project

loading libraries

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
library(DataExplorer)
```

```
## Warning: package 'DataExplorer' was built under R version 3.6.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 3.6.3
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.6.3
```

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.6.3
```

```
library(summarytools)
```

```
## Warning: package 'summarytools' was built under R version 3.6.3
```

```
## Registered S3 method overwritten by 'pryr':
##   method      from
##   print.bytes Rcpp
```

```
## For best results, restart R session and update pander using devtools:: or remotes::install_github('ra
```

```r
library(DMwR)
```

```
## Warning: package 'DMwR' was built under R version 3.6.3

## Loading required package: grid

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.3

## Loading required package: tibble

## Warning: package 'tibble' was built under R version 3.6.3

##
## Attaching package: 'tibble'

## The following object is masked from 'package:summarytools':
##
##     view

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

##
## Attaching package: 'rattle'

## The following object is masked from 'package:xgboost':
##
##     xgboost
```

load dataset:

```r
Cars_data <- read_csv("C:/Users/daoud/Downloads/PGP DSBA/prodictive modeling/week 5 project/Cars-dataset
```

```
## Parsed with column specification:
## cols(
##   Age = col_double(),
##   Gender = col_character(),
```

```
##   Engineer = col_double(),
##   MBA = col_double(),
##   `Work Exp` = col_double(),
##   Salary = col_double(),
##   Distance = col_double(),
##   license = col_double(),
##   Transport = col_character()
## )
```

```
#View(Cars_data)
```

Exploratory Data Analysis

```
summarytools::view(dfSummary(Cars_data))
```

```
## Switching method to 'browser'
```

```
## Output file written: C:\Users\daoud\AppData\Local\Temp\RtmpCUAqn1\file3118d11037.html
```

Observation : 1- we have 418 employeer with 9 varible ,Gender and Transport are character , the other varible are numeric . 2- we have only one missing value : MBA . 3- column name "Work Exp" will change to "Work_Exp" . 4- 19.9% of employee use "2Wheeler", 8.4% use "Car", 71.8% use "Public Transport". 5- dependent varible is "Transport" , independent varible are :"Age","Gender","Engineer","MBA","Work Exp","Salary","Distance","license" ## challenging problem : we have 3 classes on a target variable , it should be 2 only. the task was to predict whether or not an employee will use Car as a mode of transport. there are two methods to solve the problem : "levels" or "ifelse", for today we will use "ifelse" to assign 1 for "Car" and 0 for others " Public_Transport , 2Wheeler " as Transport_car .

```
summary(Cars_data)
```

```
##       Age            Gender             Engineer           MBA        
##  Min.   :18.00   Length:418         Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:25.00   Class :character   1st Qu.:0.2500   1st Qu.:0.0000  
##  Median :27.00   Mode  :character   Median :1.0000   Median :0.0000  
##  Mean   :27.33                      Mean   :0.7488   Mean   :0.2614  
##  3rd Qu.:29.00                      3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :43.00                      Max.   :1.0000   Max.   :1.0000  
##                                                      NA's   :1       
##     Work Exp          Salary          Distance        license      
##  Min.   : 0.000   Min.   : 6.500   Min.   : 3.20   Min.   :0.0000  
##  1st Qu.: 3.000   1st Qu.: 9.625   1st Qu.: 8.60   1st Qu.:0.0000  
##  Median : 5.000   Median :13.000   Median :10.90   Median :0.0000  
##  Mean   : 5.873   Mean   :15.418   Mean   :11.29   Mean   :0.2033  
##  3rd Qu.: 8.000   3rd Qu.:14.900   3rd Qu.:13.57   3rd Qu.:0.0000  
##  Max.   :24.000   Max.   :57.000   Max.   :23.40   Max.   :1.0000  
##                                                                    
##   Transport        
##  Length:418        
##  Class :character  
##  Mode  :character  
```

```
##
##
##
##
```

```r
str(Cars_data)
```

```
## tibble [418 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Age      : num [1:418] 28 24 27 25 25 21 23 23 24 28 ...
##  $ Gender   : chr [1:418] "Male" "Male" "Female" "Male" ...
##  $ Engineer : num [1:418] 1 1 1 0 0 0 1 0 1 1 ...
##  $ MBA      : num [1:418] 0 0 0 0 0 0 1 0 0 0 ...
##  $ Work Exp : num [1:418] 5 6 9 1 3 3 3 0 4 6 ...
##  $ Salary   : num [1:418] 14.4 10.6 15.5 7.6 9.6 9.5 11.7 6.5 8.5 13.7 ...
##  $ Distance : num [1:418] 5.1 6.1 6.1 6.3 6.7 7.1 7.2 7.3 7.5 7.5 ...
##  $ license  : num [1:418] 0 0 0 0 0 0 0 0 0 1 ...
##  $ Transport: chr [1:418] "2Wheeler" "2Wheeler" "2Wheeler" "2Wheeler" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Age = col_double(),
##   ..   Gender = col_character(),
##   ..   Engineer = col_double(),
##   ..   MBA = col_double(),
##   ..   `Work Exp` = col_double(),
##   ..   Salary = col_double(),
##   ..   Distance = col_double(),
##   ..   license = col_double(),
##   ..   Transport = col_character()
##   .. )
```

```r
Cars_data <-na.omit(Cars_data) # drop missing value
names(Cars_data)[names(Cars_data)=="Work Exp"]<-"Work_Exp" # change name without space .
Cars_data$Transport_car <- ifelse(Cars_data$Transport=="Car",1,0) # convert 'Car' to 1 else to 0
Cars_data$Gender <- ifelse(Cars_data$Gender == "Male",1,0) # convert 'Male' to 1 else to 0
prop.table(table(Cars_data$Transport))
```

```
##
##         2Wheeler              Car Public Transport
##       0.19904077       0.08393285       0.71702638
```

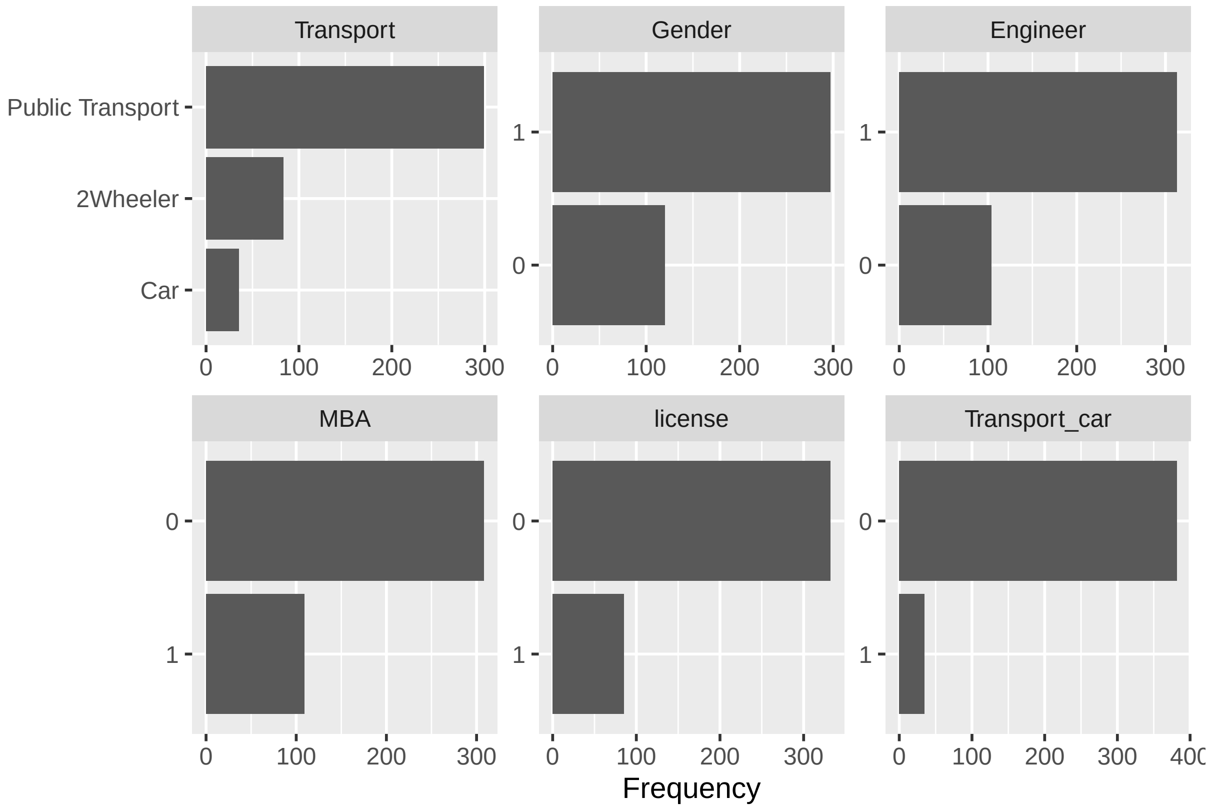
```r
Cars_data
```

```
## # A tibble: 417 x 10
##      Age Gender Engineer   MBA Work_Exp Salary Distance license Transport
##    <dbl>  <dbl>    <dbl> <dbl>    <dbl>  <dbl>    <dbl>   <dbl> <chr>
##  1    28      1        1     0        5   14.4      5.1       0 2Wheeler
##  2    24      1        1     0        6   10.6      6.1       0 2Wheeler
##  3    27      0        1     0        9   15.5      6.1       0 2Wheeler
##  4    25      1        0     0        1    7.6      6.3       0 2Wheeler
##  5    25      0        0     0        3    9.6      6.7       0 2Wheeler
##  6    21      1        0     0        3    9.5      7.1       0 2Wheeler
##  7    23      1        1     1        3   11.7      7.2       0 2Wheeler
```

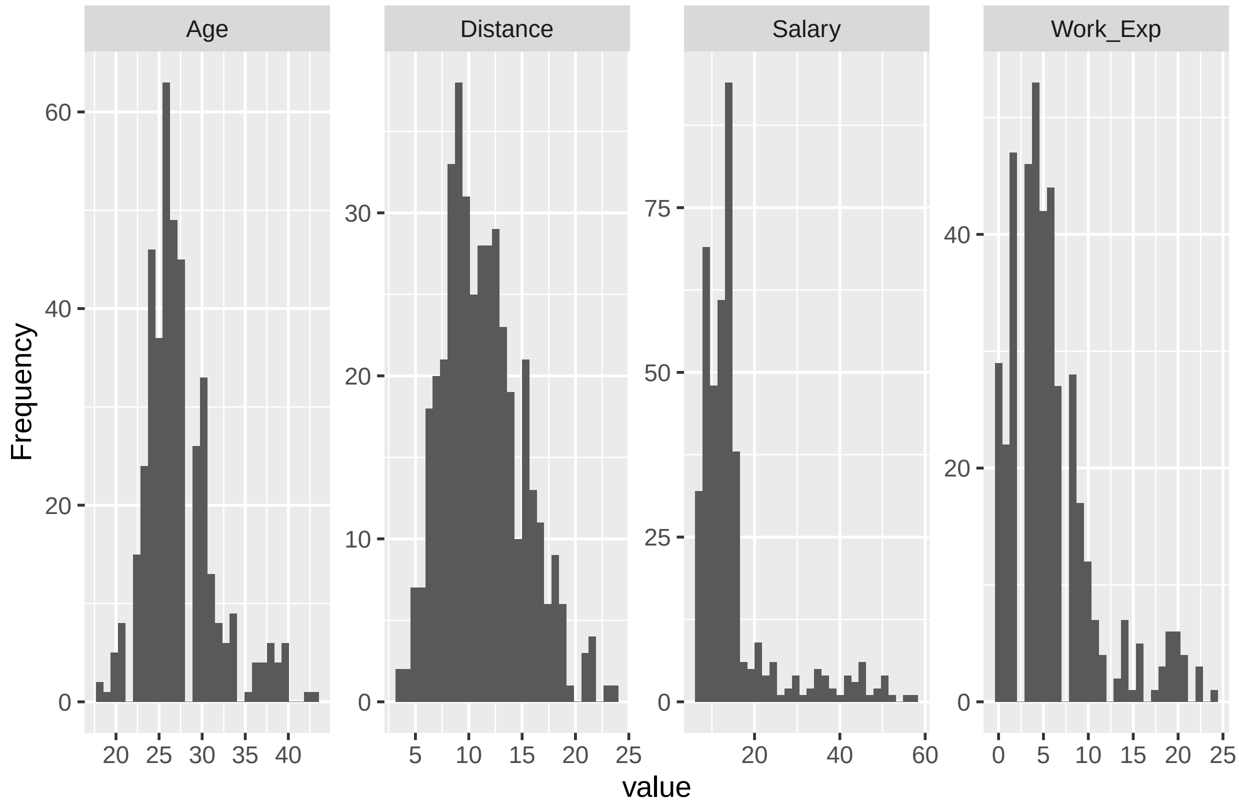
```
##  8     23       1         0      0         0     6.5       7.3        0 2Wheeler
##  9     24       1         1      0         4     8.5       7.5        0 2Wheeler
## 10     28       1         1      0         6    13.7       7.5        1 2Wheeler
## # ... with 407 more rows, and 1 more variable: Transport_car <dbl>
```

Observation : 1- ther is one missing value and we drop it. 2- we have 19.9% use 2Wheeler and 8.39% use car and 71.7% use Public Transport. ## normality distribution : # visualization:

```
plot_bar(Cars_data)
```

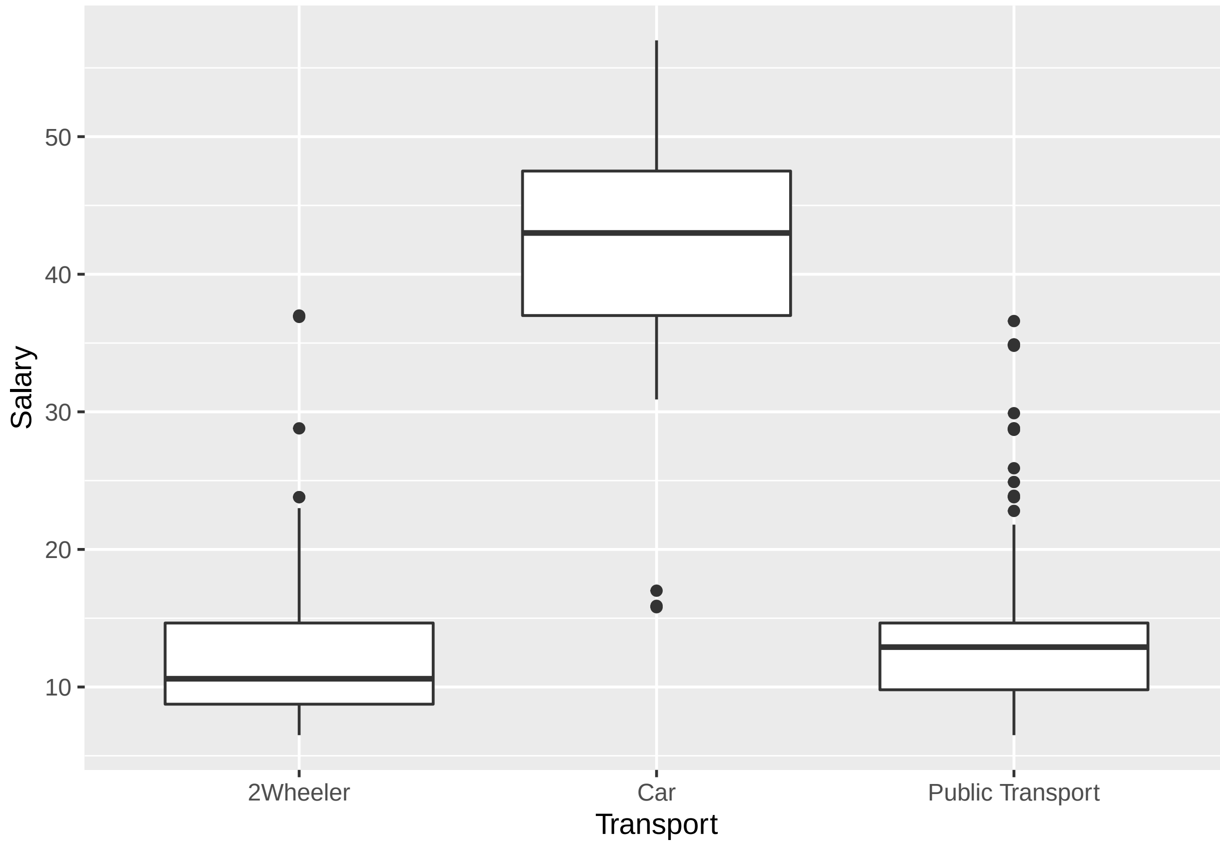



```
plot_histogram(Cars_data)
```



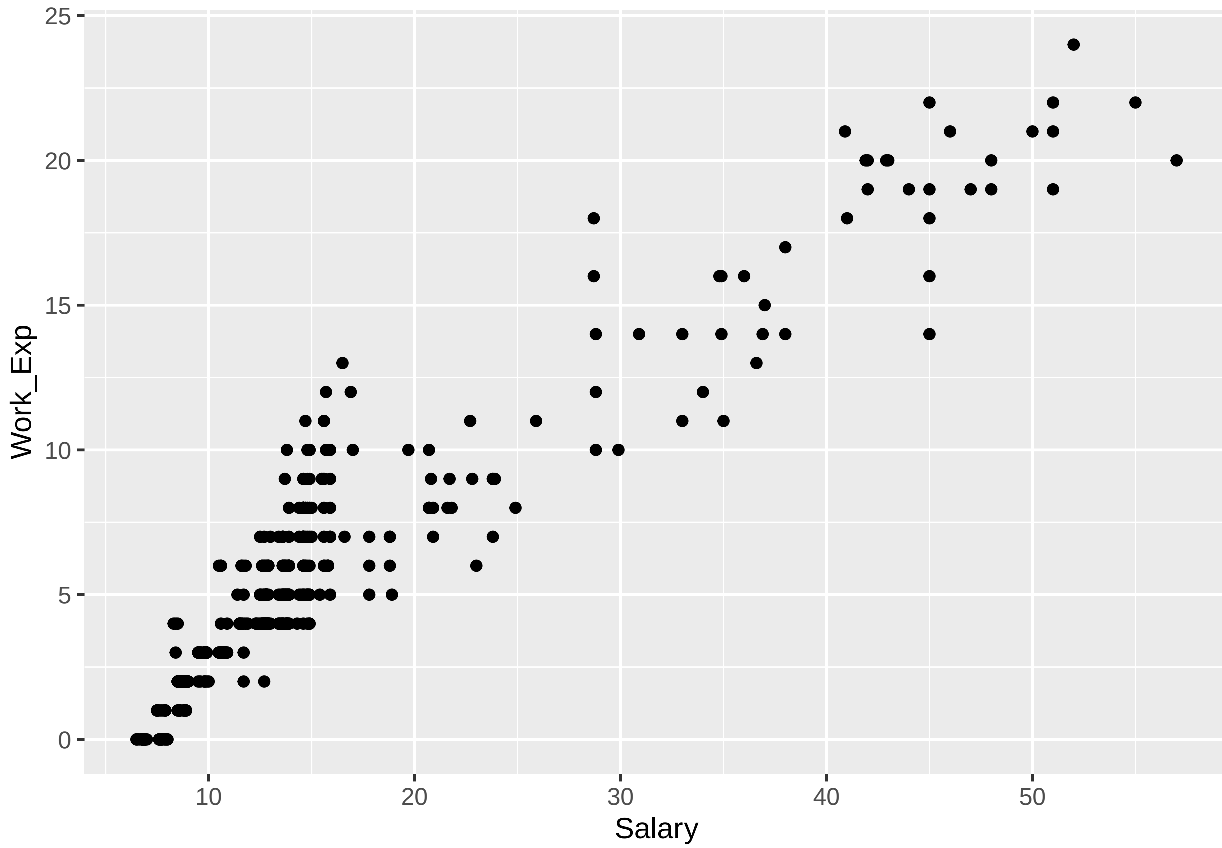
observation : 1- most of employee use Public Transport. 2- most of employee are Male. 3- most of employee have Engineer Degree. 4- most of employee have MBA Degree. 5- most of employee don't have License. 6- both Age and Distance are normally distributed. 7- both Salary and Work_Exp are skewed right, possible outlier. # varibles relationship :

```
ggplot(data = Cars_data, mapping = aes(x = Transport, y = Salary)) +
  geom_boxplot()
```



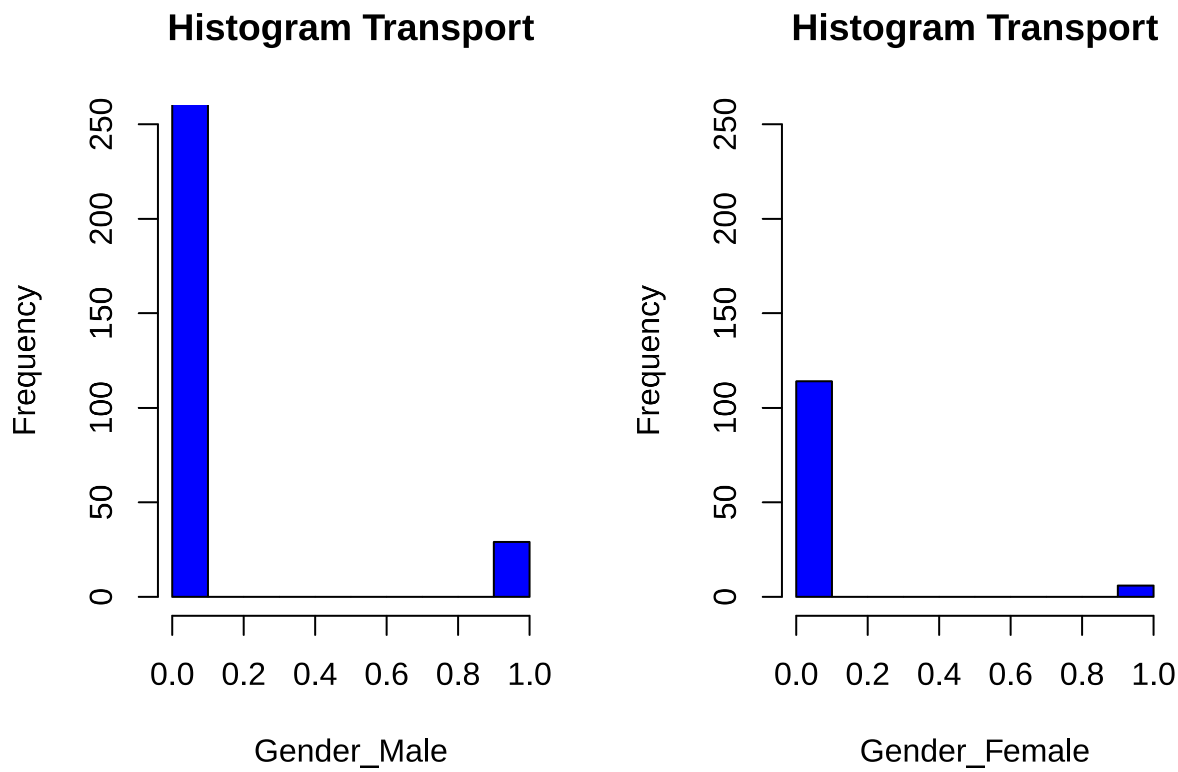
observation : high Salary employees they use Car for Trasport.

```
ggplot(data = Cars_data) +
  geom_point(mapping = aes(x = Salary, y = Work_Exp))
```



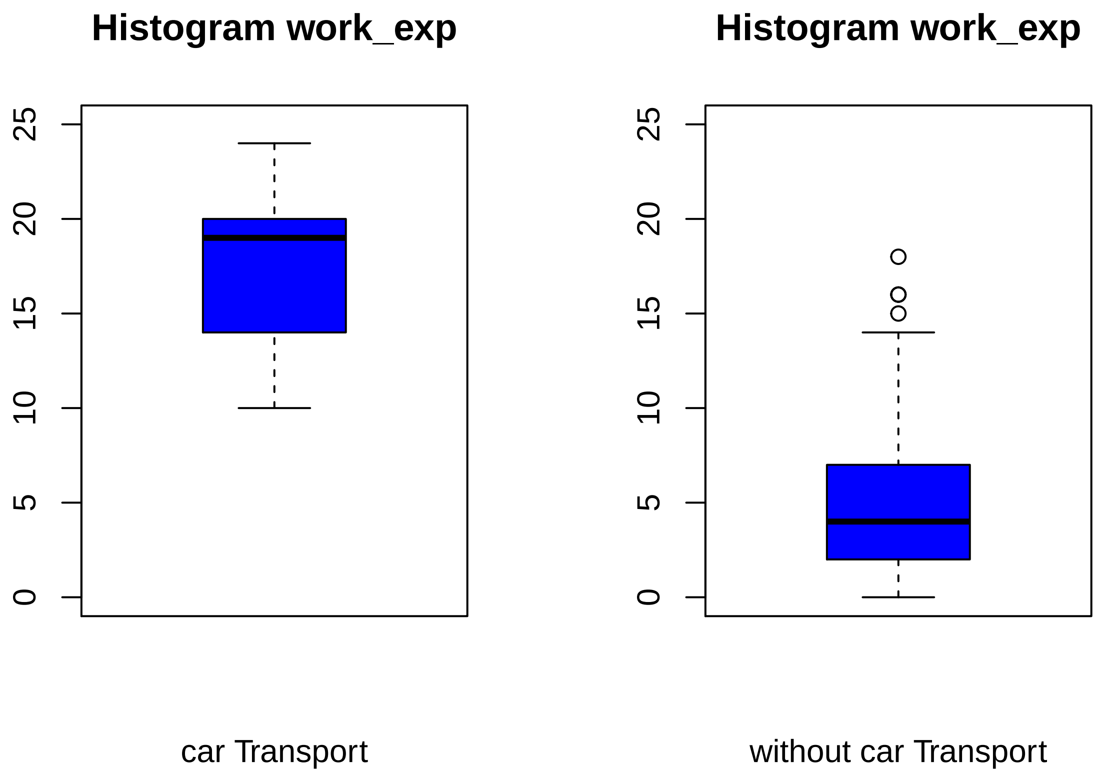
observation: 1- there is linear relationship between Salary and Work_Exp , the higher Work_Exp the higher Salary . 2- there are

```
par(mfrow=c(1,2))
hist(Cars_data$Transport_car[Cars_data$Gender==1],col = "blue",xlab = "Gender_Male",main = "Histogram T
hist(Cars_data$Transport_car[Cars_data$Gender==0],col = "blue",xlab = "Gender_Female",main = "Histogram
```



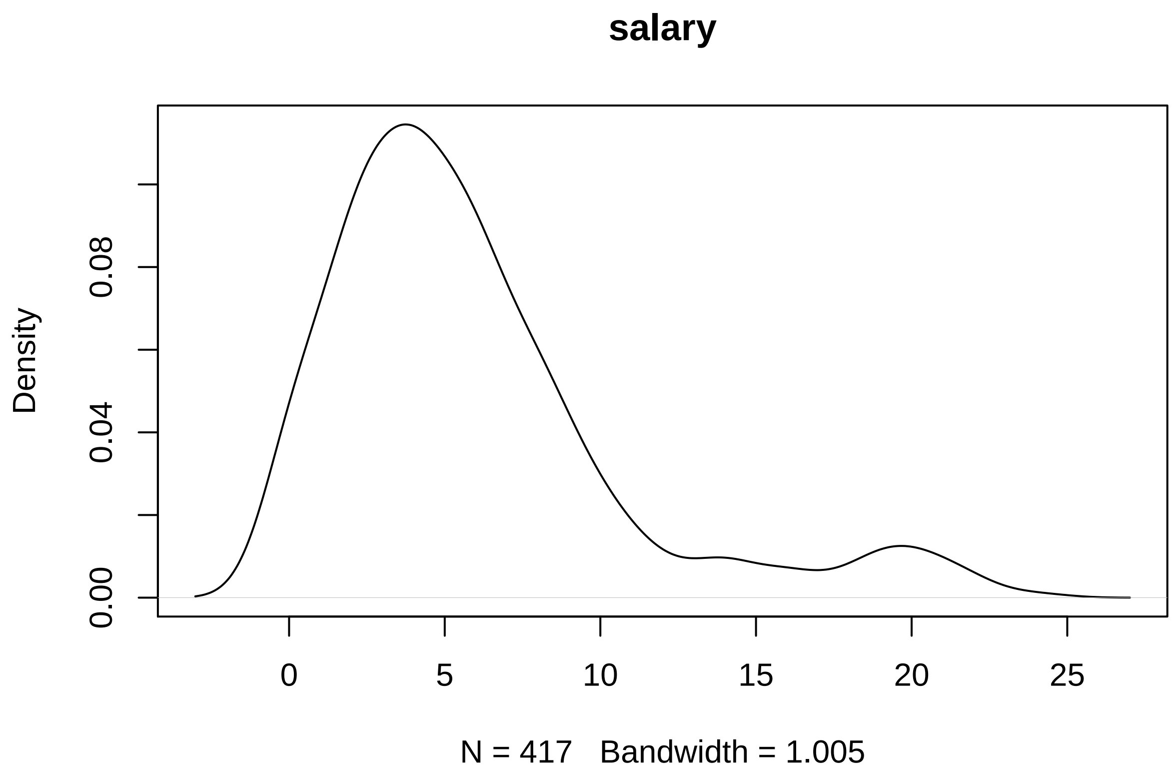
observation: most of Male and Female employee don't use car , but Male employee use car for those how use Car as Transport.

```
par(mfrow=c(1,2))
boxplot(Cars_data$Work_Exp[Cars_data$Transport_car==1],col = "blue",xlab = "car Transport",main = "Histo
boxplot(Cars_data$Work_Exp[Cars_data$Transport_car==0],col = "blue",xlab = "without car Transport",main
```

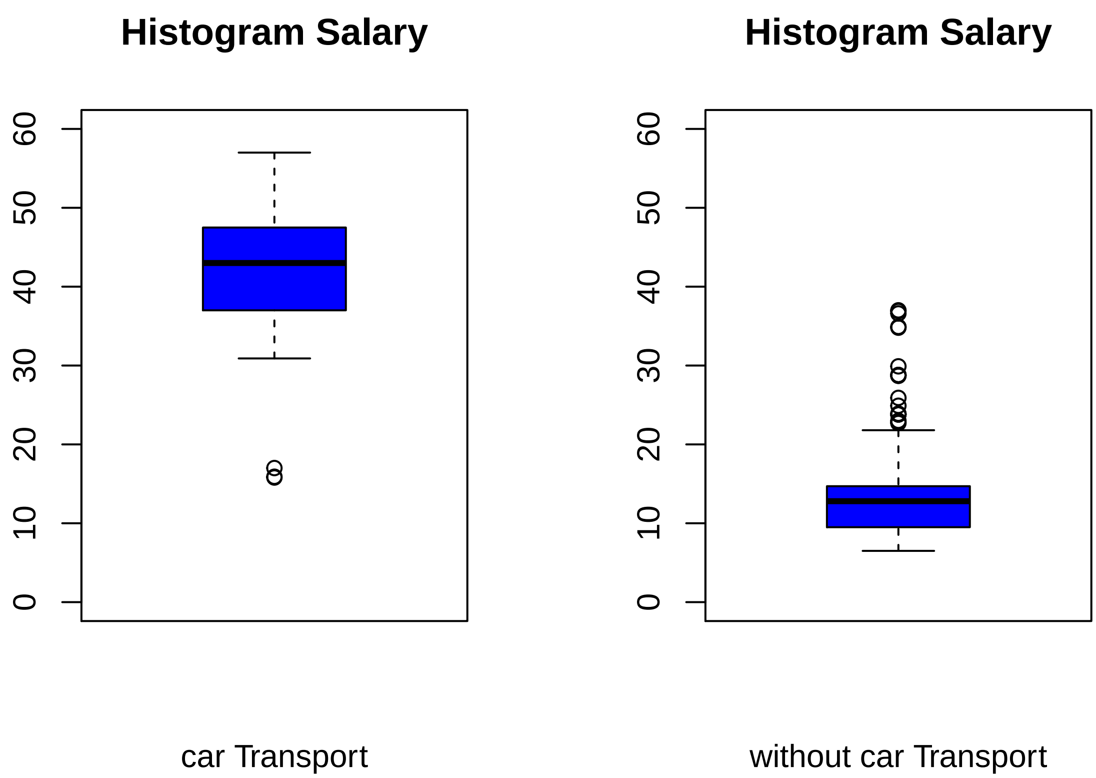


observation: most of employees that have more then 15 years of Work Experience use Car for transport.

```
plot(density(Cars_data$Work_Exp),main="salary")
```

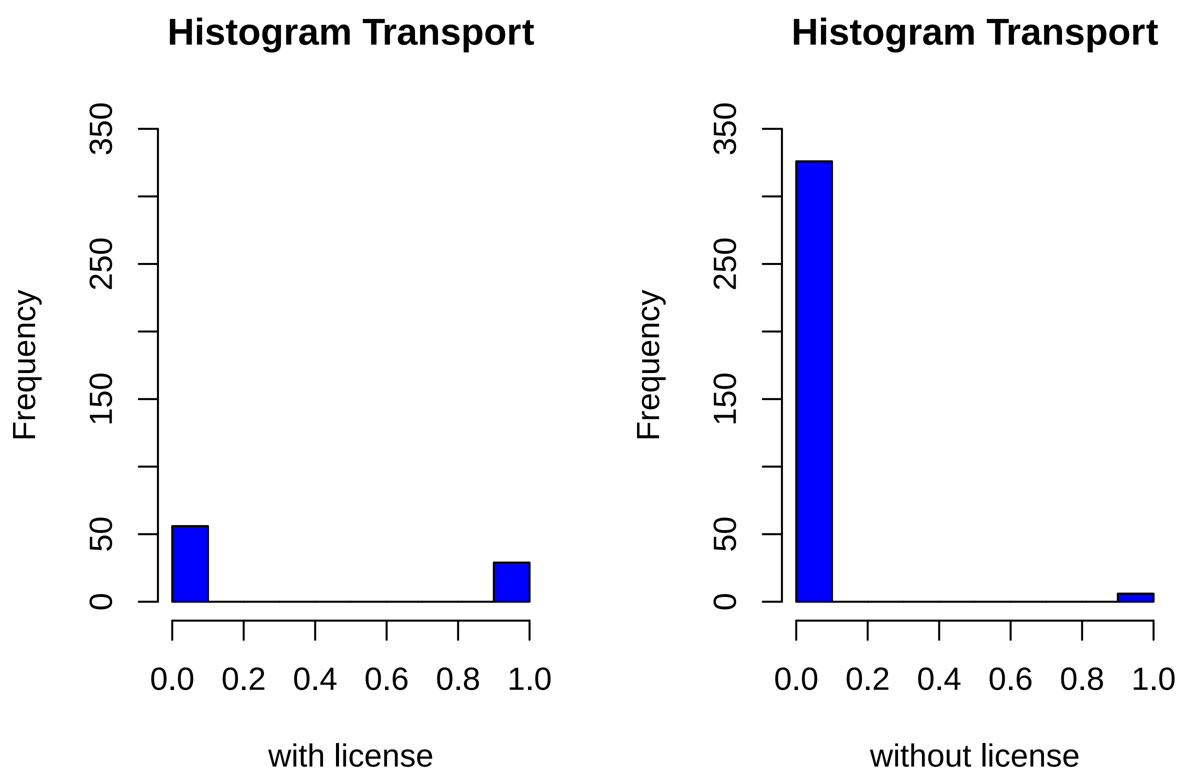



```
par(mfrow=c(1,2))
boxplot(Cars_data$Salary[Cars_data$Transport_car==1],col = "blue",xlab = "car Transport",main = "Histog
boxplot(Cars_data$Salary[Cars_data$Transport_car==0],col = "blue",xlab = "without car Transport",main =
```



observation: most of employees that have more then 30K of Salary use Car for transport.

```
par(mfrow=c(1,2))
hist(Cars_data$Transport_car[Cars_data$license==1],col = "blue",xlab = "with license",main = "Histogram
hist(Cars_data$Transport_car[Cars_data$license==0],col = "blue",xlab = "without license",main = "Histog
```

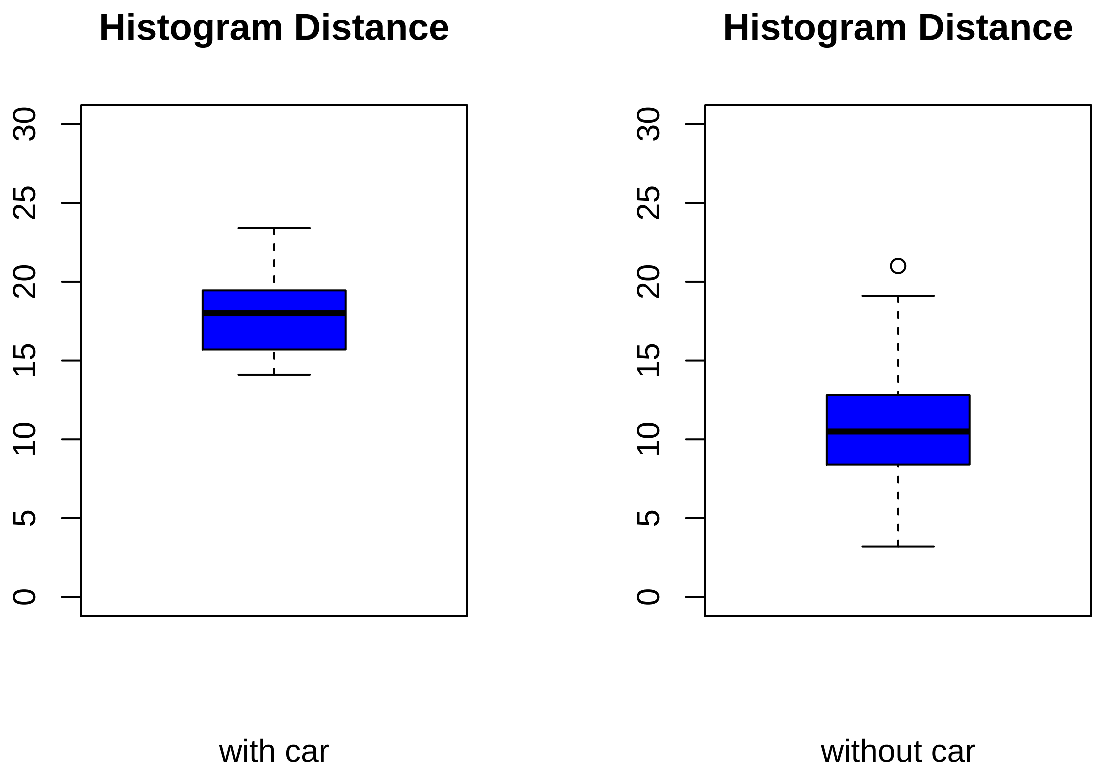



```
sum(Cars_data$Transport_car==1 & Cars_data$license==0) # to find how many have Car without license
```

```
## [1] 6
```

observation: 1- most of employees don't have license don't have Car. 2- there are 6 employees have Car but don't have License, maybe they have personal driver.

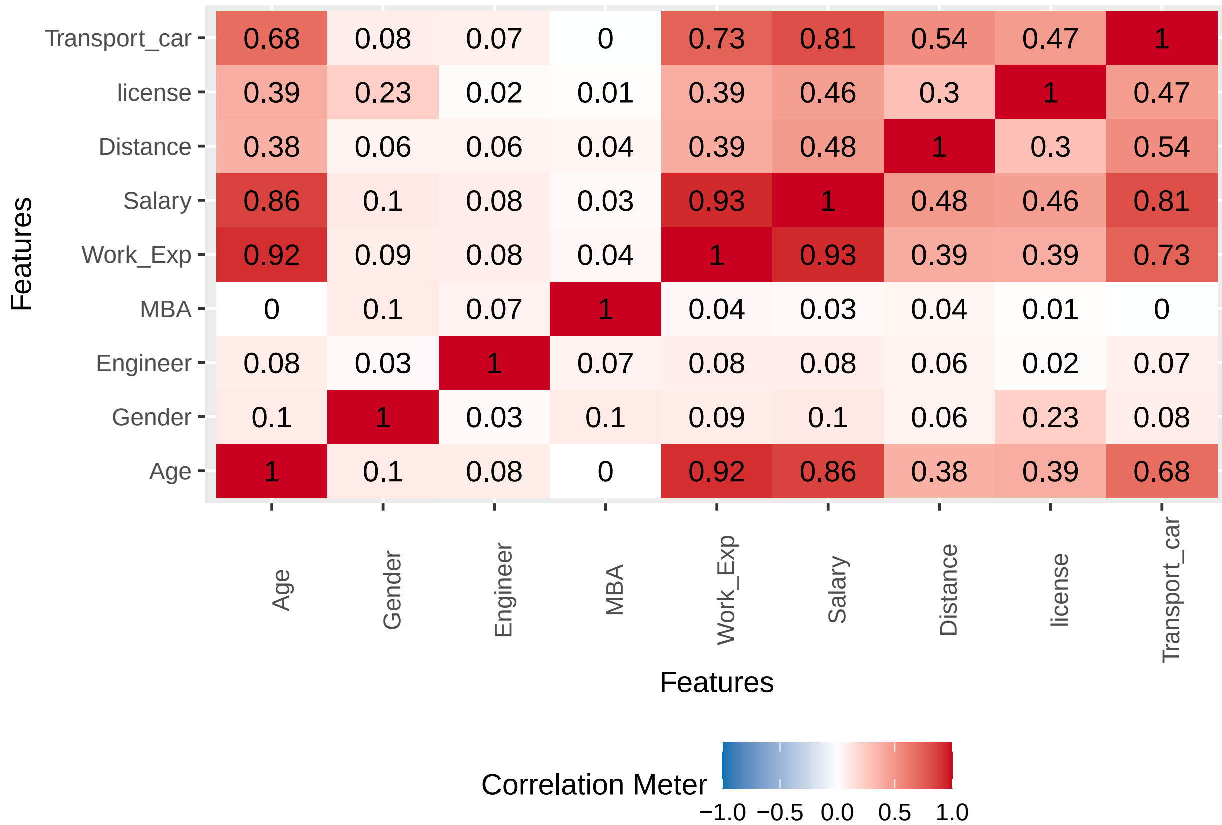
```
par(mfrow=c(1,2))
boxplot(Cars_data$Distance[Cars_data$Transport_car==1],col = "blue",xlab = "with car",main = "Histogram
boxplot(Cars_data$Distance[Cars_data$Transport_car==0],col = "blue",xlab = "without car",main = "Histog
```



observation: 1- most employees how live 15 KM or more from office have Car for Transport. 2- all employees live less then 20 KM don't have Car.

correlation:

```
# we drop Transport for now .
Cars_data1<- Cars_data[,-c(9)]
plot_correlation(Cars_data1)
```



observation: 1- as we found from the graph hight correlation between Work_exp and Salary 0.93 . 2- 0.86 correlation between Salary and Age. 3- agian hight correlation between Salary and Transport_car 0.81. # split Dataset to Train and Test :

```
set.seed(199)
Cars_data1<- as.data.frame(Cars_data1)
Cars_data1$Transport_car <- ifelse(Cars_data1$Transport==1,"Yes","No")

sample = sample.split(Cars_data1,SplitRatio = 0.75) # 75% train data , 25% test data
training = subset(Cars_data1,sample == TRUE)
testing = subset(Cars_data1,sample == FALSE)
nrow(training)
```

```
## [1] 278
```

```
nrow(testing)
```

```
## [1] 139
```

```
# the data split is equal between Train and Test with original dataset.
prop.table(table(Cars_data1$Transport_car))
```

```
##
##         No        Yes
## 0.91606715 0.08393285
```

```
prop.table(table(training$Transport_car))
```

```
##
##         No        Yes
## 0.91726619 0.08273381
```

```
prop.table(table(testing$Transport_car))
```

```
##
##         No        Yes
## 0.91366906 0.08633094
```

```
training$Transport_car<-as.factor(training$Transport_car) # to be Factor
testing$Transport_car<-as.factor(testing$Transport_car)   # to be Factor
```

Modeling :

Setting up the general parameters for training multiple models:

```
set.seed(213)
Crul<- trainControl(
  method = "repeatedcv",
  number = 5,      # number of folds
  repeats = 10,    # repeated k-fold cross-validation
  p = 10,
  allowParallel = TRUE,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
  )
```

rpart: Single decision tree :

```
rpart_model <- train(Transport_car ~ ., data = training,
                     method = "rpart",
                     minbucket = 10,
                     cp = 0,
                     tuneLength = 10,
                     trControl = Crul)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

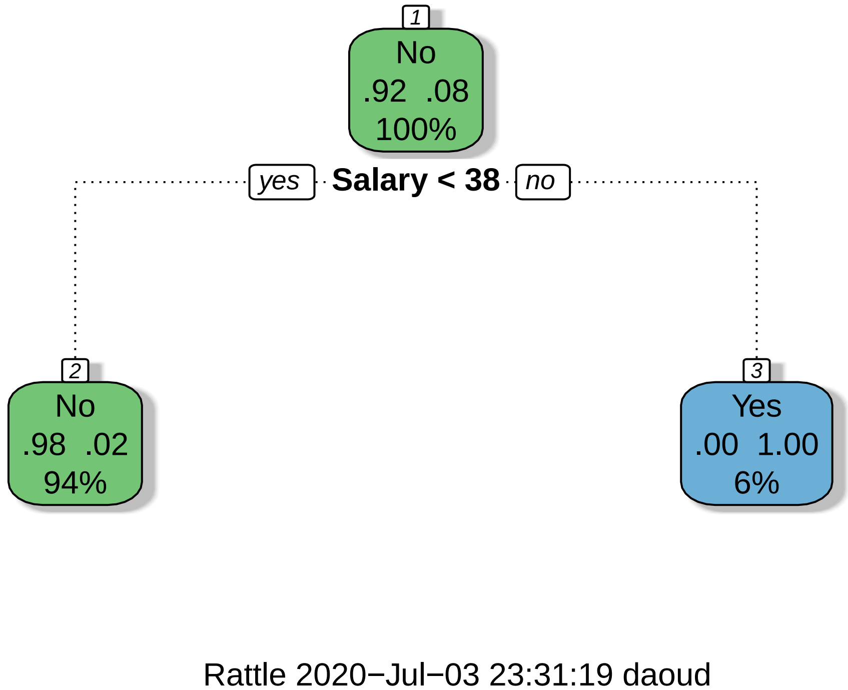
```
rpart_model
```

```
## CART
##
## 278 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 222, 222, 223, 223, 222, 222, ...
## Resampling results across tuning parameters:
##
##   cp         ROC        Sens       Spec
##   0.0000000  0.8657255  0.9843137  0.720
##   0.0821256  0.8657941  0.9866667  0.741
##   0.1642512  0.8657941  0.9866667  0.741
##   0.2463768  0.8656373  0.9862745  0.745
##   0.3285024  0.8656373  0.9862745  0.745
##   0.4106280  0.8656373  0.9862745  0.745
##   0.4927536  0.8656373  0.9862745  0.745
##   0.5748792  0.8656373  0.9862745  0.745
##   0.6570048  0.8558333  0.9866667  0.725
##   0.7391304  0.6127549  0.9945098  0.231
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.1642512.
```

```r
varimp<-varImp(rpart_model)
print(varimp)
```

```
## rpart variable importance
##
##           Overall
## Salary     100.00
## Work_Exp    79.06
## Age         75.99
## Distance    65.22
## license     28.84
## MBA          0.00
## Engineer     0.00
## Gender       0.00
```

```r
fancyRpartPlot(rpart_model$finalModel)
```



obsevation: 1- Variable Importance : salary is the most important variable 2- Work_Exp, Age, Distance, license, by sort. 3- 91% employee without Car with have Salary<38 and Distance<18 .
#Accuracy:

```
rpart_pred_test <- predict(rpart_model, newdata =testing[,1:8], type = "raw")
caret::confusionMatrix(rpart_pred_test, testing$Transport_car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  127   3
##        Yes   0   9
##
##                Accuracy : 0.9784
##                  95% CI : (0.9382, 0.9955)
##     No Information Rate : 0.9137
##     P-Value [Acc > NIR] : 0.001664
##
##                   Kappa : 0.8457
##
##  Mcnemar's Test P-Value : 0.248213
##
##             Sensitivity : 1.0000
##             Specificity : 0.7500
##          Pos Pred Value : 0.9769
```

```
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9137
##          Detection Rate : 0.9137
##    Detection Prevalence : 0.9353
##       Balanced Accuracy : 0.8750
##
##        'Positive' Class : No
##
```

KNN:

```
knn<-train(
  Transport_car~.,
  data = training,
  method="knn",
  #preProcess = c("center", "scale"),
  tuneLength = 3,
  trControl = Crul)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

#Accuracy:

```
knn_pred_test <- predict(knn, newdata =testing[,1:8], type = "raw")
caret::confusionMatrix(knn_pred_test, testing$Transport_car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  127   2
##        Yes   0  10
##
##                Accuracy : 0.9856
##                  95% CI : (0.949, 0.9983)
##     No Information Rate : 0.9137
##     P-Value [Acc > NIR] : 0.0003537
##
##                   Kappa : 0.9013
##
##  Mcnemar's Test P-Value : 0.4795001
##
##             Sensitivity : 1.0000
##             Specificity : 0.8333
##          Pos Pred Value : 0.9845
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9137
##          Detection Rate : 0.9137
##    Detection Prevalence : 0.9281
```

```
##       Balanced Accuracy : 0.9167
##
##        'Positive' Class : No
##
```

model naive base:

```
naive_base<-train(
  Transport_car~.,
  data = training,
  method="naive_bayes",
  trControl = Crul)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

#Accuracy:

```
naive_pred_test <- predict(naive_base, newdata =testing[,1:8], type = "raw")
caret::confusionMatrix(naive_pred_test, testing$Transport_car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  127   1
##        Yes   0  11
##
##                Accuracy : 0.9928
##                  95% CI : (0.9606, 0.9998)
##     No Information Rate : 0.9137
##     P-Value [Acc > NIR] : 5.011e-05
##
##                   Kappa : 0.9526
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9167
##          Pos Pred Value : 0.9922
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9137
##          Detection Rate : 0.9137
##    Detection Prevalence : 0.9209
##       Balanced Accuracy : 0.9583
##
##        'Positive' Class : No
##
```

Logistic Regression :

```
glm<-train(
  Transport_car~.,
  data = training,
  method = "glm",
  family = "binomial",
  trControl = Crul
)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

#Accuracy:

```r
glm_pred_test <- predict(glm, newdata =testing[,1:8], type = "raw")
caret::confusionMatrix(glm_pred_test, testing$Transport_car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  126   1
##        Yes   1  11
##
##                Accuracy : 0.9856
##                  95% CI : (0.949, 0.9983)
##     No Information Rate : 0.9137
##     P-Value [Acc > NIR] : 0.0003537
##
##                   Kappa : 0.9088
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.9921
##             Specificity : 0.9167
##          Pos Pred Value : 0.9921
##          Neg Pred Value : 0.9167
##              Prevalence : 0.9137
##          Detection Rate : 0.9065
##    Detection Prevalence : 0.9137
##       Balanced Accuracy : 0.9544
##
##        'Positive' Class : No
##
```

Random Forest :

```
rf<-train(
  Transport_car~.,
  data = training,
  method = "rf",
  ntree = 50,
  maxdepth = 7,
  tuneLength = 20,
  trControl = Crul)
```

```
## note: only 7 unique complexity parameters in default grid. Truncating the grid to 7 .
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

#Accuracy:

```
rf_pred_test <- predict(rf, newdata =testing[,1:8], type = "raw")
caret::confusionMatrix(rf_pred_test, testing$Transport_car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  127   3
##        Yes   0   9
##
##                Accuracy : 0.9784
##                  95% CI : (0.9382, 0.9955)
##     No Information Rate : 0.9137
##     P-Value [Acc > NIR] : 0.001664
##
##                   Kappa : 0.8457
##
##  Mcnemar's Test P-Value : 0.248213
##
##             Sensitivity : 1.0000
##             Specificity : 0.7500
##          Pos Pred Value : 0.9769
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9137
##          Detection Rate : 0.9137
##    Detection Prevalence : 0.9353
##       Balanced Accuracy : 0.8750
##
##        'Positive' Class : No
##
```

bagging :

```
bagging_model<-train(
  Transport_car~.,
  data = training,
  method = "treebag",
  nleaves=10,
  ntrees=5,
  trControl=Crul,
  importance=TRUE
)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

#Accuracy:

```
bagging_predictions_test <- predict(bagging_model, newdata = testing, type = "raw")
caret::confusionMatrix(bagging_predictions_test, testing$Transport_car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  127   3
##        Yes   0   9
##
##                Accuracy : 0.9784
##                  95% CI : (0.9382, 0.9955)
##     No Information Rate : 0.9137
##     P-Value [Acc > NIR] : 0.001664
##
##                   Kappa : 0.8457
##
##  Mcnemar's Test P-Value : 0.248213
##
##             Sensitivity : 1.0000
##             Specificity : 0.7500
##          Pos Pred Value : 0.9769
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9137
##          Detection Rate : 0.9137
##    Detection Prevalence : 0.9353
##       Balanced Accuracy : 0.8750
##
##        'Positive' Class : No
##
```

xgboost: (without SMOTE)

```
xgb.grid <- expand.grid(nrounds = 150,
                        eta = c(0.01),
                        max_depth = c(4,7),
                        gamma = 0,                #default=0
                        colsample_bytree = 1,     #default=1
                        min_child_weight = 1,     #default=1
                        subsample = 1             #default=1
    )
xgb_model <-train(Transport_car~.,
                  data=training,
                  method="xgbTree",
                  trControl=Crul,
                  tuneGrid=xgb.grid,
                  verbose=T,
                  nthread = 2
    )
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

#Accuracy:

```
xgb_predictions_test <- predict(xgb_model, newdata = testing, type = "raw")
confusionMatrix(xgb_predictions_test, testing$Transport_car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  127   1
##        Yes   0  11
##
##                Accuracy : 0.9928
##                  95% CI : (0.9606, 0.9998)
##     No Information Rate : 0.9137
##     P-Value [Acc > NIR] : 5.011e-05
##
##                   Kappa : 0.9526
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9167
##          Pos Pred Value : 0.9922
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9137
##          Detection Rate : 0.9137
##    Detection Prevalence : 0.9209
##       Balanced Accuracy : 0.9583
```

```
##
##        'Positive' Class : No
##
```

SMOTE:

```r
table(training$Transport_car)
```

```
##
##  No Yes
## 255  23
```

```r
prop.table(table(training$Transport_car))
```

```
##
##         No        Yes
## 0.91726619 0.08273381
```

```r
smote_train <- SMOTE(Transport_car ~ ., data  = training,
                     perc.over = 3000,
                     perc.under = 300,
                     k = 5)

prop.table(table(smote_train$Transport_car))
```

```
##
##        No       Yes
## 0.7438017 0.2561983
```

```r
table(smote_train$Transport_car)
```

```
##
##   No  Yes
## 2070  713
```

xgboost: (with SMOTE)

```r
xgb.grid <- expand.grid(nrounds = 150,
                            eta = c(0.01),
                            max_depth = c(4,7),
                            gamma = 0,                #default=0
                            colsample_bytree = 1,     #default=1
                            min_child_weight = 1,     #default=1
                            subsample = 1             #default=1
    )
smote_model <-train(Transport_car~.,
```

```r
                  data=smote_train,
                  method="xgbTree",
                  trControl=Crul,
                  tuneGrid=xgb.grid,
                  verbose=T,
                  nthread = 2
    )
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

#Accuracy:

```r
smote_predictions_test <- predict(smote_model, newdata = testing, type = "raw")
confusionMatrix(smote_predictions_test, testing$Transport_car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  127   1
##        Yes   0  11
##
##                Accuracy : 0.9928
##                  95% CI : (0.9606, 0.9998)
##     No Information Rate : 0.9137
##     P-Value [Acc > NIR] : 5.011e-05
##
##                   Kappa : 0.9526
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9167
##          Pos Pred Value : 0.9922
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9137
##          Detection Rate : 0.9137
##    Detection Prevalence : 0.9209
##       Balanced Accuracy : 0.9583
##
##        'Positive' Class : No
##
```

COMPARING MODELS

```r
# Compare model performances using resample()
models_to_compare <- resamples(list(Logistic_Regression = glm,
                                 Navie_Bayes = naive_base,
                                 KNN = knn,
```

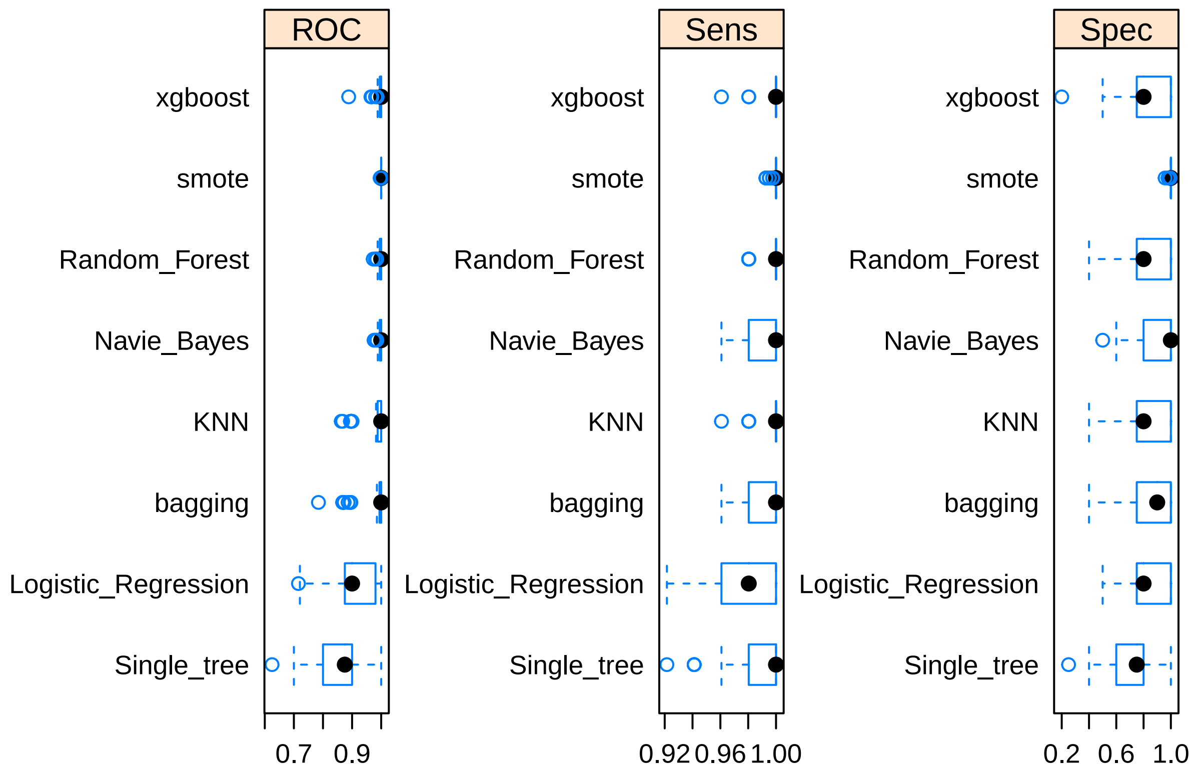
```r
                                    bagging = bagging_model,
                                    Single_tree = rpart_model,
                                    smote=smote_model,
                                    Random_Forest = rf,
                                    xgboost=xgb_model
                                    ))

# Summary of the models performances
summary(models_to_compare)
```

```
##
## Call:
## summary.resamples(object = models_to_compare)
##
## Models: Logistic_Regression, Navie_Bayes, KNN, bagging, Single_tree, smote, Random_Forest, xgboost
## Number of resamples: 50
##
## ROC
##                          Min.   1st Qu. Median      Mean   3rd Qu. Max. NA's
## Logistic_Regression 0.7156863 0.8750000  0.900 0.9175196 0.9803922    1    0
## Navie_Bayes         0.9754902 0.9950980  1.000 0.9963725 1.0000000    1    0
## KNN                 0.8627451 0.9883578  1.000 0.9793333 1.0000000    1    0
## bagging             0.7843137 0.9946078  1.000 0.9754608 1.0000000    1    0
## Single_tree         0.6250000 0.8000000  0.875 0.8657941 0.9000000    1    0
## smote               0.9963003 1.0000000  1.000 0.9999159 1.0000000    1    0
## Random_Forest       0.9725490 0.9950980  1.000 0.9960098 1.0000000    1    0
## xgboost             0.8882353 0.9950980  1.000 0.9941373 1.0000000    1    0
##
## Sens
##                          Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
## Logistic_Regression 0.9215686 0.9607843 0.9803922 0.9780392       1    1    0
## Navie_Bayes         0.9607843 0.9803922 1.0000000 0.9882353       1    1    0
## KNN                 0.9607843 1.0000000 1.0000000 0.9976471       1    1    0
## bagging             0.9607843 0.9852941 1.0000000 0.9941176       1    1    0
## Single_tree         0.9215686 0.9803922 1.0000000 0.9866667       1    1    0
## smote               0.9927536 1.0000000 1.0000000 0.9991787       1    1    0
## Random_Forest       0.9803922 1.0000000 1.0000000 0.9972549       1    1    0
## xgboost             0.9607843 1.0000000 1.0000000 0.9984314       1    1    0
##
## Spec
##                          Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
## Logistic_Regression 0.5000000    0.75   0.80 0.8340000     1.0    1    0
## Navie_Bayes         0.5000000    0.80   1.00 0.8980000     1.0    1    0
## KNN                 0.4000000    0.75   0.80 0.8330000     1.0    1    0
## bagging             0.4000000    0.75   0.90 0.8590000     1.0    1    0
## Single_tree         0.2500000    0.60   0.75 0.7410000     0.8    1    0
## smote               0.9577465    1.00   1.00 0.9977534     1.0    1    0
## Random_Forest       0.4000000    0.75   0.80 0.8230000     1.0    1    0
## xgboost             0.2000000    0.75   0.80 0.8190000     1.0    1    0
```

Draw box plots to compare models

```
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(models_to_compare, scales=scales)
```



output: pdf_document

Summary:

1- we try to understand what transport employees prefers to commute to their office Car or other , so we upload and Data Preparation and split data in to two part Train,Test and we applied multiple “7” models with general parameters. 2- lets discuss the result : biased on the best accuracy : “bagging, xgboost, random forest, naive baise” had same acuuracy value : 99.28% , after that “smote, Logistic Regression” had Accuracy of : 98.56%, befor last “knn” with accuracy of : 97.84% , and last “Single decision tree” with :96.4%. 3- but when sort baised on ROC and Sensitivity and Specificity : witch are very important for choosing the model, “smote” is the best of all, and then “bagging,naive baise” . 4- the last disussion : since smote is the best on ROC and Sensitivity and Specificity and had Accuracy of : 98.56% witch is Good. I will go with smote model.