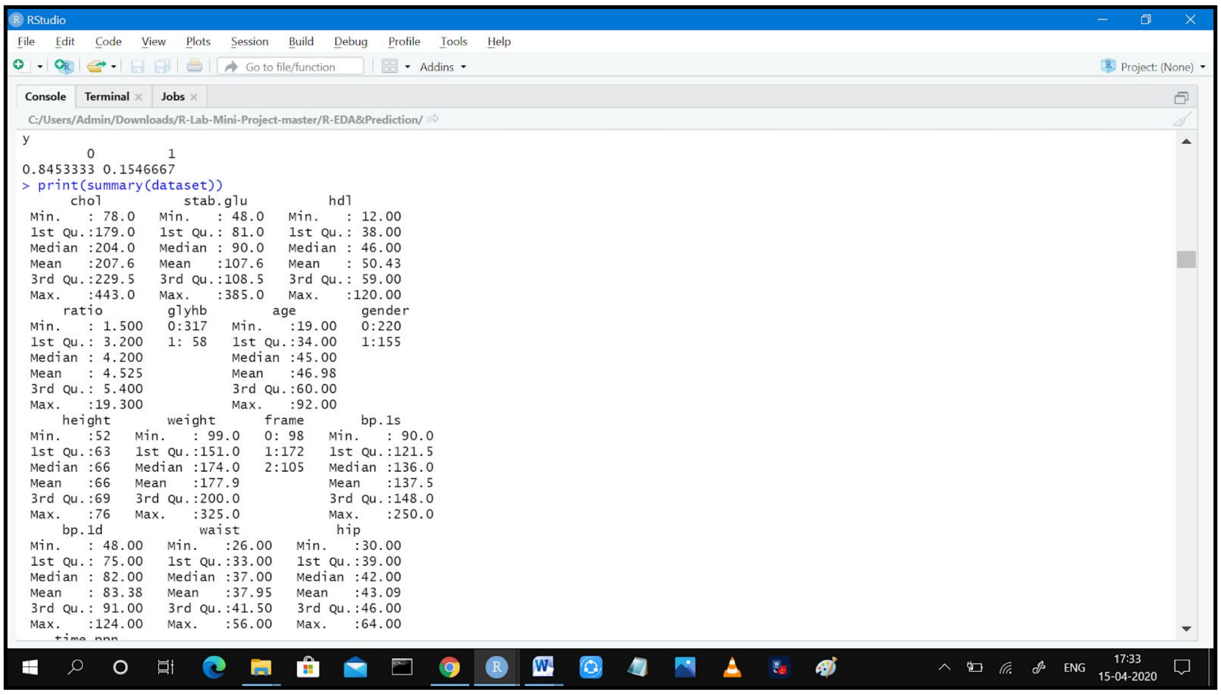
```
> print(table(y))
```

```
y
  0   1
317  58
```

```
> print(table(y)/length(y))
```

```
y
        0         1
0.8453333 0.1546667
```

```
> # summarize the dataset
> print(summary(dataset))
```



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
Jobs
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
ENG
17:33
15-04-2020

```
y
        0         1
0.8453333 0.1546667
> print(summary(dataset))
      chol          stab.glu          hdl
 Min.   : 78.0   Min.   : 48.0   Min.   : 12.00
 1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00
 Median :204.0   Median : 90.0   Median : 46.00
 Mean   :207.6   Mean   :107.6   Mean   : 50.43
 3rd Qu.:229.5   3rd Qu.:108.5   3rd Qu.: 59.00
 Max.   :443.0   Max.   :385.0   Max.   :120.00
     ratio        glyhb        age         gender
 Min.   : 1.500   0:317   Min.   :19.00   0:220
 1st Qu.: 3.200   1: 58   1st Qu.:34.00   1:155
 Median : 4.200           Median :45.00
 Mean   : 4.525           Mean   :46.98
 3rd Qu.: 5.400           3rd Qu.:60.00
 Max.   :19.300           Max.   :92.00
     height        weight       frame       bp.1s
 Min.   :52   Min.   : 99.0   0: 98   Min.   : 90.0
 1st Qu.:63   1st Qu.:151.0   1:172   1st Qu.:121.5
 Median :66   Median :174.0   2:105   Median :136.0
 Mean   :66   Mean   :177.9           Mean   :137.5
 3rd Qu.:69   3rd Qu.:200.0           3rd Qu.:148.0
 Max.   :76   Max.   :325.0           Max.   :250.0
     bp.1d            waist            hip
 Min.   : 48.00   Min.   :26.00   Min.   :30.00
 1st Qu.: 75.00   1st Qu.:33.00   1st Qu.:39.00
 Median : 82.00   Median :37.00   Median :42.00
 Mean   : 83.38   Mean   :37.95   Mean   :43.09
 3rd Qu.: 91.00   3rd Qu.:41.50   3rd Qu.:46.00
 Max.   :124.00   Max.   :56.00   Max.   :64.00
    time.ppn
```

Step 3: To Do EDA (Analysis of the dataset)

```
 # Standard Deviations for the non-categorical columns
> std=sapply(dataset[,-c(5,7,10)],sd)
> print('The standard deviations are:')
[1] "The standard deviations are:"

> print(std)
```

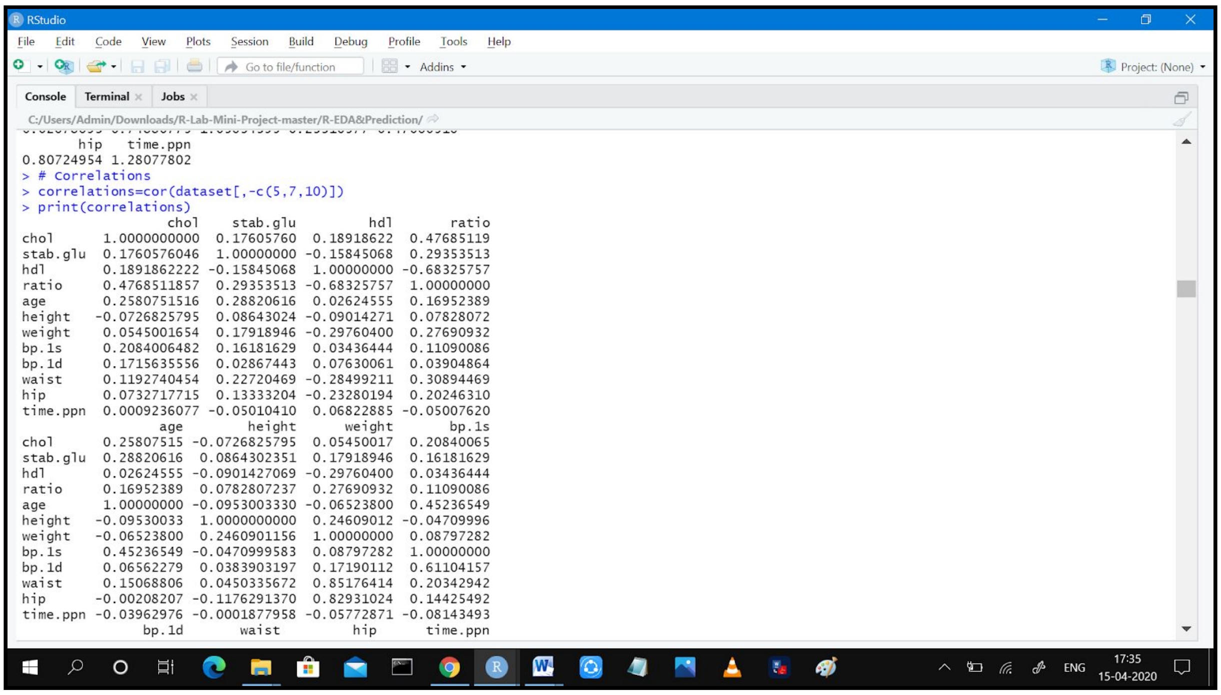
```
chol   stab.glu   hdl     ratio     age     height     weight     bp.1s
44.700780  54.082496  17.444346   1.755499  16.661203   3.915210  40.568940  23.178154
    bp.1d      waist       hip   time.ppn
 13.544167   5.777105   5.642679 309.056806
```

```
# Skewness
#The further the distribution of the skew value from zero,
# the larger the skew to the left (negative skew value) or right (positive skew value).

> library(e1071) # the library for skewness
> skew=apply(dataset[,-c(5,7,10)], 2, skewness)
> print(skew)
```

```
      chol   stab.glu        hdl      ratio        age     height     weight      bp.1s
0.97739823 2.69790949 1.21275829 2.24132546 0.30061280 0.02678693 0.74880775 1.05634395
     bp.1d      waist        hip   time.ppn
0.23310577 0.47060516 0.80724954 1.28077802
```

CORRELATION:

```
# Correlations
> correlations=cor(dataset[,-c(5,7,10)])
> print(correlations)
```

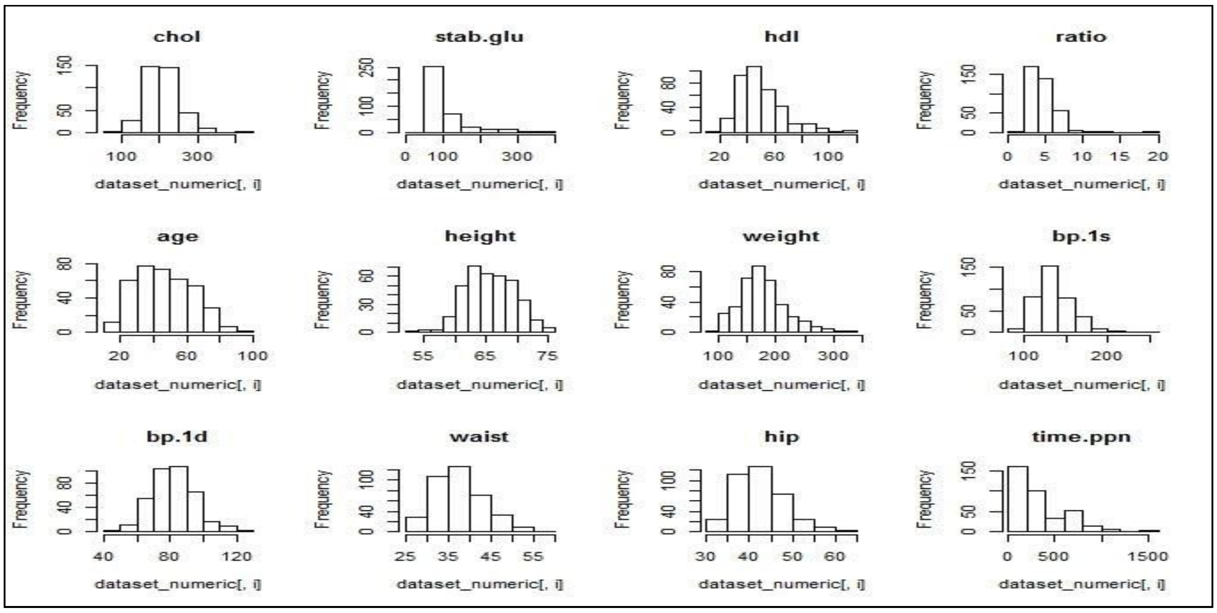


```
       hip   time.ppn
0.80724954 1.28077802
> # Correlations
> correlations=cor(dataset[,-c(5,7,10)])
> print(correlations)
                 chol    stab.glu         hdl       ratio
chol     1.0000000000  0.17605760  0.18918622  0.47685119
stab.glu 0.1760576046  1.00000000 -0.15845068  0.29353513
hdl      0.1891862222 -0.15845068  1.00000000 -0.68325757
ratio    0.4768511857  0.29353513 -0.68325757  1.00000000
age      0.2580751516  0.28820616  0.02624555  0.16952389
height  -0.0726825795  0.08643024 -0.09014271  0.07828072
weight   0.0545001654  0.17918946 -0.29760400  0.27690932
bp.1s    0.2084006482  0.16181629  0.03436444  0.11090086
bp.1d    0.1715635556  0.02867443  0.07630061  0.03904864
waist    0.1192740454  0.22720469 -0.28499211  0.30894469
hip      0.0732717715  0.13333204 -0.23280194  0.20246310
time.ppn 0.0009236077 -0.05010410  0.06822885 -0.05007620
                 age       height      weight       bp.1s
chol      0.25807515 -0.0726825795  0.05450017  0.20840065
stab.glu  0.28820616  0.0864302351  0.17918946  0.16181629
hdl       0.02624555 -0.0901427069 -0.29760400  0.03436444
ratio     0.16952389  0.0782807237  0.27690932  0.11090086
age       1.00000000 -0.0953003330 -0.06523800  0.45236549
height   -0.09530033  1.0000000000  0.24609012 -0.04709996
weight   -0.06523800  0.2460901156  1.00000000  0.08797282
bp.1s     0.45236549 -0.0470999583  0.08797282  1.00000000
bp.1d     0.06562279  0.0383903197  0.17190112  0.61104157
waist     0.15068806  0.0450335672  0.85176414  0.20342942
hip      -0.00208207 -0.1176291370  0.82931024  0.14425492
time.ppn -0.03962976 -0.0001877958 -0.05772871 -0.08143493
               bp.1d         waist         hip    time.ppn
```

PART 2 : Data visualizations:

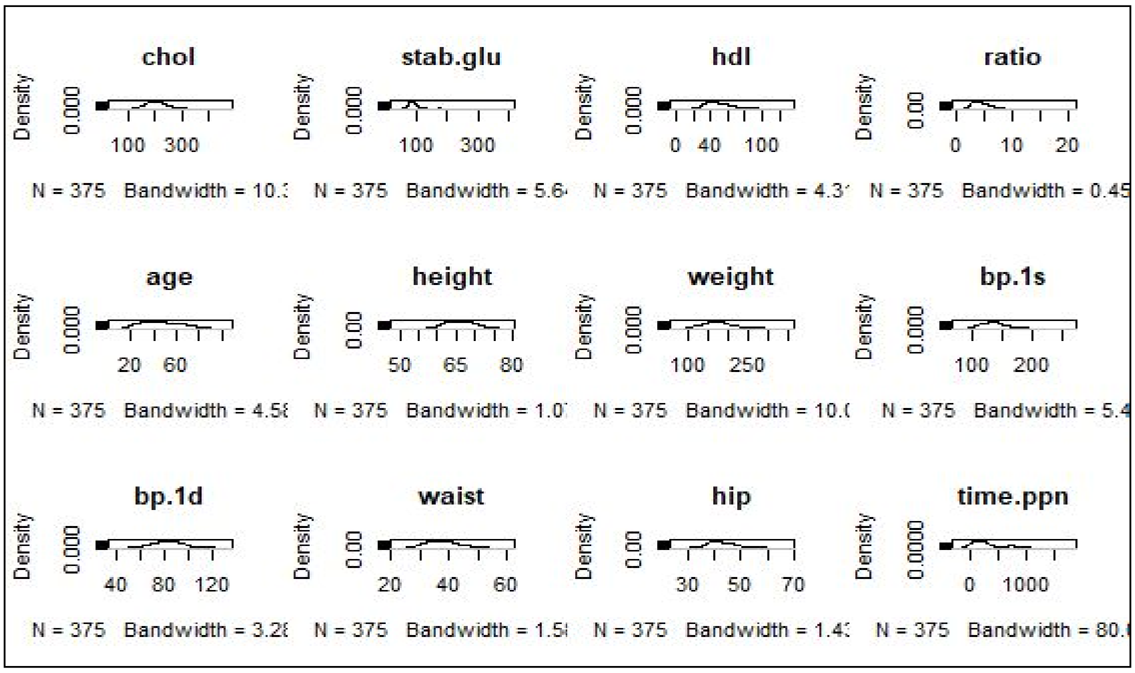
1:Histogram

```
> dataset_numeric = dataset[,-c(5,7,10)]
> #Histograms
> par(mfrow=c(3,4)) # put four figures in a row (2*4)
> for (i in 1:12) {
+   hist(dataset_numeric[,i],main=names(dataset_numeric)[i])
+ }
```



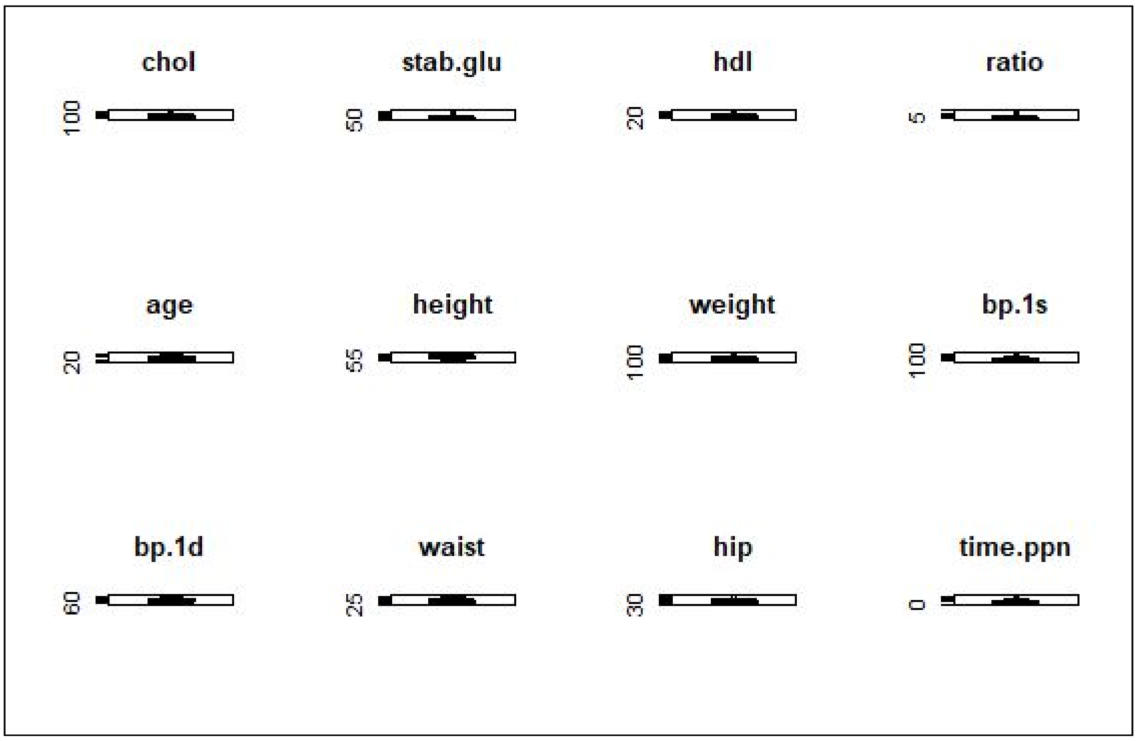
2:Density Plots

```
> par(mfrow=c(3,4))
> for(i in 1:12) {
+   plot(density(dataset_numeric[,i]), main=names(dataset_numeric)[i])
+ }
```



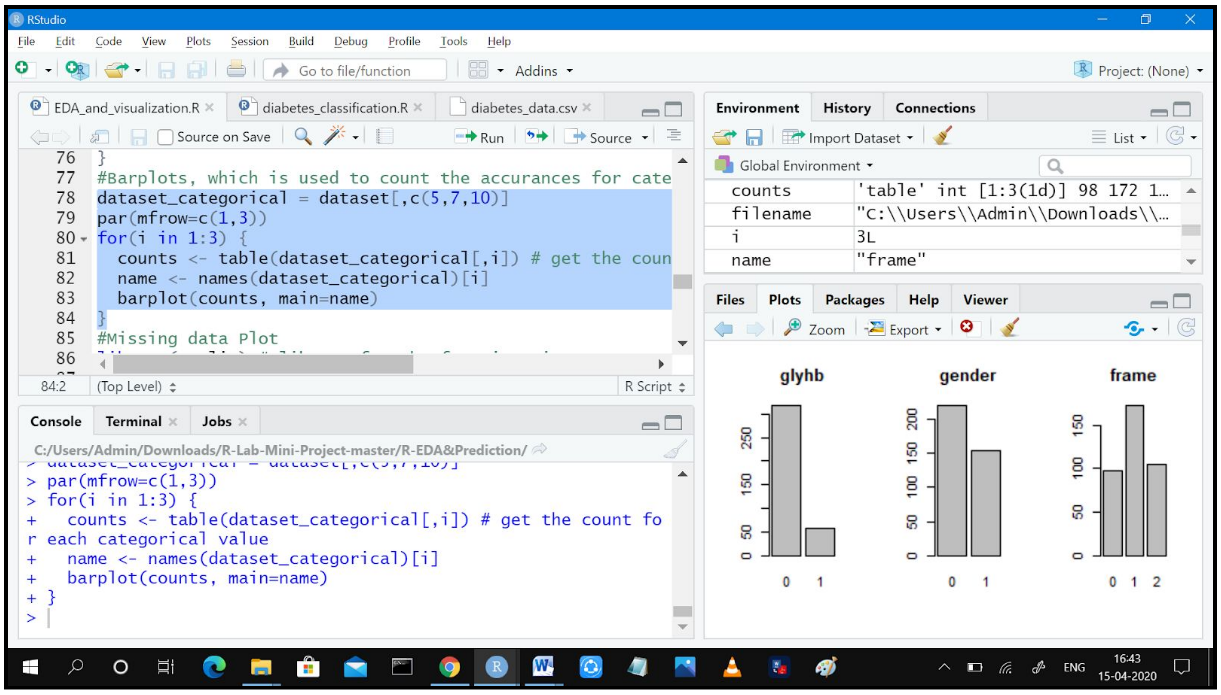
3:Box And Whisker Plots

```
> par(mfrow=c(3,4))
> for(i in 1:12) {
+   boxplot(dataset_numeric[,i], main=names(dataset_numeric)[i])
+ }
```



4:Barplots:

```
> dataset_categorical = dataset[,c(5,7,10)]
> par(mfrow=c(1,3))
> for(i in 1:3) {
+   counts <- table(dataset_categorical[,i]) # get the count for each categorical value
+   name <- names(dataset_categorical)[i]
+   barplot(counts, main=name)
+ }
```



```
}
#Barplots, which is used to count the accurances for cate
dataset_categorical = dataset[,c(5,7,10)]
par(mfrow=c(1,3))
for(i in 1:3) {
  counts <- table(dataset_categorical[,i]) # get the coun
  name <- names(dataset_categorical)[i]
  barplot(counts, main=name)
}
#Missing data Plot
```

C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/

```
> par(mfrow=c(1,3))
> for(i in 1:3) {
+   counts <- table(dataset_categorical[,i]) # get the count fo
r each categorical value
+   name <- names(dataset_categorical)[i]
+   barplot(counts, main=name)
+ }
>
```

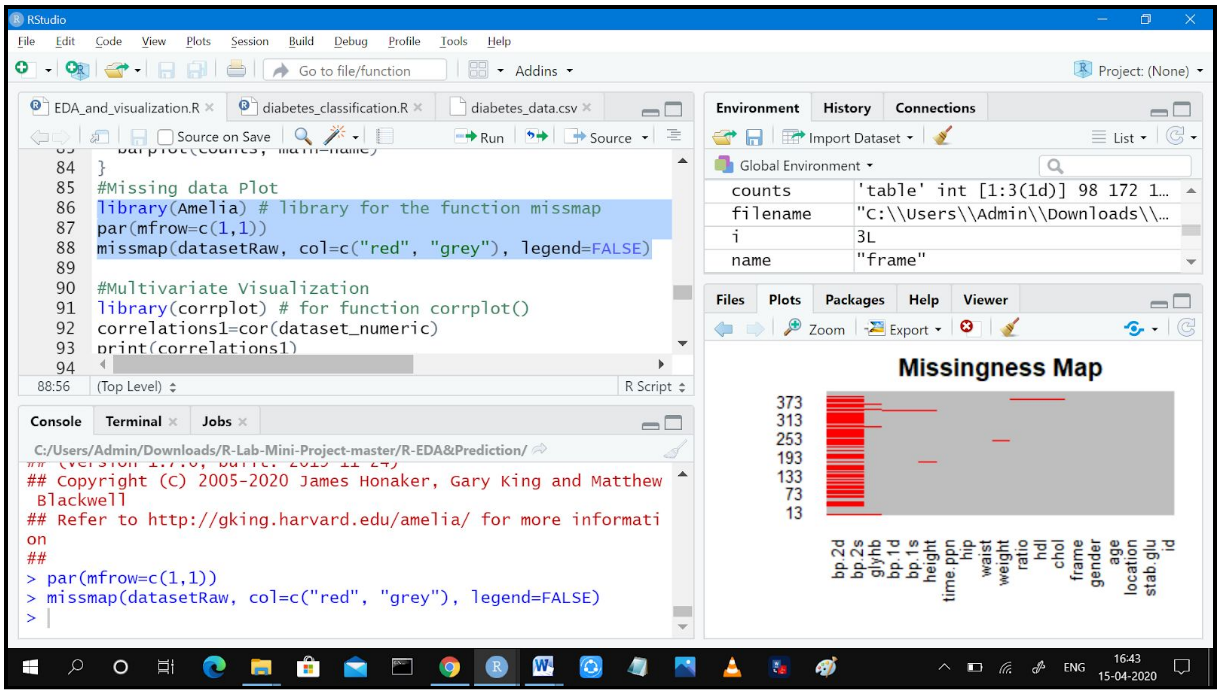
counts	'table' int [1:3(1d)] 98 172 1…
filename	"C:\\Users\\Admin\\Downloads\\…
i	3L
name	"frame"

5:Missing data Plot

> library (Amelia) # library for the function missmap

> par(mfrow=c(1,1))

> missmap (datasetRaw, col=c("red", "grey"), legend=FALSE)



```
}
#Missing data Plot
library(Amelia) # library for the function missmap
par(mfrow=c(1,1))
missmap(datasetRaw, col=c("red", "grey"), legend=FALSE)

#Multivariate Visualization
library(corrplot) # for function corrplot()
correlations1=cor(dataset_numeric)
print(correlations1)
```

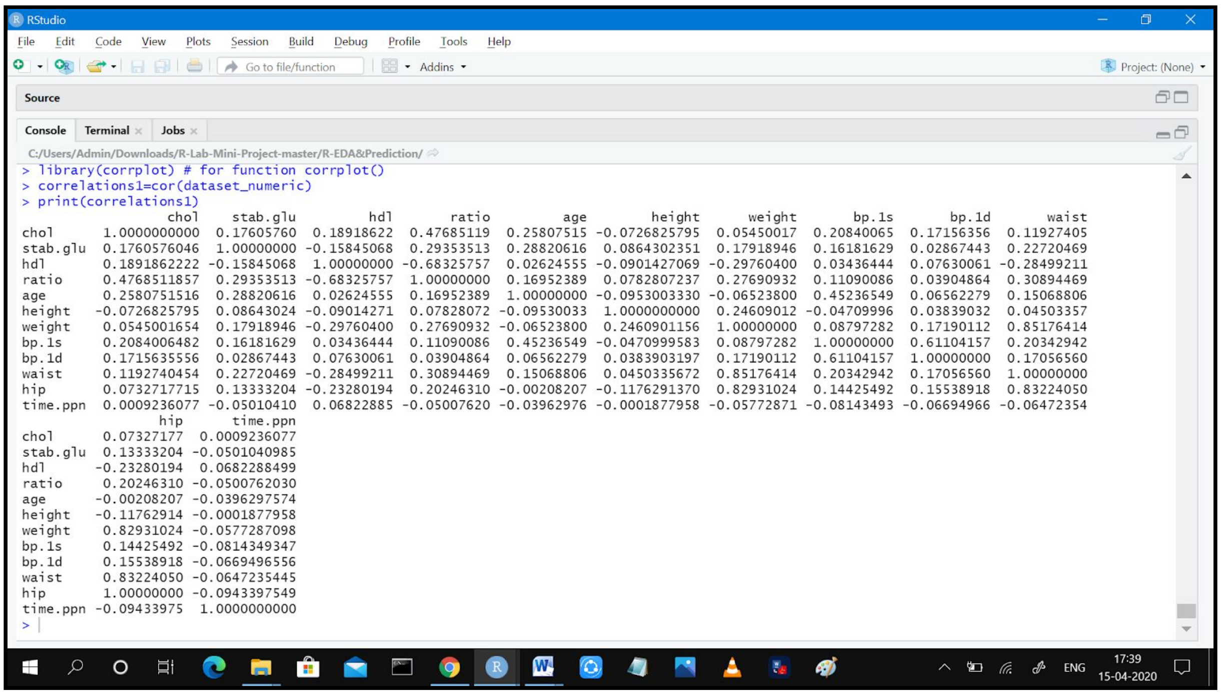
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/

```
## Copyright (C) 2005-2020 James Honaker, Gary King and Matthew
 Blackwell
## Refer to http://gking.harvard.edu/amelia/ for more informati
on
##
> par(mfrow=c(1,1))
> missmap(datasetRaw, col=c("red", "grey"), legend=FALSE)
>
```

counts	'table' int [1:3(1d)] 98 172 1…
filename	"C:\\Users\\Admin\\Downloads\\…
i	3L
name	"frame"

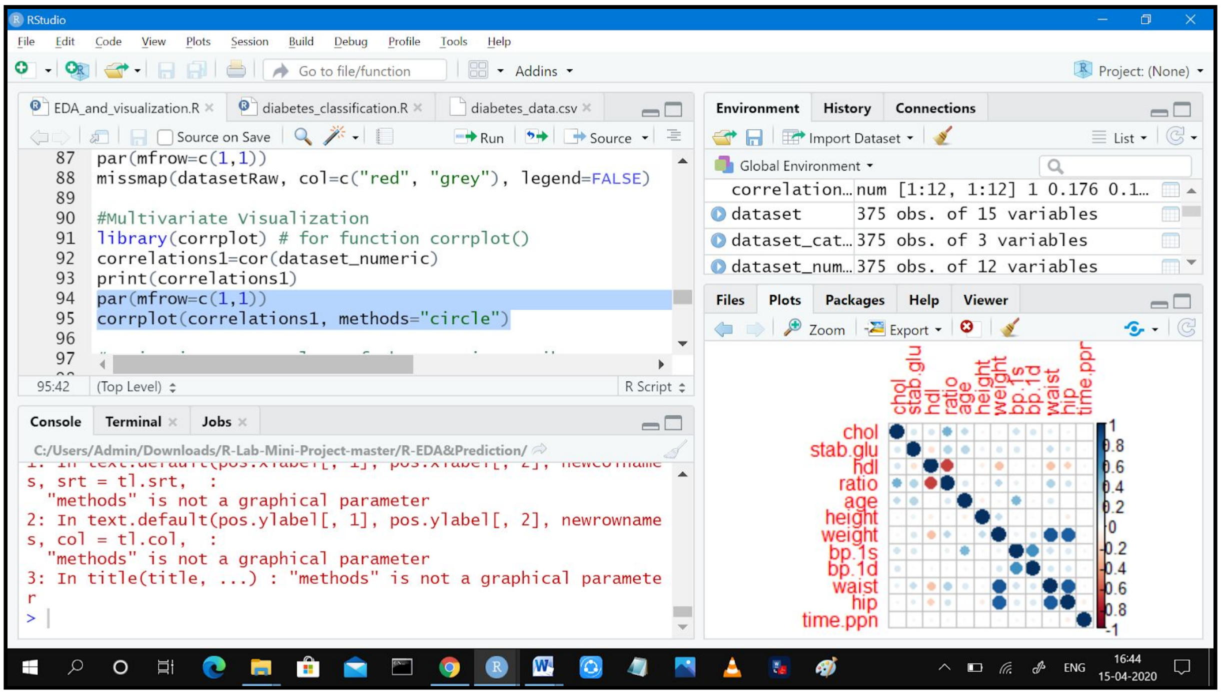
6:Multivariate Visualization

```
> library(corrplot) # for function corrplot()
Corrplot 0.84 loaded
> correlations1=cor (dataset_numeric)
> print (correlations1)
```



```
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
> library(corrplot) # for function corrplot()
> correlations1=cor(dataset_numeric)
> print(correlations1)
                 chol    stab.glu         hdl       ratio         age        height      weight       bp.1s       bp.1d       waist
chol     1.0000000000  0.17605760  0.18918622  0.47685119  0.25807515 -0.0726825795  0.05450017  0.20840065  0.17156356  0.11927405
stab.glu 0.1760576046  1.00000000 -0.15845068  0.29353513  0.28820616  0.0864302351  0.17918946  0.16181629  0.02867443  0.22720469
hdl      0.1891862222 -0.15845068  1.00000000 -0.68325757  0.02624555 -0.0901427069 -0.29760400  0.03436444  0.07630061 -0.28499211
ratio    0.4768511857  0.29353513 -0.68325757  1.00000000  0.16952389  0.0782807237  0.27690932  0.11090086  0.03904864  0.30894469
age      0.2580751516  0.28820616  0.02624555  0.16952389  1.00000000 -0.0953003330 -0.06523800  0.45236549  0.06562279  0.15068806
height  -0.0726825795  0.08643024 -0.09014271  0.07828072 -0.09530033  1.0000000000  0.24609012 -0.04709996  0.03839032  0.04503357
weight   0.0545001654  0.17918946 -0.29760400  0.27690932 -0.06523800  0.2460901156  1.00000000  0.08797282  0.17190112  0.85176414
bp.1s    0.2084006482  0.16181629  0.03436444  0.11090086  0.45236549 -0.0470999583  0.08797282  1.00000000  0.61104157  0.20342942
bp.1d    0.1715635556  0.02867443  0.07630061  0.03904864  0.06562279  0.0383903197  0.17190112  0.61104157  1.00000000  0.17056560
waist    0.1192740454  0.22720469 -0.28499211  0.30894469  0.15068806  0.0450335672  0.85176414  0.20342942  0.17056560  1.00000000
hip      0.0732717715  0.13333204 -0.23280194  0.20246310 -0.00208207 -0.1176291370  0.82931024  0.14425492  0.15538918  0.83224050
time.ppn 0.0009236077 -0.05010410  0.06822885 -0.05007620 -0.03962976 -0.0001877958 -0.05772871 -0.08143493 -0.06694966 -0.06472354
                hip      time.ppn
chol      0.07327177  0.0009236077
stab.glu  0.13333204 -0.0501040985
hdl      -0.23280194  0.0682288499
ratio     0.20246310 -0.0500762030
age      -0.00208207 -0.0396297574
height   -0.11762914 -0.0001877958
weight    0.82931024 -0.0577287098
bp.1s     0.14425492 -0.0814349347
bp.1d     0.15538918 -0.0669496556
waist     0.83224050 -0.0647235445
hip       1.00000000 -0.0943397549
time.ppn -0.09433975  1.0000000000
>
```

```
> par(mfrow=c(1,1))
> corrplot(correlations1, methods="circle")
```



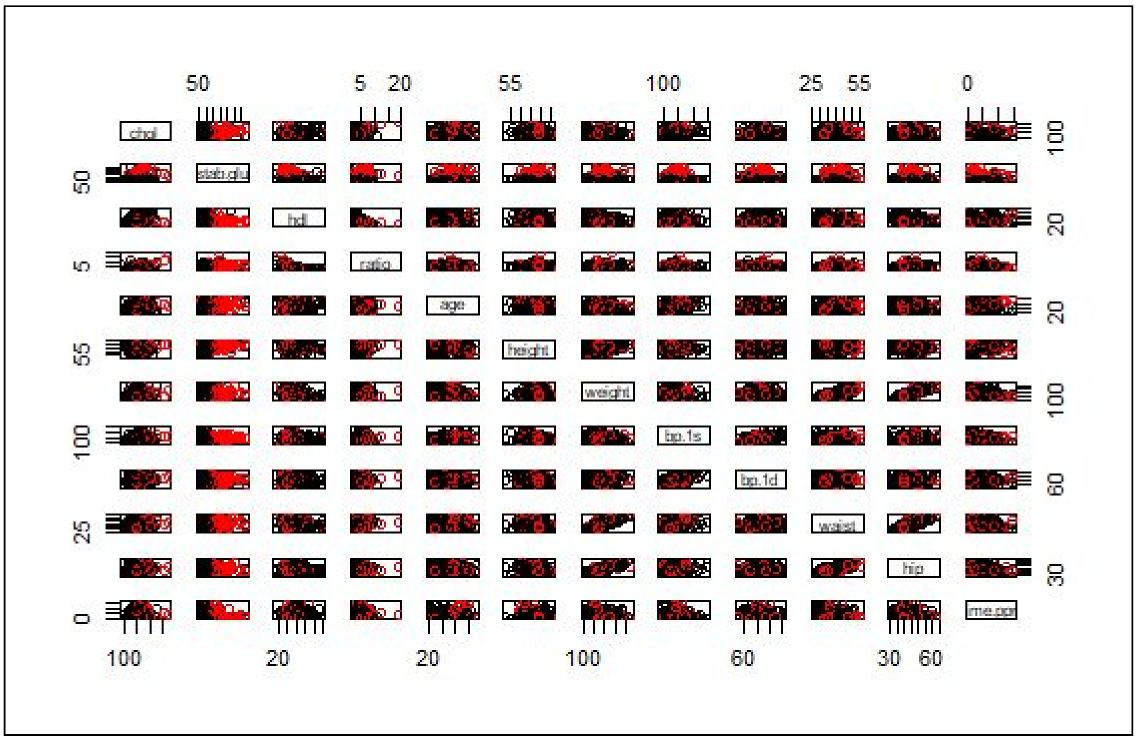
```
par(mfrow=c(1,1))
missmap(datasetRaw, col=c("red", "grey"), legend=FALSE)

#Multivariate Visualization
library(corrplot) # for function corrplot()
correlations1=cor(dataset_numeric)
print(correlations1)
par(mfrow=c(1,1))
corrplot(correlations1, methods="circle")
```

```
correlation… num [1:12, 1:12] 1 0.176 0.1…
dataset       375 obs. of 15 variables
dataset_cat…  375 obs. of 3 variables
dataset_num…  375 obs. of 12 variables
```

```
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
s, srt = tl.srt,  :
  "methods" is not a graphical parameter
2: In text.default(pos.ylabel[, 1], pos.ylabel[, 2], newrowname
s, col = tl.col,  :
  "methods" is not a graphical parameter
3: In title(title, ...) : "methods" is not a graphical paramete
r
>
```

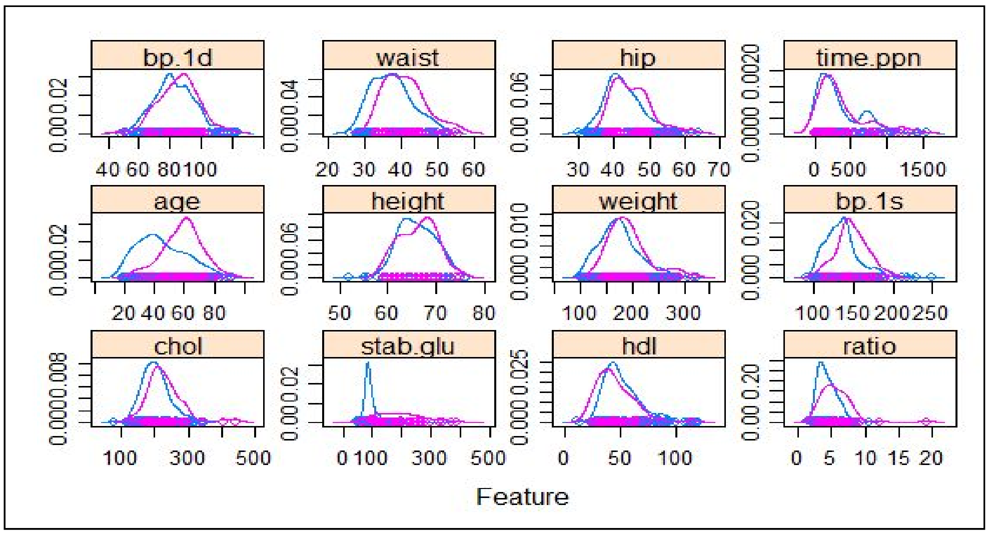
7:Pairwise scatterplots of the numeric attributes

```
> par(mfrow=c(1,1))
pairs (dataset_numeric)
> #Scatterplot Matrix By Class (use different color to distinguish different class)
> par(mfrow=c(1,1))
> pairs (dataset_numeric, col=dataset[,5])
```



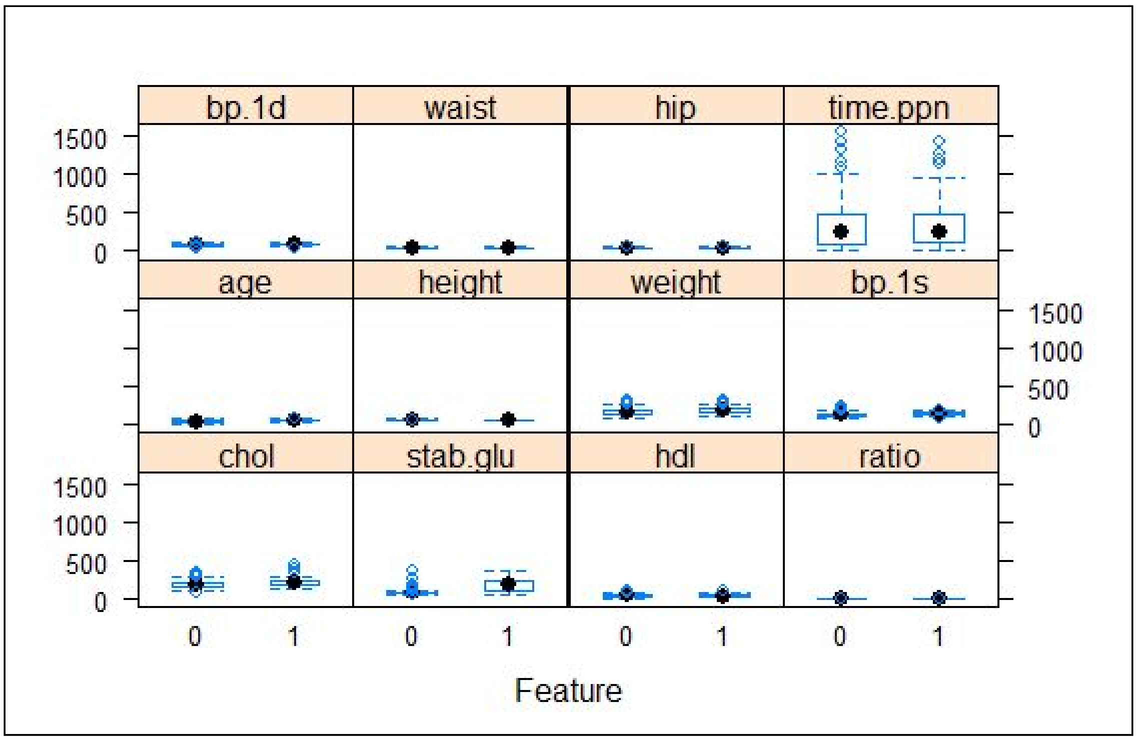
8:Density By Class

```
> library(caret)
> # load the data
> data(iris)
> # density plots for each attribute by class value
> x <- dataset_numeric
> y <- dataset[,5]
> scales <- list(x=list(relation="free"), y=list(relation="free"))
> par(mfrow=c(1,1))
> feature Plot(x=dataset_numeric, y=dataset[,5], plot="density", scales=scales)
```

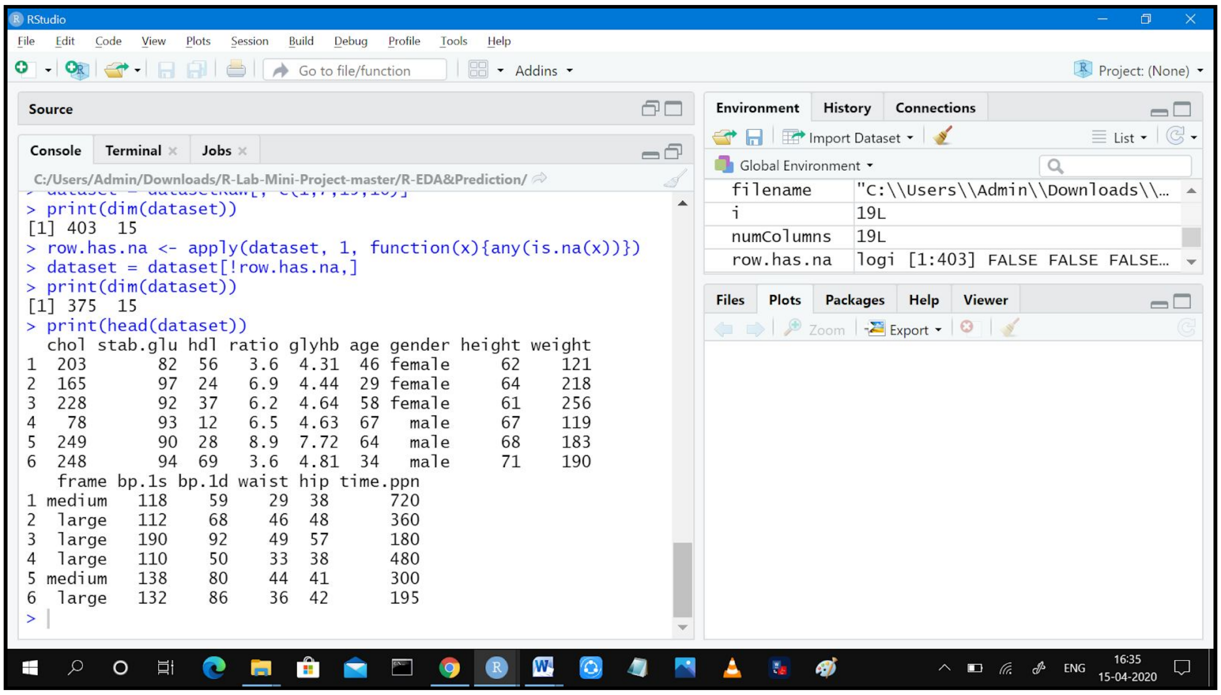


9:Box And Whisker Plots By Class

```
> featurePlot(x=dataset_numeric, y=dataset[,5], plot="box")
```



```
# Load libraries
> library(randomForest)
> library(caret)

> # load the data
> filename="C:\Users\Admin \Desktop\\\ R-Lab-Mini-Project \diabetes_data.csv"
> datasetRaw = read.csv(filename)
> print(head(datasetRaw))
```



```
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
> print(dim(dataset))
[1] 403  15
> row.has.na <- apply(dataset, 1, function(x){any(is.na(x))})
> dataset = dataset[!row.has.na,]
> print(dim(dataset))
[1] 375  15
> print(head(dataset))
  chol stab.glu hdl ratio glyhb age gender height weight
1  203       82  56   3.6  4.31  46 female     62    121
2  165       97  24   6.9  4.44  29 female     64    218
3  228       92  37   6.2  4.64  58 female     61    256
4   78       93  12   6.5  4.63  67   male     67    119
5  249       90  28   8.9  7.72  64   male     68    183
6  248       94  69   3.6  4.81  34   male     71    190
   frame bp.1s bp.1d waist hip time.ppn
1 medium   118    59    29  38      720
2  large   112    68    46  48      360
3  large   190    92    49  57      180
4  large   110    50    33  38      480
5 medium   138    80    44  41      300
6  large   132    86    36  42      195
> 
```

```
filename      "C:\\Users\\Admin\\Downloads\\...
i             19L
numColumns    19L
row.has.na    logi [1:403] FALSE FALSE FALSE...
```

```
# clean the data
> numColumns = dim(datasetRaw)[2]
> vector_NAs = rep (0, numColumns)
> for (i in 1:numColumns) {
+   vector_NAs[i] = sum (is.na (datasetRaw [,i]))
+
+ }
> print ("The missing values in each column :")
```

```
[1] "The missing values in each column:"
```

```
> print (vector_NAs)
```

```
 [1]   0   1   0   1   1  13   0   0   0   5   1   0   5   5 262 262   2   2   3
```

```
# delete columns 15 and 16 due to many missing values
> # delete column 1 (id), column 7 (location) because they contain no useful information
> dataset = datasetRaw[,-c(1,7,15,16)]
> print(dim(dataset))
```

```
[1] 403  15
```

```
# remove the row with missing values
> row.has.na <- apply(dataset, 1, function(x){any(is.na(x))})
> dataset = dataset[!row.has.na,]
> print(dim(dataset))
```

```
[1] 375  15
```

```
> print(head(dataset))
```

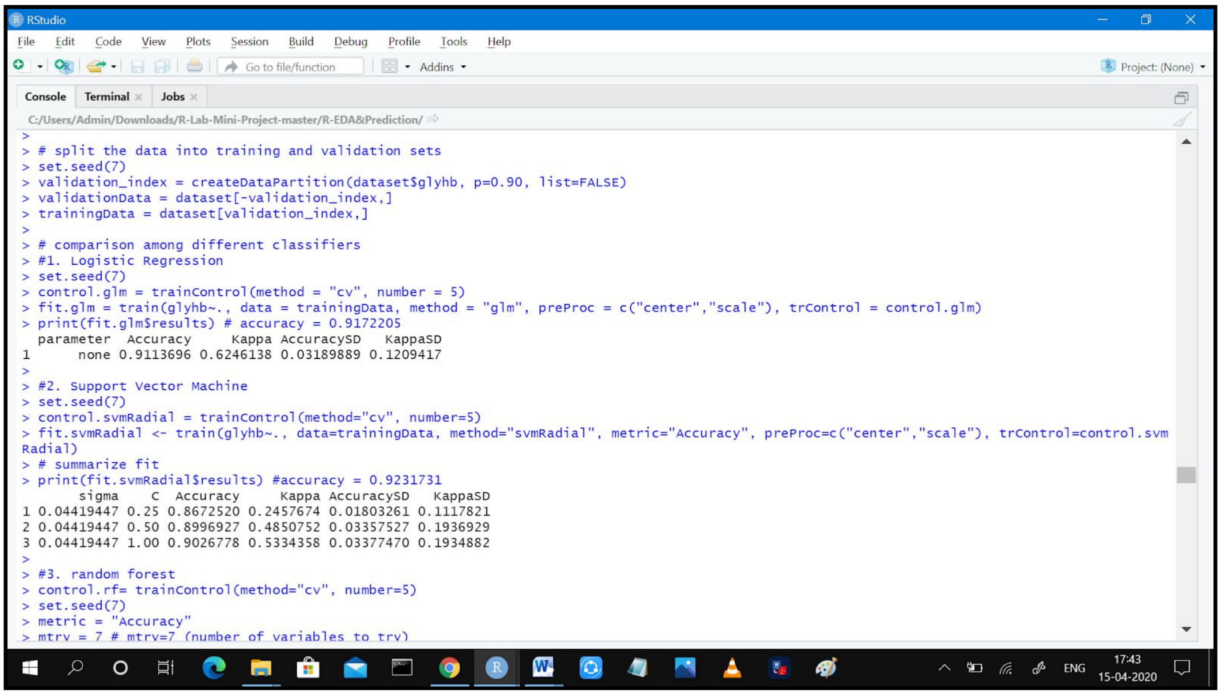
```
Id chol stab.glu hdl ratio glyhb age gender height weight  frame bp.1s bp.1d waist hip
1   203   82      56   3.6  4.31   46  female    62    121      medium   118    59    29  38
2   165    97     24   6.9 4.44   29  female    64    218      large   112    68    46  48
3   228    92     37   6.2  4.64   58  female    61    256      large   190    92    49  57
4   78     93     12   6.5  4.63   67   male      67    119     large   110    50    33  38
5    249   90     28   8.9  7.72   6 4   male     68    183      medium   138    80    44  41
6    248   94     69   3.6  4.81   34   male     71    190       large   132    86    36  42
```

```
# encode the class label (column 5): Glycosylated hemoglobin  > 7.0 is taken as a positive diagnosis of diabetes.
> dataset[,5] = ifelse(dataset[,5] >= 7.0, 1, 0)
> dataset[,5] = factor(dataset[,5])

> # encode the categorical data (column-7 gender)
> dataset[,7] = ifelse(dataset[,7] == "female", 0, 1)
> dataset[,7] = factor(dataset[,7])

> # encode the categorical data (column-10 frame)
> dataset[,10] = ifelse(dataset[,10] == "small", 0, ifelse(dataset[,10] == "medium", 1,2) )
> dataset[,10] = factor(dataset[,10])

> # split the data into training and validation sets
> set.seed(7)
> validation_index = createDataPartition(dataset$glyhb, p=0.90, list=FALSE)
> validationData = dataset[-validation_index,]
> trainingData = dataset[validation_index,]
```

<u>PART 3:DATA CLASSIFICATION</u>

#1. Logistic Regression

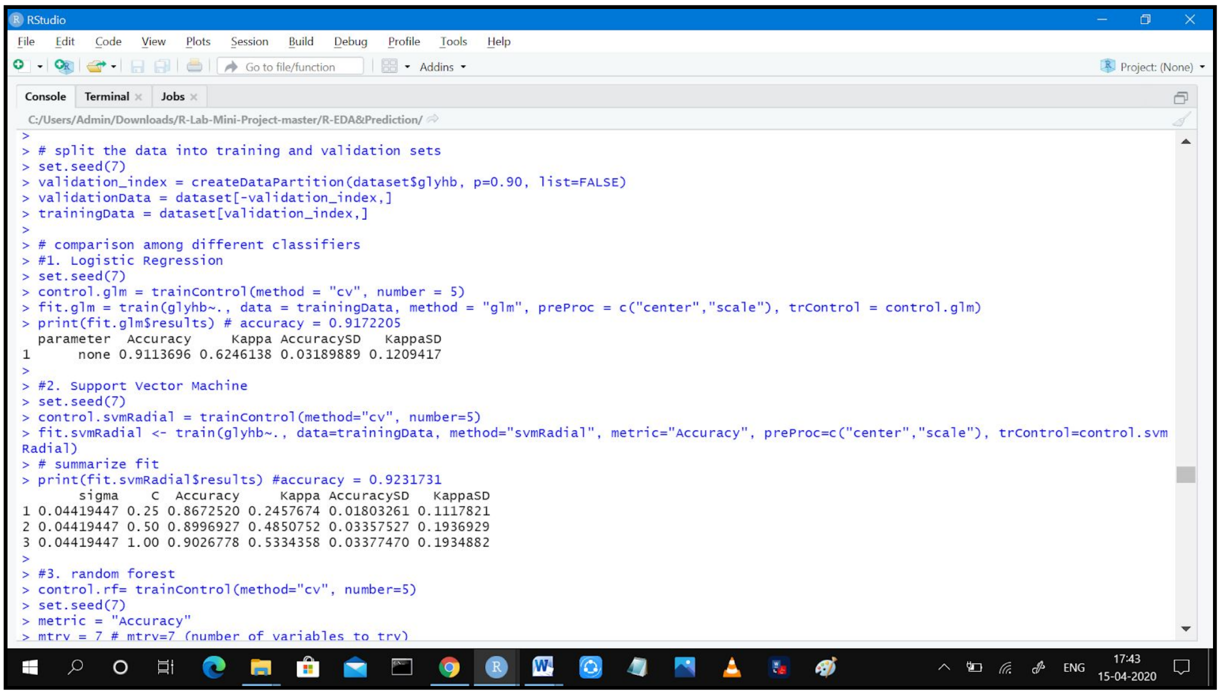
```
> set.seed(7)
> control.glm = trainControl(method = "cv", number = 5)
> fit.glm = train(glyhb~., data = trainingData, method = "glm", preProc = c("center","scale"),
trControl = control.glm)
> print(fit.glm$results)
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console Terminal Jobs
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
ENG
17:43
15-04-2020

```
>
> # split the data into training and validation sets
> set.seed(7)
> validation_index = createDataPartition(dataset$glyhb, p=0.90, list=FALSE)
> validationData = dataset[-validation_index,]
> trainingData = dataset[validation_index,]
>
> # comparison among different classifiers
> #1. Logistic Regression
> set.seed(7)
> control.glm = trainControl(method = "cv", number = 5)
> fit.glm = train(glyhb~., data = trainingData, method = "glm", preProc = c("center","scale"), trControl = control.glm)
> print(fit.glm$results) # accuracy = 0.9172205
  parameter  Accuracy     Kappa AccuracySD   KappaSD
1      none 0.9113696 0.6246138 0.03189889 0.1209417
>
> #2. Support Vector Machine
> set.seed(7)
> control.svmRadial = trainControl(method="cv", number=5)
> fit.svmRadial <- train(glyhb~., data=trainingData, method="svmRadial", metric="Accuracy", preProc=c("center","scale"), trControl=control.svm
Radial)
> # summarize fit
> print(fit.svmRadial$results) #accuracy = 0.9231731
       sigma    C  Accuracy     Kappa AccuracySD   KappaSD
1 0.04419447 0.25 0.8672520 0.2457674 0.01803261 0.1117821
2 0.04419447 0.50 0.8996927 0.4850752 0.03357527 0.1936929
3 0.04419447 1.00 0.9026778 0.5334358 0.03377470 0.1934882
>
> #3. random forest
> control.rf= trainControl(method="cv", number=5)
> set.seed(7)
> metric = "Accuracy"
> mtry = 7 # mtry=7 (number of variables to try)
```

2. <u>Support Vector Machine</u>

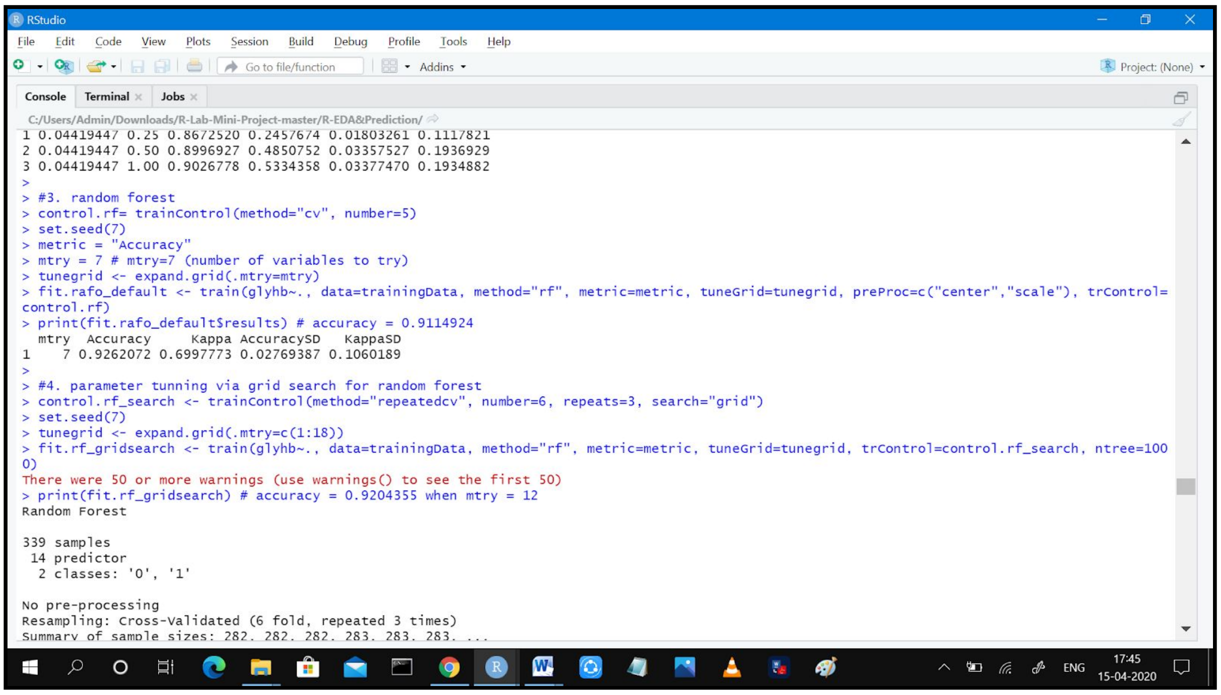
```
> set.seed(7)
> control.svmRadial = trainControl(method="cv", number=5)
> fit.svmRadial <- train(glyhb~., data=trainingData, method="svmRadial",
metric="Accuracy", preProc=c("center","scale"), trControl=control.svmRadial)
> # summarize fit
> print(fit.svmRadial$results)
```



```
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
>
> # split the data into training and validation sets
> set.seed(7)
> validation_index = createDataPartition(dataset$glyhb, p=0.90, list=FALSE)
> validationData = dataset[-validation_index,]
> trainingData = dataset[validation_index,]
>
> # comparison among different classifiers
> #1. Logistic Regression
> set.seed(7)
> control.glm = trainControl(method = "cv", number = 5)
> fit.glm = train(glyhb~., data = trainingData, method = "glm", preProc = c("center","scale"), trControl = control.glm)
> print(fit.glm$results) # accuracy = 0.9172205
  parameter  Accuracy     Kappa AccuracySD   KappaSD
1      none 0.9113696 0.6246138 0.03189889 0.1209417
>
> #2. Support Vector Machine
> set.seed(7)
> control.svmRadial = trainControl(method="cv", number=5)
> fit.svmRadial <- train(glyhb~., data=trainingData, method="svmRadial", metric="Accuracy", preProc=c("center","scale"), trControl=control.svm
Radial)
> # summarize fit
> print(fit.svmRadial$results) #accuracy = 0.9231731
       sigma    C  Accuracy     Kappa AccuracySD   KappaSD
1 0.04419447 0.25 0.8672520 0.2457674 0.01803261 0.1117821
2 0.04419447 0.50 0.8996927 0.4850752 0.03357527 0.1936929
3 0.04419447 1.00 0.9026778 0.5334358 0.03377470 0.1934882
>
> #3. random forest
> control.rf= trainControl(method="cv", number=5)
> set.seed(7)
> metric = "Accuracy"
> mtry = 7 # mtry=7 (number of variables to try)
```

3. Random forest

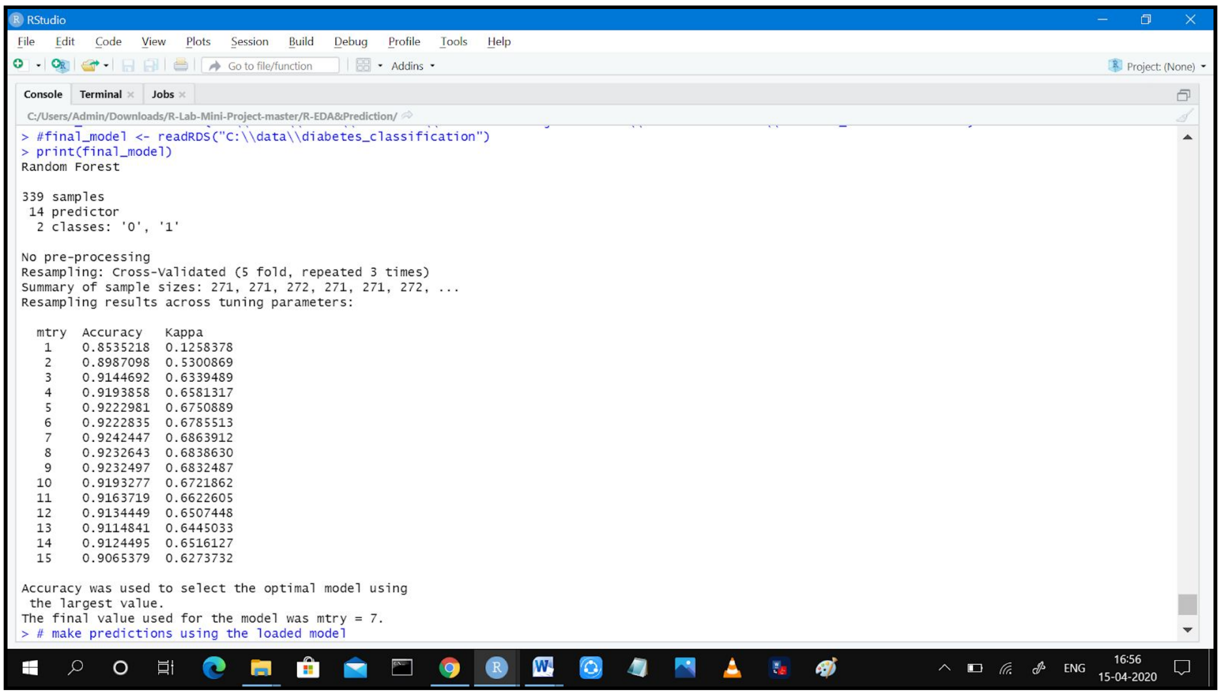
> control.rf = trainControl(method="cv", number=5)
> set.seed(7)
> metric = "Accuracy"
> mtry = 7 # mtry=7 (number of variables to try)
> tunegrid <- expand.grid(.mtry=mtry)
> fit.rf_default <- train(glyhb~., data=trainingData, method="rf", metric=metric, tuneGrid=tunegrid, preProc=c("center","scale"), trControl=control)
> print(fit.rf_default$results)



```
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
1 0.04419447 0.25 0.8672520 0.2457674 0.01803261 0.1117821
2 0.04419447 0.50 0.8996927 0.4850752 0.03357527 0.1936929
3 0.04419447 1.00 0.9026778 0.5334358 0.03377470 0.1934882
>
> #3. random forest
> control.rf= trainControl(method="cv", number=5)
> set.seed(7)
> metric = "Accuracy"
> mtry = 7 # mtry=7 (number of variables to try)
> tunegrid <- expand.grid(.mtry=mtry)
> fit.rafo_default <- train(glyhb~., data=trainingData, method="rf", metric=metric, tuneGrid=tunegrid, preProc=c("center","scale"), trControl=
control.rf)
> print(fit.rafo_default$results) # accuracy = 0.9114924
  mtry  Accuracy     Kappa AccuracySD   KappaSD
1    7 0.9262072 0.6997773 0.02769387 0.1060189
>
> #4. parameter tunning via grid search for random forest
> control.rf_search <- trainControl(method="repeatedcv", number=6, repeats=3, search="grid")
> set.seed(7)
> tunegrid <- expand.grid(.mtry=c(1:18))
> fit.rf_gridsearch <- train(glyhb~., data=trainingData, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control.rf_search, ntree=100
0)
There were 50 or more warnings (use warnings() to see the first 50)
> print(fit.rf_gridsearch) # accuracy = 0.9204355 when mtry = 12
Random Forest

339 samples
 14 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (6 fold, repeated 3 times)
Summary of sample sizes: 282, 282, 282, 283, 283, 283, ...
```

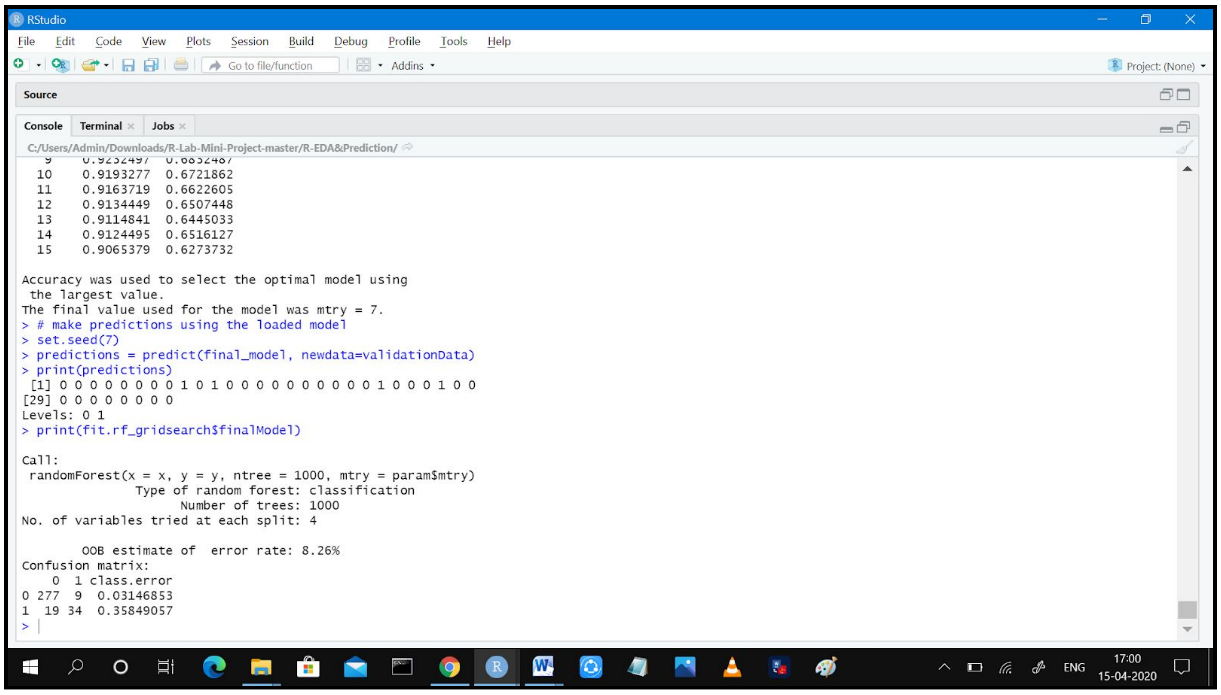
> #4. Parameter tuning via grid search for random forest

```
> control.rf_search <- trainControl(method="repeatedcv", number=5,
repeats=3, search="grid")
> set.seed(7)
> tunegrid <- expand.grid(.mtry=c(1:15))
> fit.rf_gridsearch <- train(glyhb~., data=trainingData, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=control.rf_search, ntree=1000)
> print(fit.rf_gridsearch) # accuracy = 0.9204355 when mtry = 12
```



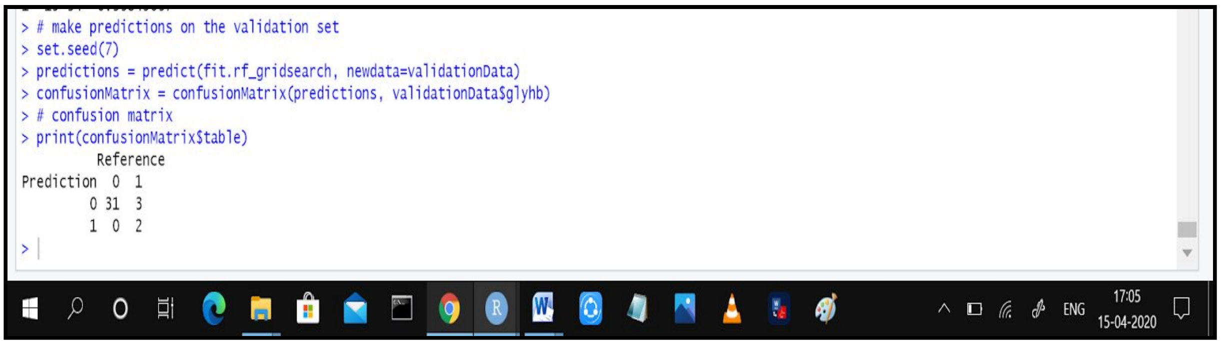
```
> #final_model <- readRDS("C:\\data\\diabetes_classification")
> print(final_model)
Random Forest

339 samples
 14 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 271, 271, 272, 271, 271, 272, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   1    0.8535218  0.1258378
   2    0.8987098  0.5300869
   3    0.9144692  0.6339489
   4    0.9193858  0.6581317
   5    0.9222981  0.6750889
   6    0.9222835  0.6785513
   7    0.9242447  0.6863912
   8    0.9232643  0.6838630
   9    0.9232497  0.6832487
  10    0.9193277  0.6721862
  11    0.9163719  0.6622605
  12    0.9134449  0.6507448
  13    0.9114841  0.6445033
  14    0.9124495  0.6516127
  15    0.9065379  0.6273732

Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was mtry = 7.
> # make predictions using the loaded model
```

```
> print(fit.rf_gridsearch$finalModel)
```

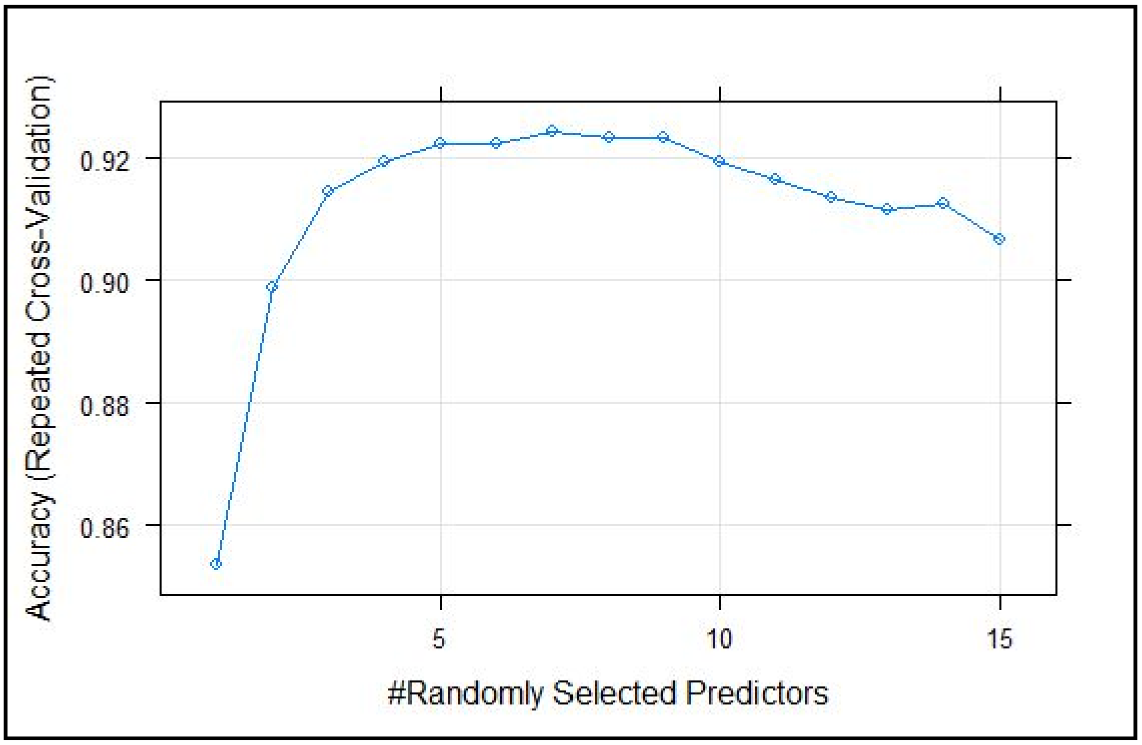


```
  10    0.9193277  0.6721862
  11    0.9163719  0.6622605
  12    0.9134449  0.6507448
  13    0.9114841  0.6445033
  14    0.9124495  0.6516127
  15    0.9065379  0.6273732

Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was mtry = 7.
> # make predictions using the loaded model
> set.seed(7)
> predictions = predict(final_model, newdata=validationData)
> print(predictions)
 [1] 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0
[29] 0 0 0 0 0 0 0 0
Levels: 0 1
> print(fit.rf_gridsearch$finalModel)

Call:
 randomForest(x = x, y = y, ntree = 1000, mtry = param$mtry)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 8.26%
Confusion matrix:
    0  1 class.error
0 277  9  0.03146853
1  19 34  0.35849057
>
```

```
> plot(fit.rf_gridsearch)
> # make predictions on the validation set
> set.seed(7)
> predictions = predict(fit.rf_gridsearch, newdata=validationData)
> confusionMatrix = confusionMatrix(predictions, validationData$glyhb)
> # confusion matrix
> print(confusionMatrix$table)
```

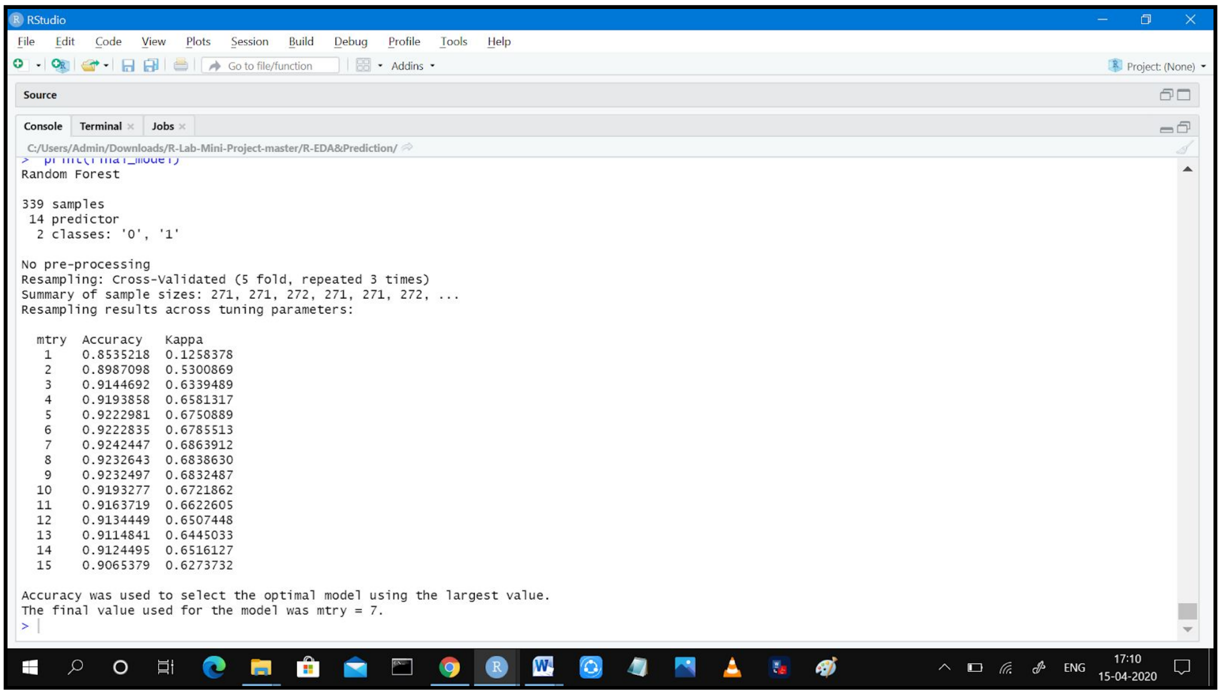


```
> # make predictions on the validation set
> set.seed(7)
> predictions = predict(fit.rf_gridsearch, newdata=validationData)
> confusionMatrix = confusionMatrix(predictions, validationData$glyhb)
> # confusion matrix
> print(confusionMatrix$table)
          Reference
Prediction  0  1
         0 31  3
         1  0  2
> 
```



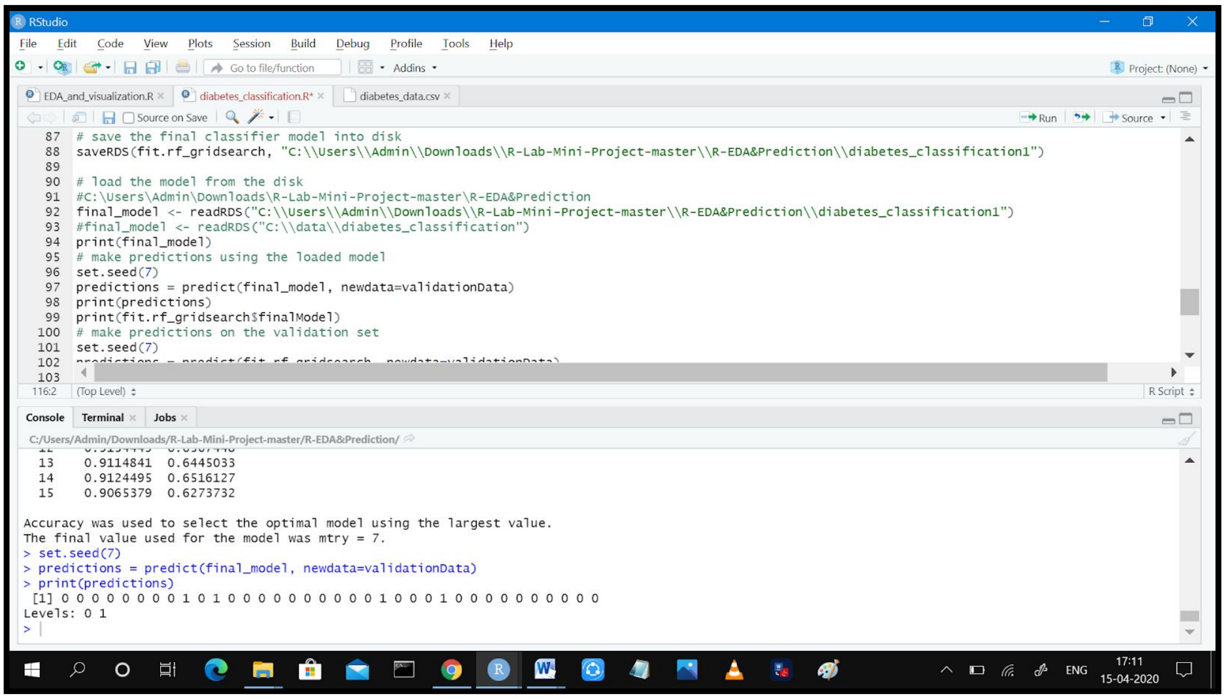
PART 4:Prediction of Accuracy

```
#save the final classifier model into disk
> saveRDS(fit.rf_gridsearch, "C:\\Users\\Admin\\Desktop\\
R-Lab-Mini-Project\\diabetes_classification1")

> # load the model from the disk
> final_model <- readRDS("C:\\Users\\Admin\\Desktop\\\\ R-Lab-Mini-Project
\\diabetes_classification")
> #final_model <- readRDS("C:\\data\\diabetes_classification")
> print(final_model)
```



```
C:/Users/Admin/Downloads/R-Lab-Mini-Project-master/R-EDA&Prediction/
> print(final_model)
Random Forest

339 samples
 14 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 271, 271, 272, 271, 271, 272, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   1    0.8535218  0.1258378
   2    0.8987098  0.5300869
   3    0.9144692  0.6339489
   4    0.9193858  0.6581317
   5    0.9222981  0.6750889
   6    0.9222835  0.6785513
   7    0.9242447  0.6863912
   8    0.9232643  0.6838630
   9    0.9232497  0.6832487
  10    0.9193277  0.6721862
  11    0.9163719  0.6622605
  12    0.9134449  0.6507448
  13    0.9114841  0.6445033
  14    0.9124495  0.6516127
  15    0.9065379  0.6273732

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 7.
>
```

```
# make predictions using the loaded model
> set.seed(7)
> predictions = predict(final_model, newdata=validationData)
> print(predictions)
```



```
# save the final classifier model into disk
saveRDS(fit.rf_gridsearch, "C:\\Users\\Admin\\Downloads\\R-Lab-Mini-Project-master\\R-EDA&Prediction\\diabetes_classification1")

# load the model from the disk
#C:\Users\Admin\Downloads\R-Lab-Mini-Project-master\R-EDA&Prediction
final_model <- readRDS("C:\\Users\\Admin\\Downloads\\R-Lab-Mini-Project-master\\R-EDA&Prediction\\diabetes_classification1")
#final_model <- readRDS("C:\\data\\diabetes_classification")
print(final_model)
# make predictions using the loaded model
set.seed(7)
predictions = predict(final_model, newdata=validationData)
print(predictions)
print(fit.rf_gridsearch$finalModel)
# make predictions on the validation set
set.seed(7)
```

```
  13    0.9114841  0.6445033
  14    0.9124495  0.6516127
  15    0.9065379  0.6273732

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 7.
> set.seed(7)
> predictions = predict(final_model, newdata=validationData)
> print(predictions)
 [1] 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
Levels: 0 1
> 
```