PROJECT 5

MACHINE LEARNING

Exploratory Data Analysis

> dim(data)

[1] 444   9

We have 444 observations and 9 variables.

```
> str(data)
'data.frame':	444 obs. of  9 variables:
 $ Age      : int  28 23 29 28 27 26 28 26 22 27 ...
 $ Gender   : Factor w/ 2 levels "Female","Male": 2 1 2 1 2 2 2 1 2 2 ...
 $ Engineer : int  0 1 1 1 1 1 1 1 1 1 ...
 $ MBA      : int  0 0 0 1 0 0 0 0 0 0 ...
 $ Work.Exp : int  4 4 7 5 4 4 5 3 1 4 ...
 $ Salary   : num  14.3 8.3 13.4 13.4 13.4 12.3 14.4 10.5 7.5 13.5 ...
 $ Distance : num  3.2 3.3 4.1 4.5 4.6 4.8 5.1 5.1 5.1 5.2 ...
 $ license  : int  0 0 0 0 0 1 0 0 0 0 ...
 $ Transport: Factor w/ 3 levels "2Wheeler","Car",..: 3 3 3 3 3 3 1 3 3 3 ...
```

- We have 2, factor variables Engineering, MBA, License needs to be converted into a factor variable.
- Also, we need to convert Transport to have binary outputs, Car = 1, 2wheeler and public transport = 0. As we are only want to know if employees are using a car or not.
- We will also convert, Gender into binary. Male =1 and Female =0.

```
> # Checking for NA
> sapply(data,function(x) sum(is.na(x)))
      Age    Gender  Engineer       MBA  Work.Exp    Salary  Distance   license Transport 
        0         0         0         1         0         0         0         0         0 
> # Checking # of unique values in each column
> sapply(data,function(x) length(unique(x)))
      Age    Gender  Engineer       MBA  Work.Exp    Salary  Distance   license Transport 
       25         2         2         3        24       122       137         2         3 
```

While checking for NA and unique values we find that there is one NA present in the MBA column. We will get rid of that row.

> #Removing NA
> data = na.omit(data)

We now have 443 observations.

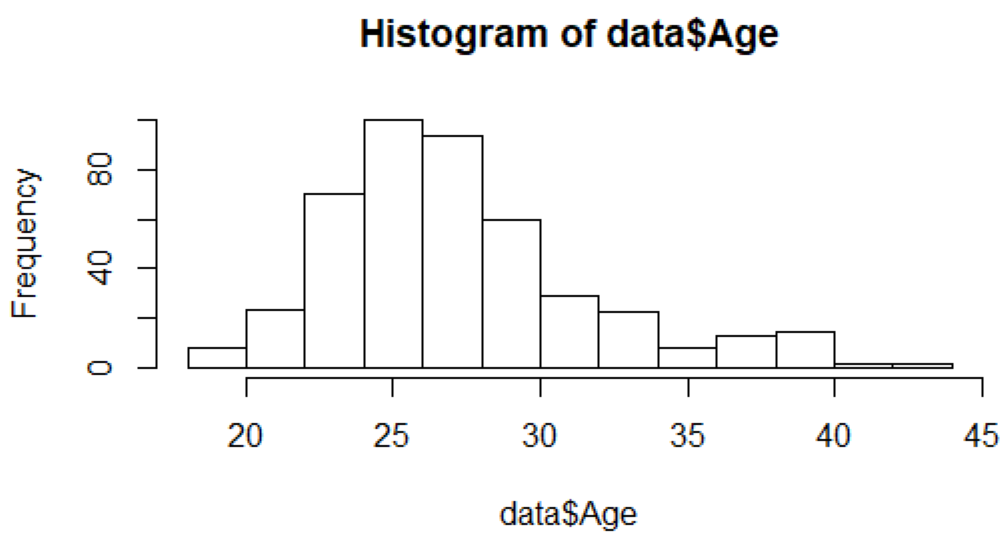
```
> # Converting Transport values to 0 or 1
> data$Transport = ifelse(data$Transport == 'Car', 1,0)
> # Male = 1 , Female = 0
> data$Gender = ifelse(data$Gender == 'Male', 1,0)
> table(data$Transport)

  0   1 
382  61 
>
> sum(data$Transport == 1)/nrow(data)
[1] 0.1376975
> sum(data$Gender == 1)/nrow(data)
[1] 0.7133183
>
```

After converting Gender and TRansport into binary we see that:

- 13.76% of Employees use a car.
- 71.33% of Employees are Males.
- We will apply SMOTE to synthetically increase our minority class.

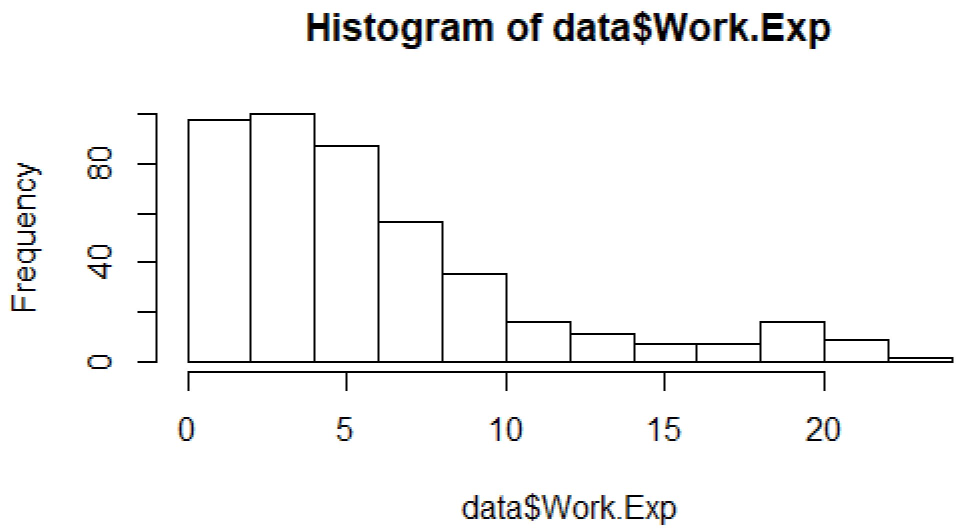
Univariate Analysis

> hist(data$Age)



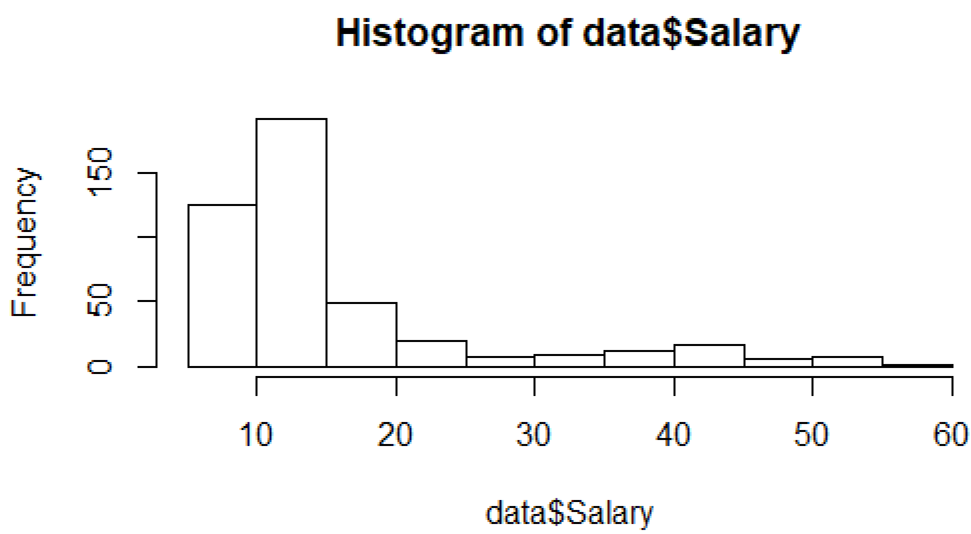
Age is slightly right-skewed.

> hist(data$Work.Exp)



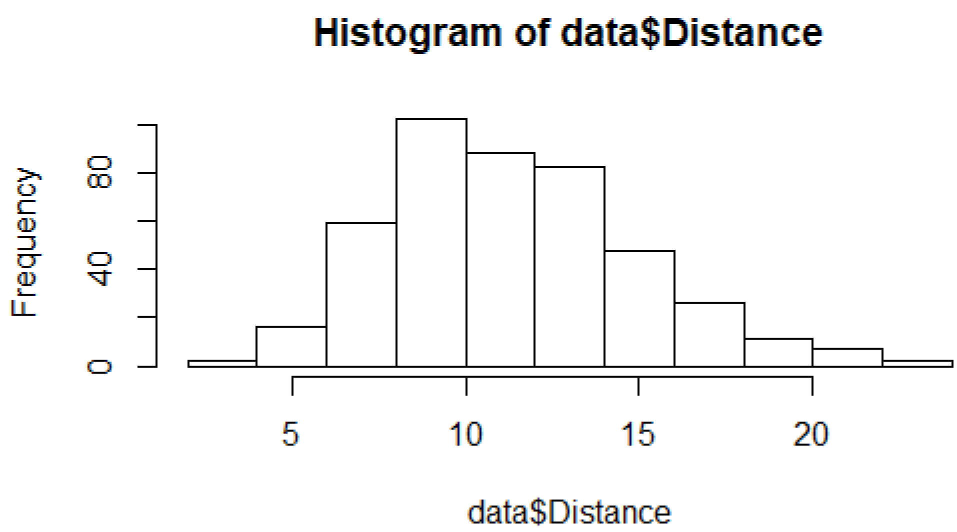
- Work.Exp is very right-skewed and tailed. Outlies might be present.
- Also, we can clearly see that there are more juniors that seniors in the firm.

> hist(data$Salary)



Salary is not evenly distributed.

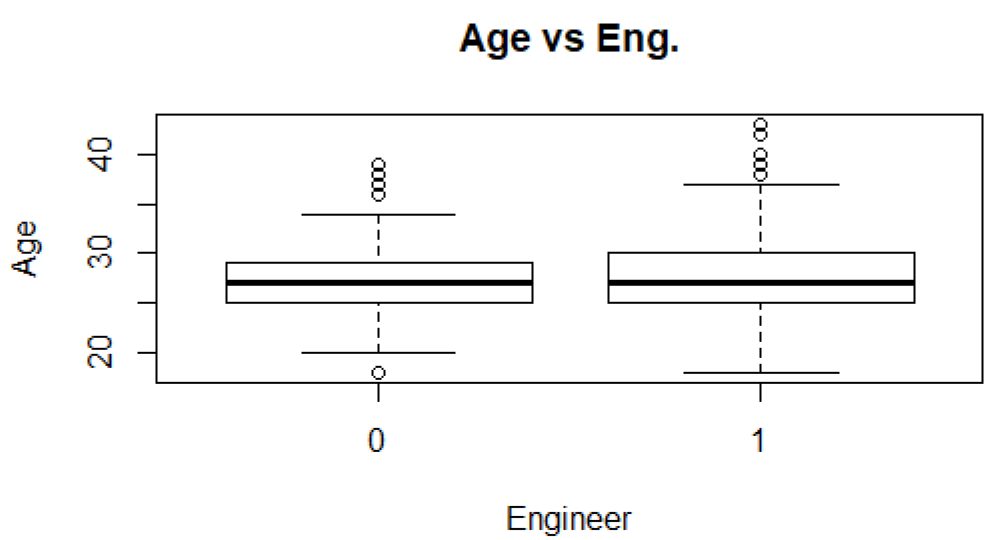
> hist(data$Distance)



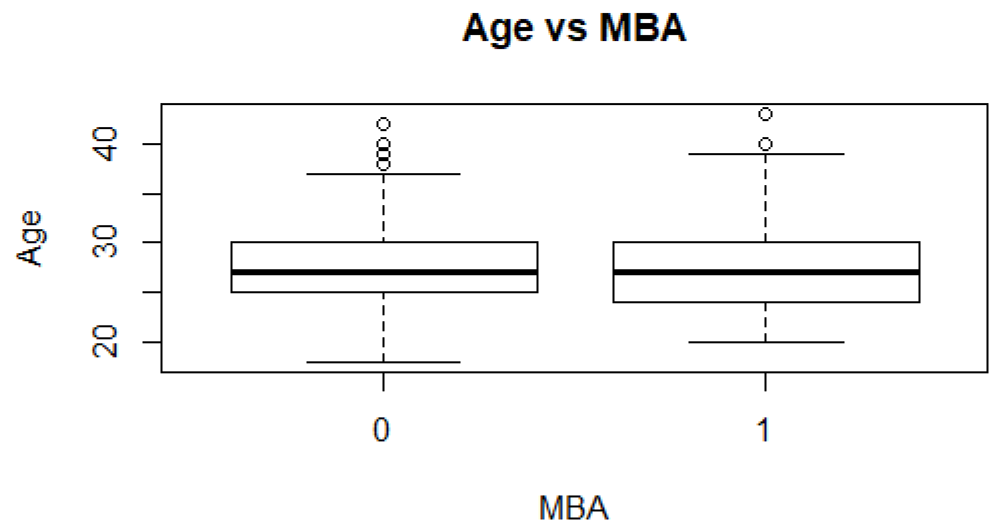
Distance is slightly right-skewed but the distribution is almost even.

Bivariate Analysis

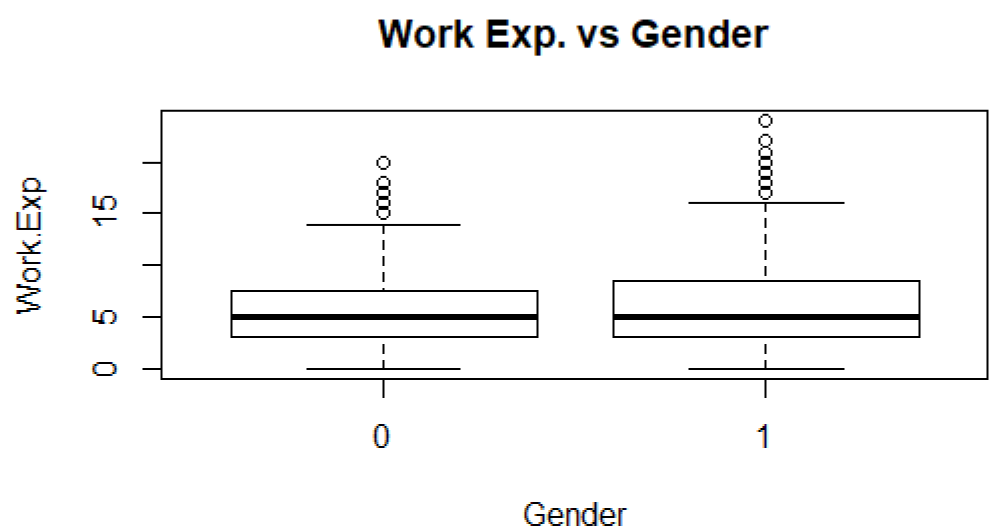
> boxplot(Age~Engineer, main = "Age vs Eng.")



> boxplot(Age~MBA, main = "Age vs MBA")

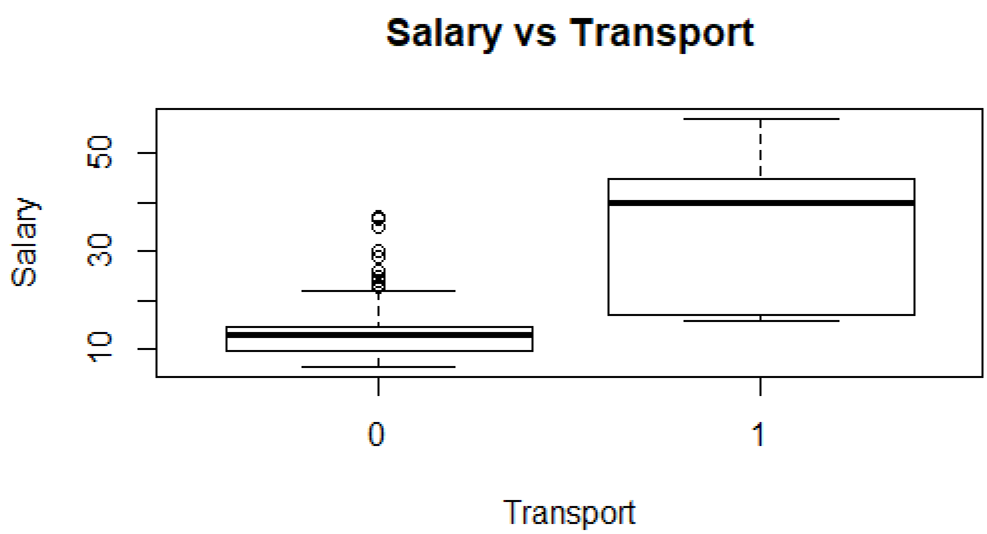


> boxplot(Work.Exp~Gender, main = "Work Exp. vs Gender")



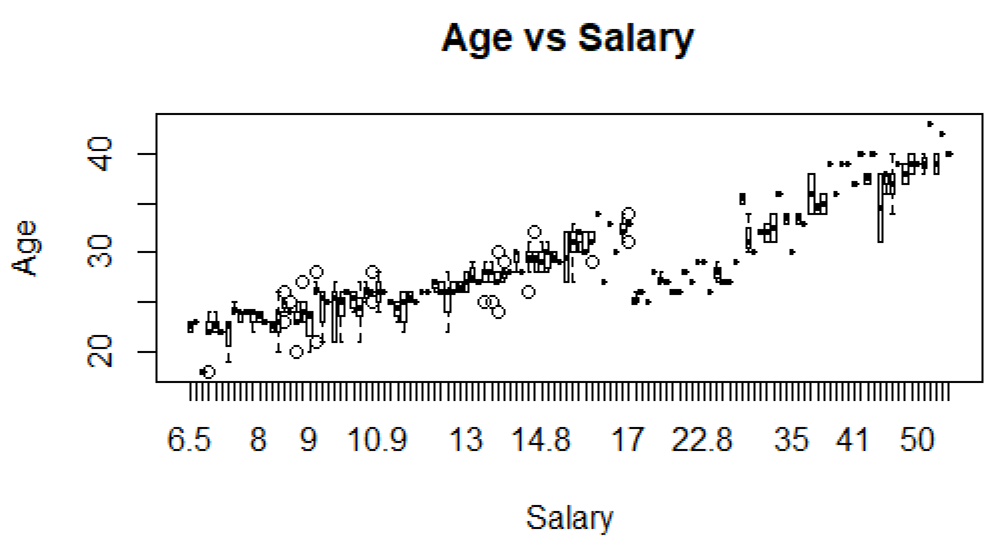
There is not a lot of difference between the work experience in the two genders, with mean work experience being 5 years for both.

> boxplot(Salary~Transport, main = "Salary vs Transport")



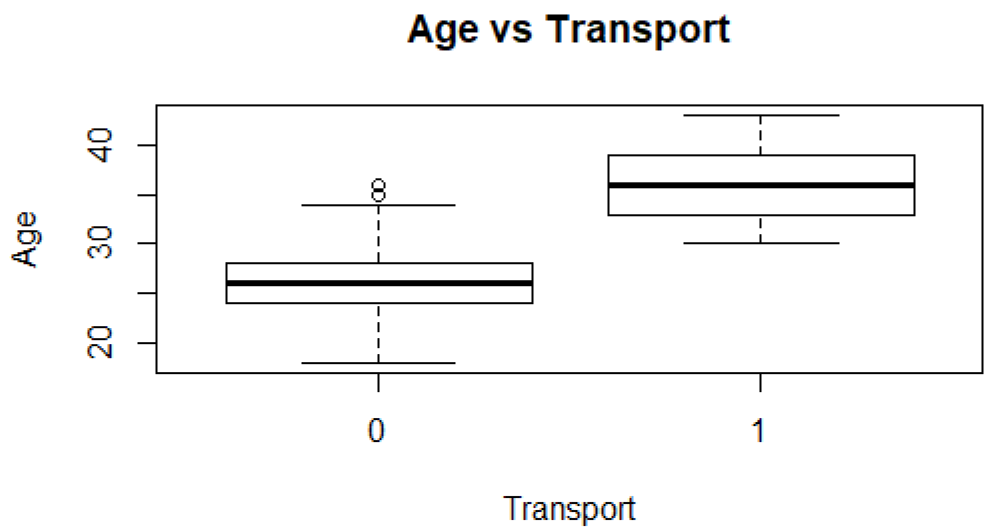
Plot clearly shows as the salary increases the chance of using a car also increases.

> boxplot(Age~Salary, main = "Age vs Salary")

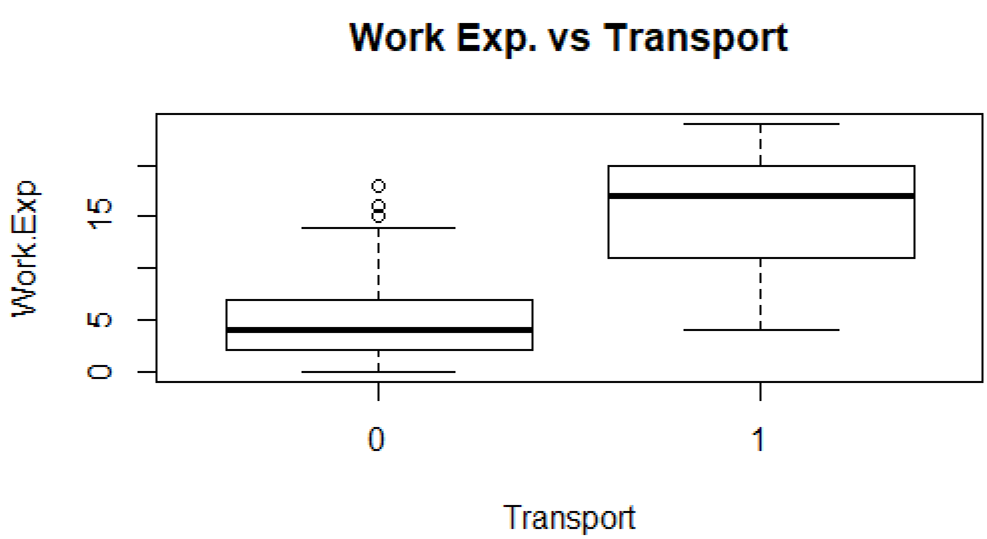


As expected, with Age, Salary clearly increase.

> boxplot(Age~Transport, main = "Age vs Transport")

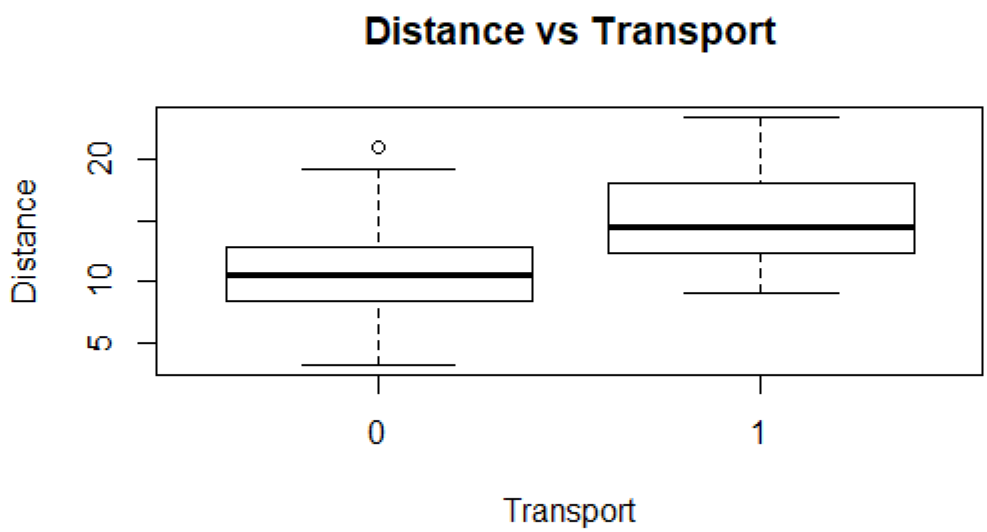


> boxplot(Work.Exp~Transport, main = "Work Exp. vs Transport")



The plot is similar to the previous plot, as the work experience increases, the chances of using a car also increase.

```
> boxplot(Distance~Transport, main = "Distance vs Transport")
```

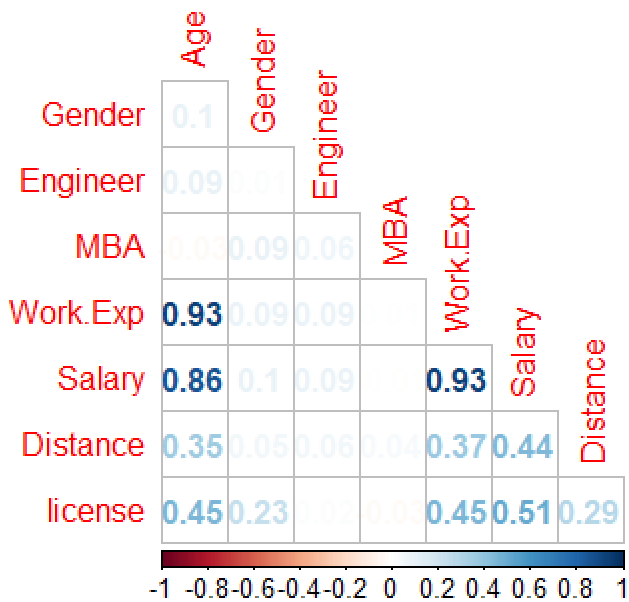


For greater distances of more than 20, a car is preferred. For shorter distances, employees prefer using 2wheeler or public transport.

Checking for Co-relation

- We will use, corrplot as well as run a logistic regression model to compute the Variance Inflation Factor (VIF) based on which we will decide if we need to drop a variable.
- From the analysis above it is clearly visible that MBA and Engineering show similar patterns.
- Work Experience and Age plots are very similar.

```
library(corrplot)
correlations = cor(data[,-9])
summary(correlations)
corrplot(correlations, type="lower", method = 'number', diag = FALSE)
```



- There is a high negative correlation between Age and Work.Experience.
- There is a high positive correlation between work experience and Salary.
- Based on the VIF values we will decide which variable to drop.

```
#Logisticregression(VIF check)
vif_logistic = glm(Transport~., data=data, family=binomial(link="logit"))
summary(vif_logistic)

# VIF
library(car)
vif(vif_logistic)
```

```
> vif(vif_logistic)
      Age    Gender  Engineer       MBA  Work.Exp    Salary  Distance   license
11.903723  1.486422  1.112810  1.461142 16.950407  3.970501  1.714671  1.844857
```

- Work Experience and Age have VIF values above 10.
- For future modeling, we will drop work experience.

```
#After removing Work Experience
vif_logistic01 = glm(Transport~., data=data[,-5], family=binomial(link="logit"))
summary(vif_logistic01)
vif(vif_logistic01)
```

```
> vif(vif_logistic01)
     Age   Gender Engineer      MBA   Salary Distance  license
1.762831 1.175206 1.064623 1.304329 1.546412 1.229332 1.258074
>
```

All the VIF values are below 5 after removing the variable Work Experience. Good to proceed.

SMOTE

Using the SMOTE function, we will synthetically increase the minority class (1).
For testing purpose, the values of prec.under was varied to find the best split.

```
#SMOTE (Increase minority rate to 50%)
library(DMwR)

smote.train = subset(data, split == TRUE)
smote.test = subset(data, split == FALSE)

str(data$Transport)
smote.train$Transport = as.factor(smote.train$Transport)

balanced.gd = SMOTE(Transport ~., smote.train, perc.over = 4800, k = 5, perc.under = 100)
table(balanced.gd$Transport)
sum(balanced.gd$Transport == 1)/nrow(balanced.gd)
```

```
> balanced.gd = SMOTE(Transport ~., smote.train, perc.over = 4800, k = 5, perc.under = 100)
>
> table(balanced.gd$Transport)

   0    1
2208 2254
> sum(balanced.gd$Transport == 1)/nrow(balanced.gd)
[1] 0.5051546
> smote_logistic01 = glm(Transport~., data=balanced.gd01, family=binomial(link="logit"))
```

We have increased the minority class to 50%

Applying this data to logistic regression

```
#Logistic Regression(with SMOTE)
smote_logistic = glm(Transport~., data=balanced.gd, family=binomial(link="logit"))

summary(smote_logistic)

smote.test$log.pred = predict(smote_logistic, smote.test[1:7], type="response")

table(smote.test$Transport,smote.test$log.pred>0.5)
```

```
>
> smote.test$log.pred = predict(smote_logistic, smote.test[1:7], type="response")
>
> table(smote.test$Transport,smote.test$log.pred>0.5)

    FALSE TRUE
  0    93    3
  1     3   12
```

Interpretation:
TPR - 12/15 - 80%
FPR - 93/96 - 96.85%

```
#SMOTE (Increase minority rate to 30%)
smote.train01 = subset(data, split == TRUE)
smote.test01 = subset(data, split == FALSE)

balanced.gd01 = SMOTE(Transport ~., smote.train, perc.over = 4800, k = 5, perc.under = 200)

table(balanced.gd01$Transport)
sum(balanced.gd01$Transport == 1)/nrow(balanced.gd01)
```

```
> balanced.gd01 = SMOTE(Transport ~., smote.train, perc.over = 4800, k = 5, perc.under = 200)
>
> table(balanced.gd01$Transport)

   0    1
4416 2254
> sum(balanced.gd01$Transport == 1)/nrow(balanced.gd01)
[1] 0.337931
```

We have increased our minority class to 34%

#Logistic Regression(with SMOTE)
smote_logistic01 = glm(Transport~., data=balanced.gd01, family=binomial(link="logit"))

summary(smote_logistic01)

smote.test01$log.pred = predict(smote_logistic01, smote.test01[1:7], type="response")

table(smote.test01$Transport,smote.test01$log.pred>0.5)

```
> table(smote.test01$Transport,smote.test01$log.pred>0.5)

     FALSE TRUE
  0     93    3
  1      5   10
>
```

TPR - 10/15 - 66.66%
FPR - 93/96 - 96.85%

We can clearly see that we got better results, in the first case when our minority class was at 50%. So we will be using the same in future models.

Logistic Regression

#Logistic Regression(with SMOTE)
smote_logistic = glm(Transport~., data=balanced.gd, family=binomial(link="logit"))

summary(smote_logistic)

smote.test$log.pred = predict(smote_logistic, smote.test[1:7], type="response")

table(smote.test$Transport,smote.test$log.pred>0.5)
smote.test$log.pred = ifelse(smote.test$log.pred>0.5 ,1,0)
smote.test$log.pred = as.factor(smote.test$log.pred)
confusionMatrix(smote.test$Transport,smote.test$log.pred, positive = '1')

```
> summary(smote_logistic)

Call:
glm(formula = Transport ~ ., family = binomial(link = "logit"),
    data = balanced.gd)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2618  -0.0289   0.0002   0.0777   2.8078

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -46.396718   2.263401 -20.499  < 2e-16 ***
Age           1.327173   0.068780  19.296  < 2e-16 ***
Gender1      -0.406815   0.230397  -1.766 0.077445 .
Engineer1    -0.203106   0.219868  -0.924 0.355611
MBA1         -0.802505   0.221506  -3.623 0.000291 ***
Salary       -0.009963   0.015779  -0.631 0.527784
Distance      0.413617   0.036793  11.242  < 2e-16 ***
license1      1.330047   0.198771   6.691 2.21e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6185.17  on 4461  degrees of freedom
Residual deviance:  813.36  on 4454  degrees of freedom
AIC: 829.36

Number of Fisher Scoring iterations: 9
```

Inference:

From the logistic regression output (screenshot above), we can conclude that Age, Distance, License and MBA are important variables.

```
> confusionMatrix(smote.test$Transport,smote.test$log.pred, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 93  3
         1  3 12

               Accuracy : 0.9459
                 95% CI : (0.8861, 0.9799)
    No Information Rate : 0.8649
    P-Value [Acc > NIR] : 0.004961

                  Kappa : 0.7687

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.8000
            Specificity : 0.9688
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.9688
             Prevalence : 0.1351
         Detection Rate : 0.1081
   Detection Prevalence : 0.1351
      Balanced Accuracy : 0.8844

       'Positive' Class : 1
```

KNN

```
### KNN
library(class)
knn.train = subset(data, split == TRUE)
knn.test = subset(data, split == FALSE)

for_out = vector()
k = c(1,3,5,6,9,10,11,14)
for (i in k) {
  knn_fit =  knn(train = balanced.gd[,1:7], test = knn.test[,1:7], cl= balanced.gd[,8],k =
i,prob=TRUE)
  for_out = cbind(for_out,sum(knn.test$Transport==1 & knn_fit==1))
}
for_out
```

```
> for_out
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]   14   14   14   14   13   13   13   13
```

Looking at the For loop output above we will keep the value of k = 3

knn_fit = knn(train = balanced.gd[,1:7], test = knn.test[,1:7], cl= balanced.gd[,8],k = 3,prob=TRUE)
table(knn.test[,8],knn_fit)
confusionMatrix(knn.test[,8],knn_fit, positive = '1')

```
  1  1 14
> confusionMatrix(knn.test[,8],knn_fit, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 93  3
         1  1 14

               Accuracy : 0.964
                 95% CI : (0.9103, 0.9901)
    No Information Rate : 0.8468
    P-Value [Acc > NIR] : 7.694e-05

                  Kappa : 0.854

 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.8235
            Specificity : 0.9894
         Pos Pred Value : 0.9333
         Neg Pred Value : 0.9687
             Prevalence : 0.1532
         Detection Rate : 0.1261
   Detection Prevalence : 0.1351
      Balanced Accuracy : 0.9064

       'Positive' Class : 1
```

Naive Bayes

nv.train = subset(data, split == TRUE)
nv.test = subset(data, split == FALSE)

nb_gd = naiveBayes(x=balanced.gd[,1:7], y=balanced.gd[,8])
nb_gd

pred_nb = predict(nb_gd,newdata = nv.test[,1:7])

table(nv.test[,8],pred_nb)
confusionMatrix(nv.test[,8],pred_nb, positive = '1')

> nb_gd

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = balanced.gd[, 1:7], y = balanced.gd[,

```
    8])

A-priori probabilities:
balanced.gd[, 8]
        0         1 
0.4948454 0.5051546 

Conditional probabilities:
                Age
balanced.gd[, 8]     [,1]     [,2]
               0 26.33062 2.859071
               1 35.32094 2.760111

                Gender
balanced.gd[, 8]         0         1
               0 0.2939312 0.7060688
               1 0.3141083 0.6858917

                Engineer
balanced.gd[, 8]         0         1
               0 0.2490942 0.7509058
               1 0.1796806 0.8203194

                MBA
balanced.gd[, 8]         0         1
               0 0.7576993 0.2423007
               1 0.6628217 0.3371783

                Salary
balanced.gd[, 8]    [,1]      [,2]
               0 12.7966  4.760753
               1 32.2631 11.021359

                Distance
balanced.gd[, 8]     [,1]     [,2]
               0 10.71363 3.211990
               1 15.41038 2.951743

                license
balanced.gd[, 8]         0         1
               0 0.8614130 0.1385870
               1 0.3811003 0.6188997
```

```
> confusionMatrix(nv.test[,8],pred_nb, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 92  4
         1  3 12

               Accuracy : 0.9369
                 95% CI : (0.8744, 0.9743)
    No Information Rate : 0.8559
    P-Value [Acc > NIR] : 0.006451

                  Kappa : 0.7376

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.7500
            Specificity : 0.9684
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.9583
             Prevalence : 0.1441
         Detection Rate : 0.1081
   Detection Prevalence : 0.1351
      Balanced Accuracy : 0.8592

       'Positive' Class : 1
```

Bagging

```
bag.train = subset(data, split == TRUE)
bag.test = subset(data, split == FALSE)

Bagging = bagging(Transport~., data=balanced.gd, control=rpart.control(maxdepth=5,
minsplit=4))

bag.test$pred.class = predict(Bagging, bag.test)

confusionMatrix(data=factor(bag.test$pred.class),reference=factor(bag.test$Transport),positive
='1')
table(bag.test$Transport,bag.test$pred.class)
```

```
> confusionMatrix(data=factor(bag.test$pred.class),reference=factor(bag.test$Transport),positive='1')
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 93  3
         1  3 12

               Accuracy : 0.9459
                 95% CI : (0.8861, 0.9799)
    No Information Rate : 0.8649
    P-Value [Acc > NIR] : 0.004961

                  Kappa : 0.7687

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.8000
            Specificity : 0.9688
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.9688
             Prevalence : 0.1351
         Detection Rate : 0.1081
   Detection Prevalence : 0.1351
      Balanced Accuracy : 0.8844

       'Positive' Class : 1
```

Boosting

```
boo.train = subset(data, split == TRUE)
boo.test = subset(data, split == FALSE)

features_train = data.matrix(balanced.gd[,1:7])
label_train = data.matrix(balanced.gd[,8])
features_test = data.matrix(boo.test[,1:7])

tp_xgb<-vector()
lr = c(0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1)
md = c(1,3,5,7,9,15, 18,20,25)
nr = c(2, 50, 100, 1000, 10000,200000)

for (i in md) {
  xgb.fit <- xgboost(
    data = features_train,
    label = label_train,
    eta = 0.5,
    max_depth = i,
    min_child_weight = 3,
    nrounds = 50,
    nfold = 5,
    objective = "binary:logistic",
    verbose = 0,
    early_stopping_rounds = 10
```

```
  )
  boo.test$xgb.pred.class = predict(xgb.fit, features_test)
  tp_xgb = cbind(tp_xgb,sum(boo.test$Transport==1 & boo.test$xgb.pred.class>=0.5))
}

tp_xgb
```

Using For loop the model is tuned by varying factors like eta, max_depth, nrounds.

Varying eta:

```
> tp_xgb
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   12   12   13   13   14   12   11
```

Varying nrounds:

```
> tp_xgb
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   14   14   14   14   14   14

> tp_xgb
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]   12   13   14   13   13   13   13   13   13
```

Varying min_chil_weight:

```
> tp_xgb
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]   12   13   11   12   13   11   13   12   14
```

```
### Tuned Boosting

xgb.fit <- xgboost(
  data = features_train,
  label = label_train,
  eta = 0.5,
  max_depth = 5,
  min_child_weight = 3,
  nrounds = 50,
  nfold = 5,
  objective = "binary:logistic",
  verbose = 0,
  early_stopping_rounds = 10
)
```

xgb.fit
summary(xgb.fit)
xgb.importance(model = xgb.fit)

```
> xgb.importance(model = xgb.fit)
    Feature         Gain       Cover  Frequency
1:      Age 0.9035528908 0.445610564 0.31914894
2:   Salary 0.0560887195 0.292336405 0.29787234
3: Distance 0.0327931758 0.222216223 0.27659574
4:  license 0.0069793433 0.032084971 0.08510638
5:      MBA 0.0005858706 0.007751837 0.02127660
```

Inference:

We can clearly see that Age, Salary and Distance are important features in out model.

boo.test$xgb.pred.class = predict(xgb.fit, features_test)
table(boo.test$Transport,boo.test$xgb.pred.class>=0.5)

boo.test$xgb.pred.class = ifelse(boo.test$xgb.pred.class>0.5 ,1,0)
boo.test$xgb.pred.class = as.factor(boo.test$xgb.pred.class)
confusionMatrix(boo.test$Transport,boo.test$xgb.pred.class, positive = '1')

```
> boo.test$xgb.pred.class = as.factor(boo.test$xgb.pred.class)
> confusionMatrix(boo.test$Transport,boo.test$xgb.pred.class, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 93  3
         1  1 14

               Accuracy : 0.964
                 95% CI : (0.9103, 0.9901)
    No Information Rate : 0.8468
    P-Value [Acc > NIR] : 7.694e-05

                  Kappa : 0.854

 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.8235
            Specificity : 0.9894
         Pos Pred Value : 0.9333
         Neg Pred Value : 0.9687
             Prevalence : 0.1532
         Detection Rate : 0.1261
   Detection Prevalence : 0.1351
      Balanced Accuracy : 0.9064

       'Positive' Class : 1
```

Conclusion and model comparison:

Model	Sensitivity	Specificity	Accuracy
Logistic Regression	80%	96.88%	94.59%
KNN	82.35%	98.94%	96.40%
Naive Bayes	75%	96.84%	93.69%
Bagging	80%	96.88%	94.59%
Boosting	82.35%	98.94%	96.40%

- After creating and comparing 5 models namely, Logistic Regression, KNN, Naive Bayes, Bagging and boosting. We conclude that KNN and Boosting give us best results.
- Naive Bayes, proves to be the worst predictor, Boosting and KNN give pretty similar results.
- Surprising results of KNN could be as the training data set (SMOTE Data) provided to the model, like every other model had minority at 50%. This might not be the case, when the minority is below 50%.
- As expected, boosting shows high Specificity and Accuracy, as it is an Ensemble method and I would recommend Boosting as the model to be used for future predictions. Which has an accuracy of 96.4%.

Conclusion:

- Important variables are Age, Distance, Salary and License.
- Employees travelling a distance more that 15 are more likely to use a Car.
- Probability of using a car increases with Age and Work Experience. We haven't used work experience in or model and age and work ex. Were highly correlated.
- As the value of Salary goes above 30, employees are more likely to use a car.