Auto-Insurance Loan Classification

Cleaning Data to omit null values, and turning data into factors

```
library('caret')
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
InsuranceClaims = read.csv("car_ic.csv")
InsuranceClaims = na.omit(InsuranceClaims)

VehicleType = rep(0, length(InsuranceClaims$VEHICLE_TYPE))
VehicleType[InsuranceClaims$VEHICLE_TYPE == "sports car"] = 1
InsuranceClaims$VEHICLE_TYPE = VehicleType

InsuranceClaims$OUTCOME = as.factor(InsuranceClaims$OUTCOME)
VehicleYear = rep(0, length(InsuranceClaims$VEHICLE_YEAR))
VehicleYear[InsuranceClaims$VEHICLE_YEAR == "after 2015"] = 1
InsuranceClaims$VEHICLE_YEAR = VehicleYear

Race = rep(0, length(InsuranceClaims$RACE))
Race[InsuranceClaims$RACE == "majority"] = 1
InsuranceClaims$RACE = Race

Gender = rep(0, length(InsuranceClaims$GENDER))
Gender[InsuranceClaims$GENDER == "male"] = 1
InsuranceClaims$GENDER = Gender

Age = rep(0, length(InsuranceClaims$AGE))
Age[InsuranceClaims$AGE == "16-25"] = 1
Age[InsuranceClaims$AGE == "26-39"] = 2
Age[InsuranceClaims$AGE == "40-64"] = 3
Age[InsuranceClaims$AGE == "65+"] = 4
InsuranceClaims$AGE = Age

Income = rep(0, length(InsuranceClaims$Income))
Income[InsuranceClaims$INCOME == "poverty"] = 1
Income[InsuranceClaims$INCOME == "working class"] = 2
Income[InsuranceClaims$INCOME == "middle class"] = 3
Income[InsuranceClaims$INCOME == "upper class"] = 4
InsuranceClaims$INCOME = Income

summary(InsuranceClaims)
```

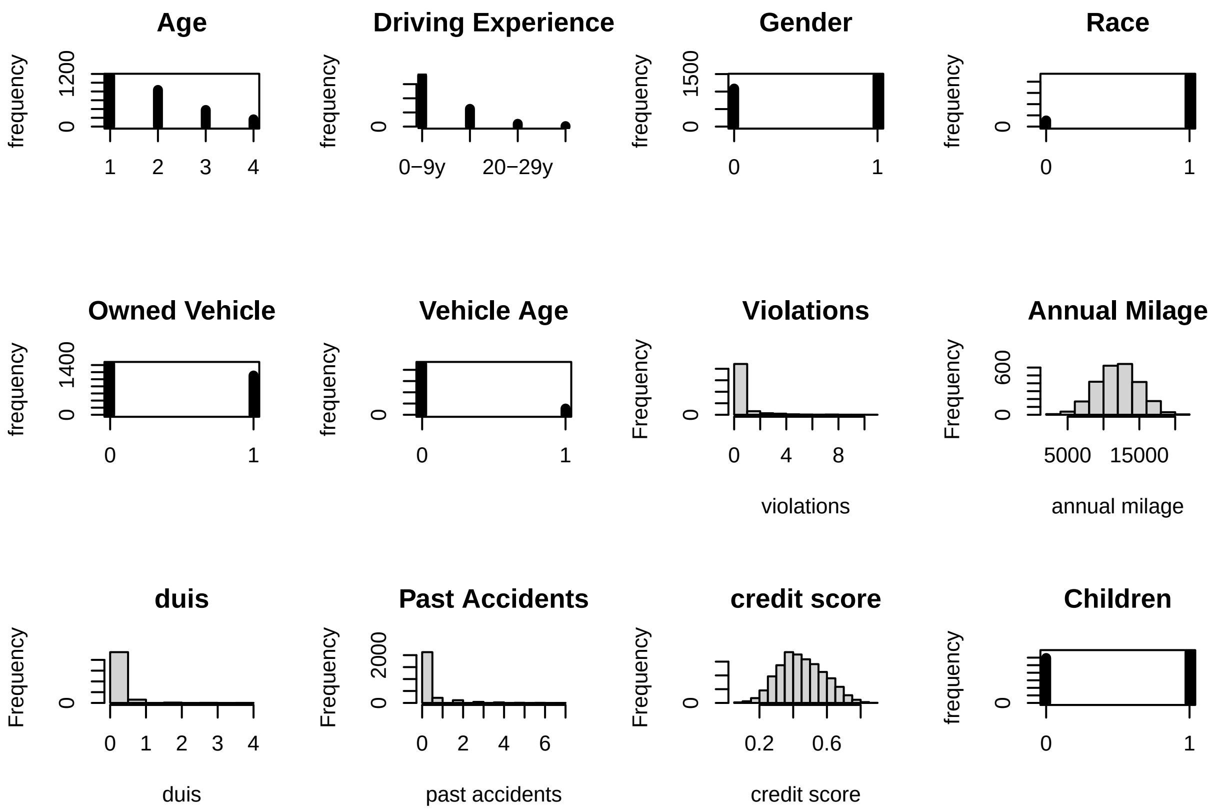
```
##        ID              AGE          GENDER            RACE
##  Min.   :   101   Min.   :1.0   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:247706   1st Qu.:2.0   1st Qu.:0.0000   1st Qu.:1.0000
##  Median :503269   Median :2.0   Median :0.0000   Median :1.0000
##  Mean   :501340   Mean   :2.5   Mean   :0.4988   Mean   :0.8986
##  3rd Qu.:756207   3rd Qu.:3.0   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :999976   Max.   :4.0   Max.   :1.0000   Max.   :1.0000
##  DRIVING_EXPERIENCE  EDUCATION            INCOME       CREDIT_SCORE
##  Length:8149        Length:8149        Min.   :1.000   Min.   :0.05336
##  Class :character   Class :character   1st Qu.:2.000   1st Qu.:0.41789
##  Mode  :character   Mode  :character   Median :3.000   Median :0.52676
##                                        Mean   :2.913   Mean   :0.51637
##                                        3rd Qu.:4.000   3rd Qu.:0.62007
##                                        Max.   :4.000   Max.   :0.96082
##  VEHICLE_OWNERSHIP  VEHICLE_YEAR       MARRIED         CHILDREN
##  Min.   :0.0000    Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000    1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :1.0000    Median :0.0000   Median :1.000   Median :1.0000
##  Mean   :0.6992    Mean   :0.3076   Mean   :0.501   Mean   :0.6893
##  3rd Qu.:1.0000    3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000
##  Max.   :1.0000    Max.   :1.0000   Max.   :1.000   Max.   :1.0000
##   POSTAL_CODE    ANNUAL_MILEAGE   VEHICLE_TYPE     SPEEDING_VIOLATIONS
##  Min.   :10238   Min.   : 2000   Min.   :0.00000   Min.   : 0.000
##  1st Qu.:10238   1st Qu.:10000   1st Qu.:0.00000   1st Qu.: 0.000
##  Median :10238   Median :12000   Median :0.00000   Median : 0.000
##  Mean   :19726   Mean   :11693   Mean   :0.04761   Mean   : 1.486
##  3rd Qu.:32765   3rd Qu.:14000   3rd Qu.:0.00000   3rd Qu.: 2.000
##  Max.   :92101   Max.   :22000   Max.   :1.00000   Max.   :22.000
##       DUIS        PAST_ACCIDENTS   OUTCOME
##  Min.   :0.0000   Min.   : 0.000   0:5613
##  1st Qu.:0.0000   1st Qu.: 0.000   1:2536
##  Median :0.0000   Median : 0.000
##  Mean   :0.2408   Mean   : 1.066
##  3rd Qu.:0.0000   3rd Qu.: 2.000
##  Max.   :6.0000   Max.   :15.000
```

Visualizations

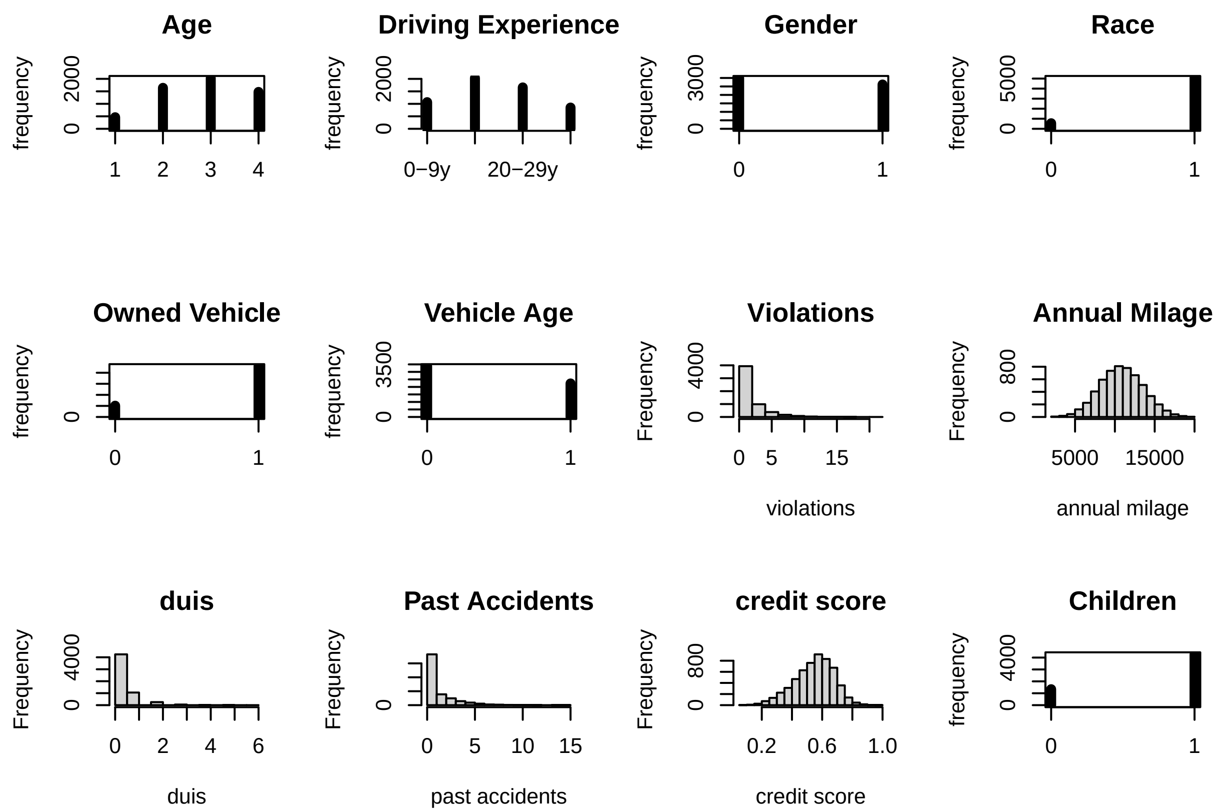
Split the data into two subsets based on outcome

```r
FiledClaim = subset(InsuranceClaims, InsuranceClaims$OUTCOME == 1)
DidNotFileClaim = subset(InsuranceClaims, InsuranceClaims$OUTCOME == 0)

#filed
par(mfrow=c(3, 4))
plot(x=table(FiledClaim$AGE), main ="Age", type = "h", lwd = 5, ylab="frequency")
plot(x=table(FiledClaim$DRIVING_EXPERIENCE), main ="Driving Experience", type = "h", lwd = 5, ylab="fre
plot(x=table(FiledClaim$GENDER), main ="Gender",type = "h", ylim = c(0, max(table(FiledClaim$GENDER))),
plot(x=table(FiledClaim$RACE), main ="Race",ylim = c(0, max(table(FiledClaim$RACE))), lwd = 5, ylab="fr
plot(x=table(FiledClaim$VEHICLE_OWNERSHIP), main ="Owned Vehicle",ylim = c(0, max(table(FiledClaim$VEHIC
plot(x=table(FiledClaim$VEHICLE_YEAR), main ="Vehicle Age", ylim = c(0, max(table(FiledClaim$VEHICLE_YE
hist(FiledClaim$SPEEDING_VIOLATIONS,  main ="Violations",xlab="violations")
```

```
hist(FiledClaim$ANNUAL_MILEAGE, xlab="annual milage", main ="Annual Milage")
hist(FiledClaim$DUIS, xlab = "duis", main="duis")
hist(FiledClaim$PAST_ACCIDENTS, xlab="past accidents", main ="Past Accidents")
hist(FiledClaim$CREDIT_SCORE, xlab="credit score", main ="credit score")
plot(x=table(FiledClaim$CHILDREN), main ="Children",ylim = c(0, max(table(FiledClaim$CHILDREN))), lwd =
```




```
#non filed
par(mfrow=c(3, 4))
plot(x=table(DidNotFileClaim$AGE), main ="Age", type = "h", lwd = 5, ylab="frequency")
plot(x=table(DidNotFileClaim$DRIVING_EXPERIENCE), main ="Driving Experience", type = "h", lwd = 5, ylab=
plot(x=table(DidNotFileClaim$GENDER), main ="Gender",type = "h", ylim = c(0, max(table(DidNotFileClaim$G
plot(x=table(DidNotFileClaim$RACE), main ="Race",ylim = c(0, max(table(DidNotFileClaim$RACE))), lwd = 5
plot(x=table(DidNotFileClaim$VEHICLE_OWNERSHIP), main ="Owned Vehicle",ylim = c(0, max(table(DidNotFile
plot(x=table(DidNotFileClaim$VEHICLE_YEAR), main ="Vehicle Age", ylim = c(0, max(table(DidNotFileClaim$V
hist(DidNotFileClaim$SPEEDING_VIOLATIONS,  main ="Violations",xlab="violations")
hist(DidNotFileClaim$ANNUAL_MILEAGE, xlab="annual milage", main ="Annual Milage")
hist(DidNotFileClaim$DUIS, xlab = "duis", main="duis")
hist(DidNotFileClaim$PAST_ACCIDENTS, xlab="past accidents", main ="Past Accidents")
hist(DidNotFileClaim$CREDIT_SCORE, xlab="credit score", main ="credit score")
plot(x=table(DidNotFileClaim$CHILDREN), main ="Children",ylim = c(0, max(table(DidNotFileClaim$CHILDREN)
```



Logistic Regression

```
InsuranceClaims = subset(InsuranceClaims, select = -c(ID))
trainingSet = sample(dim(InsuranceClaims)[1], dim(InsuranceClaims)[1] * 0.7)
logistic = glm(OUTCOME ~ ., data = InsuranceClaims, family = "binomial", subset = trainingSet)
summary(logistic)
```

```
##
## Call:
## glm(formula = OUTCOME ~ ., family = "binomial", data = InsuranceClaims,
##     subset = trainingSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9881  -0.5277  -0.1953   0.4708   3.4444
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 4.797e-01  3.359e-01   1.428  0.15322
## AGE                         2.774e-03  6.055e-02   0.046  0.96346
## GENDER                      1.006e+00  8.593e-02  11.704  < 2e-16 ***
## RACE                        1.171e-01  1.244e-01   0.941  0.34662
## DRIVING_EXPERIENCE10-19y -1.993e+00  1.122e-01 -17.759  < 2e-16 ***
## DRIVING_EXPERIENCE20-29y -3.516e+00  2.120e-01 -16.588  < 2e-16 ***
## DRIVING_EXPERIENCE30y+    -4.296e+00  4.068e-01 -10.560  < 2e-16 ***
## EDUCATIONnone              -1.754e-02  1.089e-01  -0.161  0.87198
```

```
## EDUCATIONuniversity      -2.330e-02  9.938e-02  -0.234  0.81467
## INCOME                   -8.319e-02  6.579e-02  -1.265  0.20604
## CREDIT_SCORE              6.468e-01  4.325e-01   1.495  0.13483
## VEHICLE_OWNERSHIP        -1.782e+00  9.066e-02 -19.661  < 2e-16 ***
## VEHICLE_YEAR             -1.690e+00  1.082e-01 -15.611  < 2e-16 ***
## MARRIED                  -3.561e-01  9.392e-02  -3.792  0.00015 ***
## CHILDREN                 -1.524e-01  9.455e-02  -1.612  0.10704
## POSTAL_CODE               2.078e-05  2.237e-06   9.291  < 2e-16 ***
## ANNUAL_MILEAGE            7.200e-05  1.802e-05   3.995 6.48e-05 ***
## VEHICLE_TYPE              3.050e-02  1.837e-01   0.166  0.86815
## SPEEDING_VIOLATIONS       3.215e-02  3.348e-02   0.960  0.33689
## DUIS                      1.487e-01  9.780e-02   1.520  0.12850
## PAST_ACCIDENTS           -1.341e-01  4.675e-02  -2.869  0.00412 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7103.4  on 5703  degrees of freedom
## Residual deviance: 4073.0  on 5683  degrees of freedom
## AIC: 4115
##
## Number of Fisher Scoring iterations: 6
```

```
pVals = predict(logistic, newdata = InsuranceClaims[-trainingSet, ], type="response")
predictions = rep(0, length(pVals))
predictions[pVals > 0.5] = 1
paste("Testing Error: ", toString(mean(predictions != InsuranceClaims[-trainingSet, ]$OUTCOME)))
```

```
## [1] "Testing Error:  0.154192229038855"
```

```
confusionMatrix(data = factor(predictions), reference = factor(InsuranceClaims[-trainingSet, ]$OUTCOME))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1521  195
##          1  182  547
##
##                Accuracy : 0.8458
##                  95% CI : (0.8309, 0.8599)
##     No Information Rate : 0.6965
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6335
##
##  Mcnemar's Test P-Value : 0.5366
##
##             Sensitivity : 0.8931
##             Specificity : 0.7372
##          Pos Pred Value : 0.8864
##          Neg Pred Value : 0.7503
```

```
##               Prevalence : 0.6965
##           Detection Rate : 0.6221
##     Detection Prevalence : 0.7018
##        Balanced Accuracy : 0.8152
##
##         'Positive' Class : 0
##
```

KNN

```
library(class)
#remove outcome variable
icKNN = InsuranceClaims[c('CREDIT_SCORE', 'AGE','INCOME', 'GENDER', 'VEHICLE_OWNERSHIP','MARRIED','CHIL

#normalize numeric variables
normalize = function(x) {
  return (x-min(x))/(max(x) - min(x))
}
InsuranceClaimsNormal = as.data.frame(lapply(icKNN[,1:12], normalize))

#test and train subsets
dataPoints = sample(1:nrow(InsuranceClaimsNormal), size = nrow(InsuranceClaimsNormal)*0.7, replace = FA
training = icKNN[dataPoints, ]
testing = icKNN[-dataPoints, ]
trainingOutcome = icKNN[dataPoints, 13]
testingOutcome = icKNN[-dataPoints, 13]

#knn
error =  100
bestK = 0
bestKNN = 0
for(j in 1:25) {
  for(i in 1:25) {
    knnVal =  knn(train = training, test = testing, cl = trainingOutcome, k = i)
    testError = 1 - sum(testingOutcome == knnVal) / NROW(testingOutcome)
    if(testError < error) {
      error = testError
      bestK = i
      bestKNN = knnVal
    }
  }
}
print(bestK)
```

```
## [1] 5
```

```
paste("Test error: ", toString(error))
```

```
## [1] "Test error:  0.056441717791411"
```

```
confusionMatrix(data = factor(bestKNN), factor(testingOutcome))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1670  114
##          1   24  637
##
##                Accuracy : 0.9436
##                  95% CI : (0.9337, 0.9524)
##     No Information Rate : 0.6928
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8628
##
##  Mcnemar's Test P-Value : 3.559e-14
##
##             Sensitivity : 0.9858
##             Specificity : 0.8482
##          Pos Pred Value : 0.9361
##          Neg Pred Value : 0.9637
##              Prevalence : 0.6928
##          Detection Rate : 0.6830
##    Detection Prevalence : 0.7297
##       Balanced Accuracy : 0.9170
##
##        'Positive' Class : 0
##
```

Random Forest

```
InsuranceClaims = read.csv("car_ic.csv")
InsuranceClaims = na.omit(InsuranceClaims)

InsuranceClaims$OUTCOME = as.factor(InsuranceClaims$OUTCOME)
InsuranceClaims$VEHICLE_OWNERSHIP = as.factor(InsuranceClaims$VEHICLE_OWNERSHIP)
InsuranceClaims$CHILDREN = as.factor(InsuranceClaims$CHILDREN)
InsuranceClaims$MARRIED = as.factor(InsuranceClaims$MARRIED)
InsuranceClaims = subset(InsuranceClaims, select = -c(ID))
library(randomForest)
```

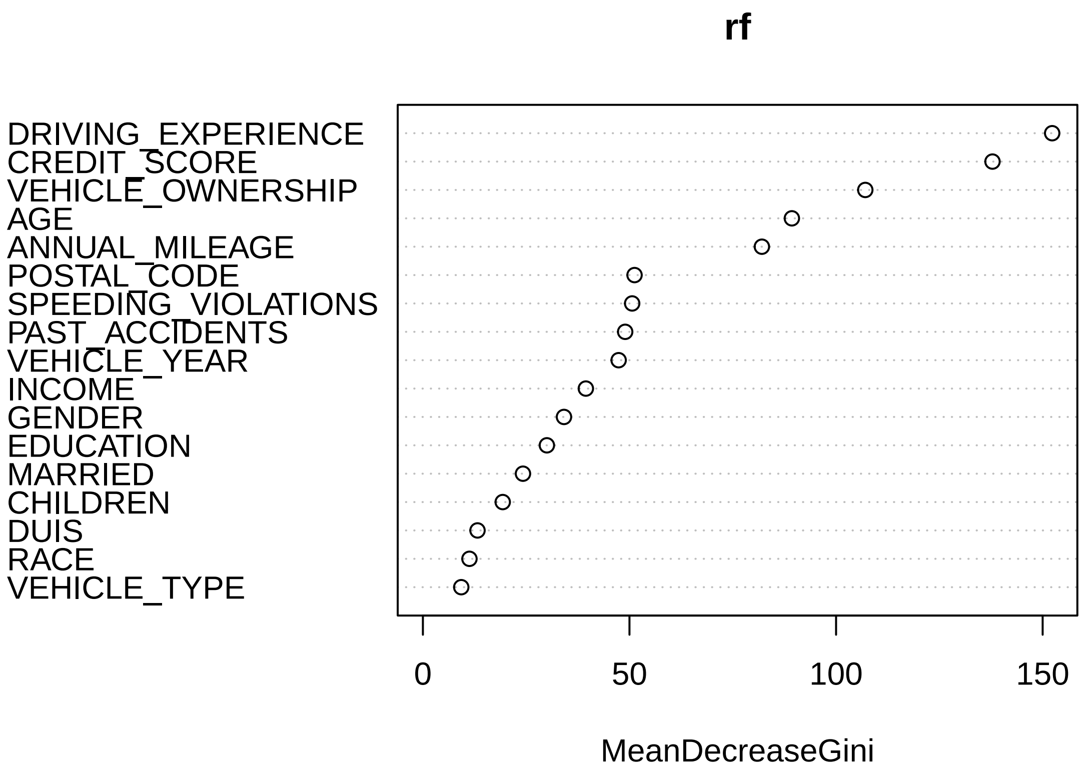
```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
#training and testing set
trainingSet = sample(dim(InsuranceClaims)[1], dim(InsuranceClaims)[1] * 0.7)
training = InsuranceClaims[trainingSet, ]
testing = InsuranceClaims[-trainingSet, ]

#random forest
rf = randomForest(OUTCOME ~ . , data = testing, ntree = 500)
varImpPlot(rf)
```




```r
print(rf)
```

```
##
## Call:
##  randomForest(formula = OUTCOME ~ ., data = testing, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 16.56%
## Confusion matrix:
##      0   1 class.error
## 0 1518 174   0.1028369
## 1  231 522   0.3067729
```

```r
#prediction
prediction = predict(rf, newdata = testing)
paste("test error: ", toString(1 - mean(prediction == testing$OUTCOME)))
```

```
## [1] "test error:  0.00858895705521467"
```

```r
confusionMatrix(data = factor(prediction), factor(testing$OUTCOME))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1672    1
##          1   20  752
##
##                Accuracy : 0.9914
##                  95% CI : (0.9869, 0.9947)
##     No Information Rate : 0.692
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.98
##
##  Mcnemar's Test P-Value : 8.568e-05
##
##             Sensitivity : 0.9882
##             Specificity : 0.9987
##          Pos Pred Value : 0.9994
##          Neg Pred Value : 0.9741
##              Prevalence : 0.6920
##          Detection Rate : 0.6838
##    Detection Prevalence : 0.6843
##       Balanced Accuracy : 0.9934
##
##        'Positive' Class : 0
##
```

QDA

```r
attach(InsuranceClaims,warn.conflicts = FALSE)
library(MASS)
trainingSet = sample(dim(InsuranceClaims)[1], dim(InsuranceClaims)[1] * 0.7)
training = InsuranceClaims[trainingSet, ]
testing = InsuranceClaims[-trainingSet, ]
# Fitting and prediction
qda_fit = qda(OUTCOME~., data=training)
print(qda_fit)
```

```
## Call:
## qda(OUTCOME ~ ., data = training)
##
## Prior probabilities of groups:
```

```
##         0         1 
## 0.6919705 0.3080295 
## 
## Group means:
##    AGE26-39  AGE40-64    AGE65+ GENDERmale RACEminority
## 0 0.2956676 0.3602736 0.2634913  0.4654168   0.09982265
## 1 0.3386454 0.1491178 0.0580535  0.5520774   0.10756972
##   DRIVING_EXPERIENCE10-19y DRIVING_EXPERIENCE20-29y DRIVING_EXPERIENCE30y+
## 0                0.3617938               0.29592095             0.15201419
## 1                0.2481503               0.03756403             0.00569152
##   EDUCATIONnone EDUCATIONuniversity INCOMEpoverty INCOMEupper class
## 0     0.1482138           0.4438814    0.08994173         0.5558652
## 1     0.2840068           0.2794536    0.36653386         0.1861127
##   INCOMEworking class CREDIT_SCORE VEHICLE_OWNERSHIP1 VEHICLE_YEARbefore 2015
## 0           0.1317456    0.5479556          0.8236635               0.6024829
## 1           0.2544109    0.4504384          0.4308480               0.8912920
##    MARRIED1 CHILDREN1 POSTAL_CODE ANNUAL_MILEAGE VEHICLE_TYPEsports car
## 0 0.5872815 0.7638713    18594.98       11338.74             0.04965797
## 1 0.3022197 0.5281730    22013.58       12527.60             0.04667046
##   SPEEDING_VIOLATIONS       DUIS PAST_ACCIDENTS
## 0            1.917912 0.31010894      1.4068913
## 1            0.483210 0.07968127      0.2743312
```

```
summary(qda_fit)
```

```
##         Length Class  Mode     
## prior      2   -none- numeric  
## counts     2   -none- numeric  
## means     48   -none- numeric  
## scaling 1152   -none- numeric  
## ldet       2   -none- numeric  
## lev        2   -none- character
## N          1   -none- numeric  
## call       3   -none- call     
## terms      3   terms  call     
## xlevels   11   -none- list
```

```
qda_predictions = predict(qda_fit, newdata=testing)
predictions <- as.data.frame(lapply(qda_predictions, unlist))
confusionMatrix(data = factor(predictions$class), factor(testing$OUTCOME))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1183  107
##          1  483  672
##                                           
##                Accuracy : 0.7587          
##                  95% CI : (0.7412, 0.7755)
##     No Information Rate : 0.6814          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
```

```
##                  Kappa : 0.5075
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##            Sensitivity : 0.7101
##            Specificity : 0.8626
##         Pos Pred Value : 0.9171
##         Neg Pred Value : 0.5818
##             Prevalence : 0.6814
##         Detection Rate : 0.4838
##   Detection Prevalence : 0.5276
##      Balanced Accuracy : 0.7864
##
##       'Positive' Class : 0
##
```

LDA

```
attach(InsuranceClaims,warn.conflicts = FALSE)
library(MASS)
trainingSet = sample(dim(InsuranceClaims)[1], dim(InsuranceClaims)[1] * 0.7)
training = InsuranceClaims[trainingSet, ]
testing = InsuranceClaims[-trainingSet, ]
# Fit data and predict
lda_fit = lda(OUTCOME~., data=training)
lda_fit
```

```
## Call:
## lda(OUTCOME ~ ., data = training)
##
## Prior probabilities of groups:
##         0         1
## 0.6882889 0.3117111
##
## Group means:
##    AGE26-39  AGE40-64     AGE65+ GENDERmale RACEminority
## 0 0.2919002 0.3565970 0.26770250  0.4610290   0.09780948
## 1 0.3239595 0.1552306 0.06186727  0.5663667   0.11192351
##   DRIVING_EXPERIENCE10-19y DRIVING_EXPERIENCE20-29y DRIVING_EXPERIENCE30y+
## 0                0.3627101               0.29139073            0.155374427
## 1                0.2429696               0.04274466            0.005624297
##   EDUCATIONnone EDUCATIONuniversity INCOMEpoverty INCOMEupper class
## 0     0.1505349           0.4421803    0.09679063         0.5476312
## 1     0.2710911           0.2896513    0.35489314         0.1917885
##   INCOMEworking class CREDIT_SCORE VEHICLE_OWNERSHIP1 VEHICLE_YEARbefore 2015
## 0           0.1309221    0.5453444          0.8125318              0.6023943
## 1           0.2570304    0.4512775          0.4465692              0.8993251
##    MARRIED1 CHILDREN1 POSTAL_CODE ANNUAL_MILEAGE VEHICLE_TYPEsports car
## 0 0.5858380 0.7600611    18458.17       11358.63             0.04941416
## 1 0.3166479 0.5343082    22401.08       12482.56             0.05174353
##   SPEEDING_VIOLATIONS       DUIS PAST_ACCIDENTS
## 0           1.8851248 0.31533367      1.4215487
## 1           0.5359955 0.08773903      0.2992126
```

```
##
## Coefficients of linear discriminants:
##                                   LD1
## AGE26-39                 -2.267838e-01
## AGE40-64                 -3.441987e-01
## AGE65+                   -3.416893e-01
## GENDERmale                4.787693e-01
## RACEminority             -1.942204e-02
## DRIVING_EXPERIENCE10-19y -1.360595e+00
## DRIVING_EXPERIENCE20-29y -1.733866e+00
## DRIVING_EXPERIENCE30y+   -1.696353e+00
## EDUCATIONnone            -3.349811e-02
## EDUCATIONuniversity       4.171899e-03
## INCOMEpoverty             1.458855e-01
## INCOMEupper class         5.160265e-02
## INCOMEworking class       2.000019e-01
## CREDIT_SCORE              8.031355e-02
## VEHICLE_OWNERSHIP1       -1.053010e+00
## VEHICLE_YEARbefore 2015   7.930410e-01
## MARRIED1                 -1.648882e-01
## CHILDREN1                -8.402989e-02
## POSTAL_CODE               1.138638e-05
## ANNUAL_MILEAGE            3.983389e-05
## VEHICLE_TYPEsports car    4.879767e-02
## SPEEDING_VIOLATIONS       7.488996e-03
## DUIS                     -8.620588e-03
## PAST_ACCIDENTS           -4.176870e-02
```

```
summary(lda_fit)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   48     -none- numeric
## scaling 24     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels 11     -none- list
```

```
lda_predictions = predict(lda_fit, newdata=testing)
predictions <- as.data.frame(lapply(lda_predictions, unlist))
confusionMatrix(data = factor(predictions$class), factor(testing$OUTCOME))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1501  204
##          1  186  554
##
```

```
##                Accuracy : 0.8405
##                  95% CI : (0.8254, 0.8548)
##     No Information Rate : 0.69
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6247
##
##  Mcnemar's Test P-Value : 0.3893
##
##             Sensitivity : 0.8897
##             Specificity : 0.7309
##          Pos Pred Value : 0.8804
##          Neg Pred Value : 0.7486
##              Prevalence : 0.6900
##          Detection Rate : 0.6139
##    Detection Prevalence : 0.6973
##       Balanced Accuracy : 0.8103
##
##        'Positive' Class : 0
##
```

Boosting

```
attach(InsuranceClaims)
```

```
## The following objects are masked from InsuranceClaims (pos = 3):
##
##     AGE, ANNUAL_MILEAGE, CHILDREN, CREDIT_SCORE, DRIVING_EXPERIENCE,
##     DUIS, EDUCATION, GENDER, INCOME, MARRIED, OUTCOME, PAST_ACCIDENTS,
##     POSTAL_CODE, RACE, SPEEDING_VIOLATIONS, VEHICLE_OWNERSHIP,
##     VEHICLE_TYPE, VEHICLE_YEAR

## The following objects are masked from InsuranceClaims (pos = 5):
##
##     AGE, ANNUAL_MILEAGE, CHILDREN, CREDIT_SCORE, DRIVING_EXPERIENCE,
##     DUIS, EDUCATION, GENDER, INCOME, MARRIED, OUTCOME, PAST_ACCIDENTS,
##     POSTAL_CODE, RACE, SPEEDING_VIOLATIONS, VEHICLE_OWNERSHIP,
##     VEHICLE_TYPE, VEHICLE_YEAR
```

```
InsuranceClaims = read.csv("car_ic.csv")
InsuranceClaims = na.omit(InsuranceClaims)

VehicleType = rep(0, length(InsuranceClaims$VEHICLE_TYPE))
VehicleType[InsuranceClaims$VEHICLE_TYPE == "sports car"] = 1
InsuranceClaims$VEHICLE_TYPE = VehicleType

InsuranceClaims$OUTCOME = as.factor(InsuranceClaims$OUTCOME)
VehicleYear = rep(0, length(InsuranceClaims$VEHICLE_YEAR))
VehicleYear[InsuranceClaims$VEHICLE_YEAR == "after 2015"] = 1
InsuranceClaims$VEHICLE_YEAR = VehicleYear

Race = rep(0, length(InsuranceClaims$RACE))
Race[InsuranceClaims$RACE == "majority"] = 1
```

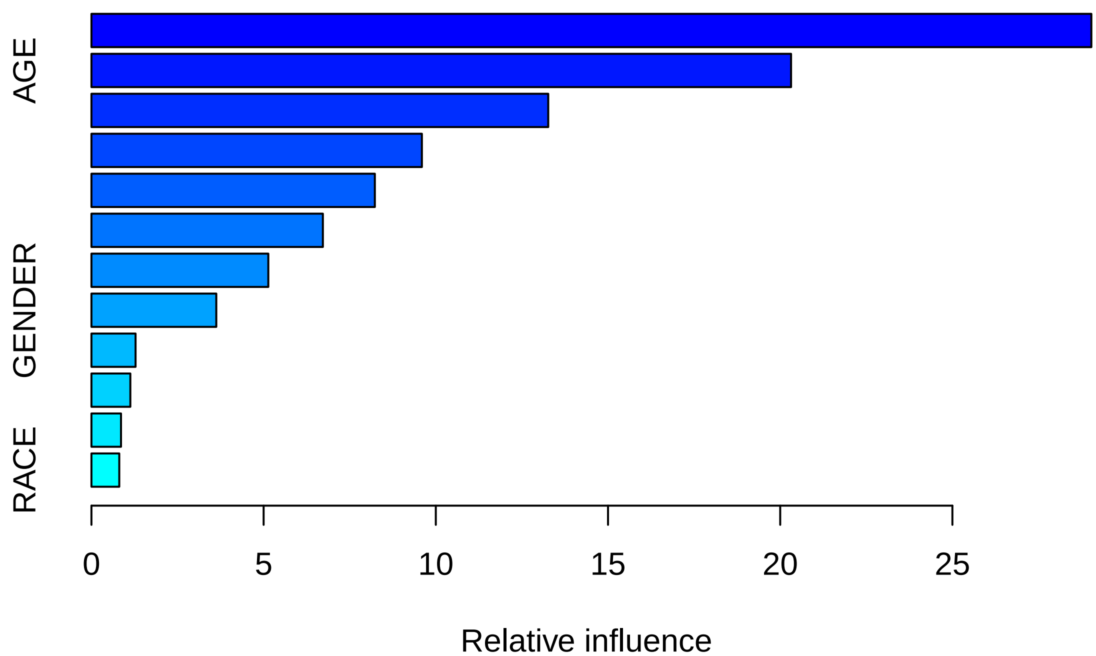
```
InsuranceClaims$RACE = Race

Gender = rep(0, length(InsuranceClaims$GENDER))
Gender[InsuranceClaims$GENDER == "male"] = 1
InsuranceClaims$GENDER = Gender

Age = rep(0, length(InsuranceClaims$AGE))
Age[InsuranceClaims$AGE == "16-25"] = 1
Age[InsuranceClaims$AGE == "26-39"] = 2
Age[InsuranceClaims$AGE == "40-64"] = 3
Age[InsuranceClaims$AGE == "65+"] = 4
InsuranceClaims$AGE = Age

Income = rep(0, length(InsuranceClaims$Income))
Income[InsuranceClaims$INCOME == "poverty"] = 1
Income[InsuranceClaims$INCOME == "working class"] = 2
Income[InsuranceClaims$INCOME == "middle class"] = 3
Income[InsuranceClaims$INCOME == "upper class"] = 4
InsuranceClaims$INCOME = Income
InsuranceClaims$AGE = as.factor(InsuranceClaims$AGE)
InsuranceClaims$VEHICLE_TYPE = as.factor(InsuranceClaims$VEHICLE_TYPE)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
trainingSet = sample(dim(InsuranceClaims)[1], dim(InsuranceClaims)[1] * 0.7)
training = InsuranceClaims[trainingSet, c('CREDIT_SCORE', 'AGE','INCOME', 'GENDER', 'VEHICLE_OWNERSHIP'
testing = InsuranceClaims[-trainingSet, c('CREDIT_SCORE', 'AGE','INCOME', 'GENDER', 'VEHICLE_OWNERSHIP'
set.seed(1)
boosting.claims = gbm(as.integer(OUTCOME) - 1 ~., data=training,
                      distribution="bernoulli",n.trees=2500, cv.folds=3)
print(boosting.claims)
```

```
## gbm(formula = as.integer(OUTCOME) - 1 ~ ., distribution = "bernoulli",
##     data = training, n.trees = 2500, cv.folds = 3)
## A gradient boosted model with bernoulli loss function.
## 2500 iterations were performed.
## The best cross-validation iteration was 231.
## There were 12 predictors of which 12 had non-zero influence.
```

```
summary(boosting.claims)
```




```
##                                     var    rel.inf
## CREDIT_SCORE               CREDIT_SCORE 29.0350801
## AGE                                 AGE 20.3153033
## VEHICLE_OWNERSHIP     VEHICLE_OWNERSHIP 13.2641504
## PAST_ACCIDENTS           PAST_ACCIDENTS  9.5953358
## ANNUAL_MILEAGE           ANNUAL_MILEAGE  8.2293298
## SPEEDING_VIOLATIONS SPEEDING_VIOLATIONS  6.7203926
## INCOME                           INCOME  5.1360001
## GENDER                           GENDER  3.6257085
## CHILDREN                       CHILDREN  1.2811421
## MARRIED                         MARRIED  1.1298563
## DUIS                               DUIS  0.8590180
## RACE                               RACE  0.8086832
```

```
predict.trees.gbm = predict.gbm(boosting.claims, newdata=testing, n.trees=2500)
predictions = predict(boosting.claims, newdata=testing )
```

```
## Using 231 trees...
```

```
prediction_classifier = vector()
for (i in 1:length(predictions)) {
  if(predictions[[i]]>=0) {
    prediction_classifier = append(prediction_classifier, 1)
  }
```

```r
  else {
    prediction_classifier = append(prediction_classifier, 0)
  }
}
confusionMatrix(data = factor(prediction_classifier), factor(testing$OUTCOME))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1529  277
##          1  179  460
##
##                Accuracy : 0.8135
##                  95% CI : (0.7975, 0.8288)
##     No Information Rate : 0.6986
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5398
##
##  Mcnemar's Test P-Value : 5.561e-06
##
##             Sensitivity : 0.8952
##             Specificity : 0.6242
##          Pos Pred Value : 0.8466
##          Neg Pred Value : 0.7199
##              Prevalence : 0.6986
##          Detection Rate : 0.6254
##    Detection Prevalence : 0.7387
##       Balanced Accuracy : 0.7597
##
##        'Positive' Class : 0
##
```

Classification Trees

```r
library(tree)
training = InsuranceClaims[trainingSet, ]
testing = InsuranceClaims[-trainingSet, ]
tree.claims = tree(OUTCOME~., data=training)
```

```
## Warning in tree(OUTCOME ~ ., data = training): NAs introduced by coercion
```

```r
tree.claims
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 5704 7111.0 0 ( 0.68461 0.31539 )
##    2) AGE: 2,3,4 4581 4774.0 0 ( 0.78454 0.21546 )
##      4) VEHICLE_OWNERSHIP < 0.5 1219 1660.0 0 ( 0.57752 0.42248 )
##        8) AGE: 3,4 587  673.6 0 ( 0.73935 0.26065 )
```

```
##         16) SPEEDING_VIOLATIONS < 0.5 204  282.7 1 ( 0.49020 0.50980 ) *
##         17) SPEEDING_VIOLATIONS > 0.5 383  293.0 0 ( 0.87206 0.12794 ) *
##        9) AGE: 2 632  862.7 1 ( 0.42722 0.57278 ) *
##      5) VEHICLE_OWNERSHIP > 0.5 3362 2728.0 0 ( 0.85961 0.14039 )
##       10) PAST_ACCIDENTS < 0.5 1455 1610.0 0 ( 0.75808 0.24192 )
##         20) VEHICLE_YEAR < 0.5 914 1164.0 0 ( 0.66630 0.33370 ) *
##         21) VEHICLE_YEAR > 0.5 541  319.5 0 ( 0.91312 0.08688 ) *
##       11) PAST_ACCIDENTS > 0.5 1907  896.1 0 ( 0.93707 0.06293 )
##         22) POSTAL_CODE < 15727.5 1396  384.0 0 ( 0.96920 0.03080 ) *
##         23) POSTAL_CODE > 15727.5 511  433.2 0 ( 0.84932 0.15068 )
##           46) POSTAL_CODE < 26991 22    0.0 1 ( 0.00000 1.00000 ) *
##           47) POSTAL_CODE > 26991 489  343.9 0 ( 0.88753 0.11247 ) *
##    3) AGE: 1 1123 1325.0 1 ( 0.27694 0.72306 )
##      6) VEHICLE_OWNERSHIP < 0.5 566  387.0 1 ( 0.10777 0.89223 ) *
##      7) VEHICLE_OWNERSHIP > 0.5 557  766.3 1 ( 0.44883 0.55117 ) *
```

```
cv.tree.claims = cv.tree(tree.claims, FUN=prune.misclass)
```

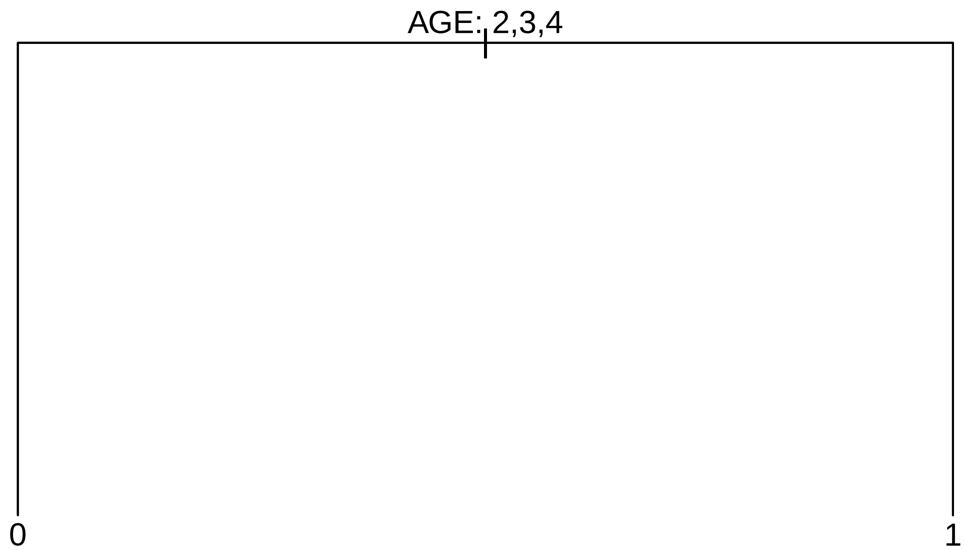
```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
cv.tree.claims
```

```
## $size
## [1] 10  8  7  4  2  1
##
## $dev
## [1] 1225 1233 1222 1243 1299 1799
##
## $k
## [1]       -Inf   0.000000   4.000000   7.333333  46.000000 501.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
prune.claims <- prune.misclass(tree.claims , best = 2)
plot(prune.claims)
text(prune.claims, pretty =0)
```




```r
predictions = predict(tree.claims, testing, type="class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```r
predictions.cv = predict(prune.claims, testing, type="class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```r
confusionMatrix(data = factor(predictions), factor(testing$OUTCOME))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1429  207
##          1  279  530
##
##                Accuracy : 0.8012
##                  95% CI : (0.7848, 0.8169)
##     No Information Rate : 0.6986
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5408
##
```

```
##  Mcnemar's Test P-Value : 0.001279
##
##             Sensitivity : 0.8367
##             Specificity : 0.7191
##          Pos Pred Value : 0.8735
##          Neg Pred Value : 0.6551
##              Prevalence : 0.6986
##          Detection Rate : 0.5845
##    Detection Prevalence : 0.6691
##       Balanced Accuracy : 0.7779
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(data=factor(predictions.cv), factor(testing$OUTCOME))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1556  391
##          1  152  346
##
##                Accuracy : 0.7779
##                  95% CI : (0.7609, 0.7943)
##     No Information Rate : 0.6986
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4191
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9110
##             Specificity : 0.4695
##          Pos Pred Value : 0.7992
##          Neg Pred Value : 0.6948
##              Prevalence : 0.6986
##          Detection Rate : 0.6364
##    Detection Prevalence : 0.7963
##       Balanced Accuracy : 0.6902
##
##        'Positive' Class : 0
##
```

```
cv.tree.claims = cv.tree(tree.claims, FUN=prune.misclass)
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

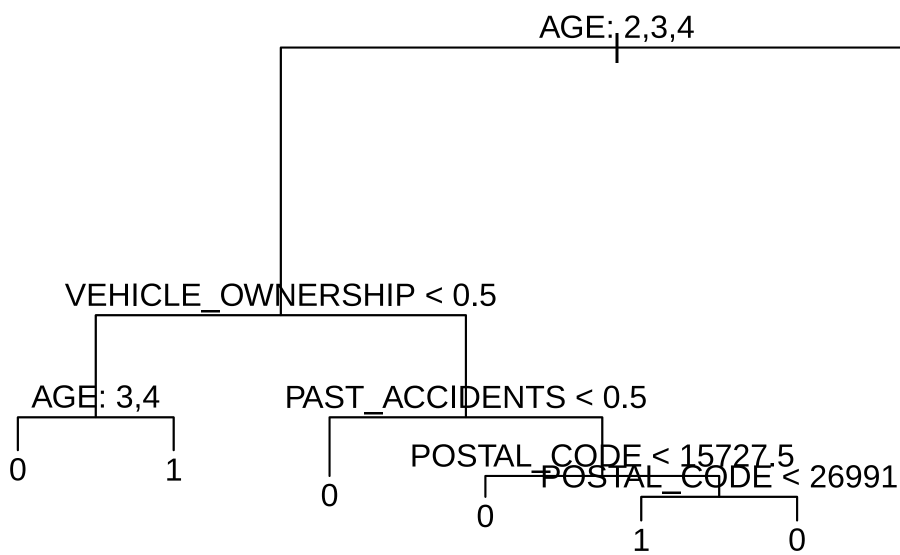
```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
cv.tree.claims
```

```
## $size
## [1] 10  8  7  4  2  1
##
## $dev
## [1] 1236 1236 1238 1247 1306 1799
```

```
##
## $k
## [1]       -Inf   0.000000   4.000000   7.333333  46.000000 501.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
prune.claims <- prune.misclass(tree.claims , best = 6)
plot(prune.claims)
text(prune.claims, pretty =0)
```




```r
predictions.cv = predict(prune.claims, testing, type="class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```r
confusionMatrix(data=factor(predictions.cv), factor(testing$OUTCOME))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
```

```
##          0 1459  243
##          1  249  494
##
##                Accuracy : 0.7988
##                  95% CI : (0.7823, 0.8145)
##     No Information Rate : 0.6986
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5233
##
##  Mcnemar's Test P-Value : 0.8217
##
##             Sensitivity : 0.8542
##             Specificity : 0.6703
##          Pos Pred Value : 0.8572
##          Neg Pred Value : 0.6649
##              Prevalence : 0.6986
##          Detection Rate : 0.5967
##    Detection Prevalence : 0.6961
##       Balanced Accuracy : 0.7623
##
##        'Positive' Class : 0
##
```

SVM

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(e1071)

#Grab features
col = c('AGE','CHILDREN','VEHICLE_TYPE','PAST_ACCIDENTS','SPEEDING_VIOLATIONS','DUIS', 'OUTCOME')
InsuranceClaims$OUTCOME = as.factor(InsuranceClaims$OUTCOME)

#training and testing set
trainingSet = sample(dim(InsuranceClaims)[1], dim(InsuranceClaims)[1] * 0.3)
training = InsuranceClaims[trainingSet, col]
testing = InsuranceClaims[-trainingSet, col]

#tune
tuned = tune(svm, OUTCOME ~ ., data = training, ranges = list(epsilon = seq(0,1, 0.1), cost =  2^(2:7)))
summary(tuned)
```

```
##
## Parameter tuning of 'svm':
```

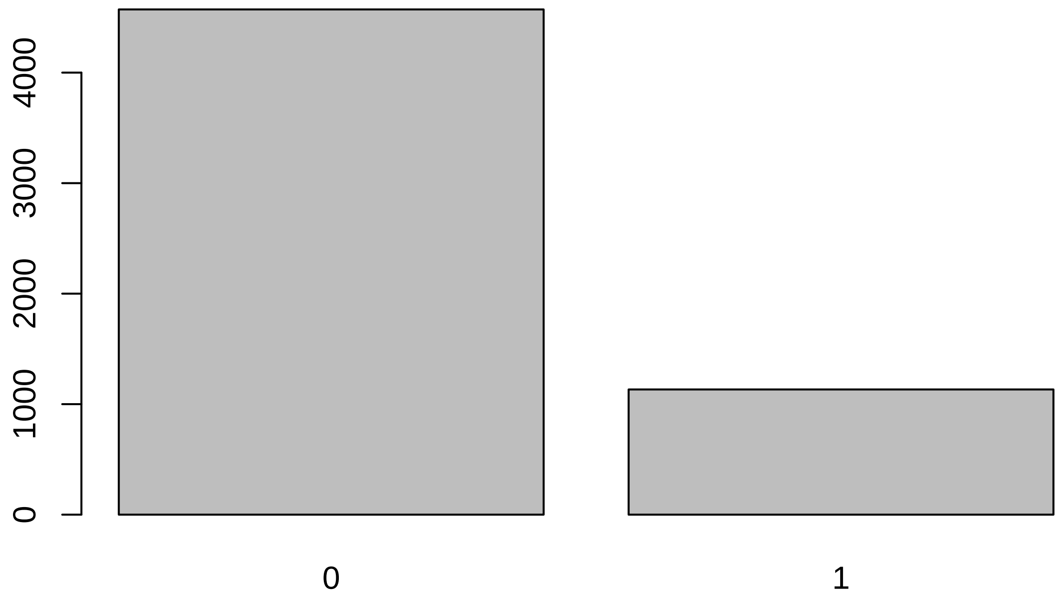
```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  epsilon cost
##        0    4
##
## - best performance: 0.2356892
##
## - Detailed performance results:
##    epsilon cost     error dispersion
## 1      0.0    4 0.2356892 0.02328011
## 2      0.1    4 0.2356892 0.02328011
## 3      0.2    4 0.2356892 0.02328011
## 4      0.3    4 0.2356892 0.02328011
## 5      0.4    4 0.2356892 0.02328011
## 6      0.5    4 0.2356892 0.02328011
## 7      0.6    4 0.2356892 0.02328011
## 8      0.7    4 0.2356892 0.02328011
## 9      0.8    4 0.2356892 0.02328011
## 10     0.9    4 0.2356892 0.02328011
## 11     1.0    4 0.2356892 0.02328011
## 12     0.0    8 0.2365105 0.02311990
## 13     0.1    8 0.2365105 0.02311990
## 14     0.2    8 0.2365105 0.02311990
## 15     0.3    8 0.2365105 0.02311990
## 16     0.4    8 0.2365105 0.02311990
## 17     0.5    8 0.2365105 0.02311990
## 18     0.6    8 0.2365105 0.02311990
## 19     0.7    8 0.2365105 0.02311990
## 20     0.8    8 0.2365105 0.02311990
## 21     0.9    8 0.2365105 0.02311990
## 22     1.0    8 0.2365105 0.02311990
## 23     0.0   16 0.2410054 0.02704238
## 24     0.1   16 0.2410054 0.02704238
## 25     0.2   16 0.2410054 0.02704238
## 26     0.3   16 0.2410054 0.02704238
## 27     0.4   16 0.2410054 0.02704238
## 28     0.5   16 0.2410054 0.02704238
## 29     0.6   16 0.2410054 0.02704238
## 30     0.7   16 0.2410054 0.02704238
## 31     0.8   16 0.2410054 0.02704238
## 32     0.9   16 0.2410054 0.02704238
## 33     1.0   16 0.2410054 0.02704238
## 34     0.0   32 0.2446889 0.02432108
## 35     0.1   32 0.2446889 0.02432108
## 36     0.2   32 0.2446889 0.02432108
## 37     0.3   32 0.2446889 0.02432108
## 38     0.4   32 0.2446889 0.02432108
## 39     0.5   32 0.2446889 0.02432108
## 40     0.6   32 0.2446889 0.02432108
## 41     0.7   32 0.2446889 0.02432108
## 42     0.8   32 0.2446889 0.02432108
## 43     0.9   32 0.2446889 0.02432108
```

```
## 44     1.0   32 0.2446889 0.02432108
## 45     0.0   64 0.2438675 0.02572766
## 46     0.1   64 0.2438675 0.02572766
## 47     0.2   64 0.2438675 0.02572766
## 48     0.3   64 0.2438675 0.02572766
## 49     0.4   64 0.2438675 0.02572766
## 50     0.5   64 0.2438675 0.02572766
## 51     0.6   64 0.2438675 0.02572766
## 52     0.7   64 0.2438675 0.02572766
## 53     0.8   64 0.2438675 0.02572766
## 54     0.9   64 0.2438675 0.02572766
## 55     1.0   64 0.2438675 0.02572766
## 56     0.0  128 0.2455002 0.02707454
## 57     0.1  128 0.2455002 0.02707454
## 58     0.2  128 0.2455002 0.02707454
## 59     0.3  128 0.2455002 0.02707454
## 60     0.4  128 0.2455002 0.02707454
## 61     0.5  128 0.2455002 0.02707454
## 62     0.6  128 0.2455002 0.02707454
## 63     0.7  128 0.2455002 0.02707454
## 64     0.8  128 0.2455002 0.02707454
## 65     0.9  128 0.2455002 0.02707454
## 66     1.0  128 0.2455002 0.02707454
```

```r
svmBest = tuned$best.model
summary(svmBest)
```

```
##
## Call:
## best.tune(method = svm, train.x = OUTCOME ~ ., data = training, ranges = list(epsilon = seq(0,
##     1, 0.1), cost = 2^(2:7)))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  4
##
## Number of Support Vectors:  1253
##
##  ( 671 582 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
#prediction
prediction = predict(svmBest, testing[, col], type = "class")
plot(prediction)
```




```
paste("test error: ", toString(1 - mean(prediction == testing[, 7])))
```

```
## [1] "test error:  0.222261174408414"
```

```
confusionMatrix(data = factor(prediction), factor(testing[, 7]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3627  945
##          1  323  810
##
##                Accuracy : 0.7777
##                  95% CI : (0.7667, 0.7885)
##     No Information Rate : 0.6924
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4212
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9182
##             Specificity : 0.4615
```

```
##          Pos Pred Value : 0.7933
##          Neg Pred Value : 0.7149
##              Prevalence : 0.6924
##          Detection Rate : 0.6358
##    Detection Prevalence : 0.8014
##       Balanced Accuracy : 0.6899
##
##        'Positive' Class : 0
##
```