FinalProjectCode

STAT515-005-Team 2: Preethal, Nivedita, Grace

2023-05-05

```
library(psych)
library("reshape2")
library(tidyverse)
library(caret)
library(randomForest)
library(ROCR)
library(Boruta)
library(rpart)
library(rattle)
library(tree)
library(corrplot)

InsData = read.csv("InsuranceDataTrain.csv")
InsDataTest = read.csv("InsuranceDataTest.csv")

# check the structure of the features
str(InsData)
```

```
## 'data.frame':    5822 obs. of  86 variables:
##  $ CustomerSubtype                  : int  33 37 37 9 40 23 39 33 33 11 ..
.
##  $ NbHouses                         : int  1 1 1 1 1 1 2 1 1 2 ...
##  $ AvgSizeHousehold                 : int  3 2 2 3 4 2 3 2 2 3 ...
##  $ AvgAge                           : int  2 2 2 3 2 1 2 3 4 3 ...
##  $ CustomerMainType                 : int  8 8 8 3 10 5 9 8 8 3 ...
##  $ RomanCatholic                    : int  0 1 0 2 1 0 2 0 0 3 ...
##  $ Protestant                       : int  5 4 4 3 4 5 2 7 1 5 ...
##  $ OtherReligion                    : int  1 1 2 2 1 0 0 0 3 0 ...
##  $ NoReligion                       : int  3 4 4 4 4 5 5 2 6 2 ...
##  $ Married                          : int  7 6 3 5 7 0 7 7 6 7 ...
##  $ LivingTogether                   : int  0 2 2 2 1 6 2 2 0 0 ...
##  $ OtherRelation                    : int  2 2 4 2 2 3 0 0 3 2 ...
##  $ Singles                          : int  1 0 4 2 2 3 0 0 3 2 ...
##  $ HouseholdNochildren              : int  2 4 4 3 4 5 3 5 3 2 ...
##  $ HouseholdWithchildren            : int  6 5 2 4 4 2 6 4 3 6 ...
##  $ HighLevelEducation               : int  1 0 0 3 5 0 0 0 0 0 ...
##  $ MediumLevelEducation             : int  2 5 5 4 4 5 4 3 1 4 ...
##  $ LowerLevelEducation              : int  7 4 4 2 0 4 5 6 8 5 ...
##  $ HighStatus                       : int  1 0 0 4 0 2 0 2 1 2 ...
##  $ Entrepreneur                     : int  0 0 0 0 5 0 0 0 1 0 ...
##  $ Farmer                           : int  1 0 0 0 4 0 0 0 0 0 ...
##  $ MiddleManagement                 : int  2 5 7 3 0 4 4 2 1 3 ...
```

```
##  $ SkilledLabourers                  : int  5 0 0 1 0 2 1 5 8 3 ...
##  $ UnskilledLabourers                : int  2 4 2 2 0 2 5 2 1 3 ...
##  $ SocialclassA                      : int  1 0 0 3 9 2 0 2 1 1 ...
##  $ SocialclassB1                     : int  1 2 5 2 0 2 1 1 1 2 ...
##  $ SocialclassB2                     : int  2 3 0 1 0 2 4 2 0 1 ...
##  $ SocialclassC                      : int  6 5 4 4 0 4 5 5 8 4 ...
##  $ SocialclassD                      : int  1 0 0 0 0 2 0 2 1 2 ...
##  $ RentedHouse                       : int  1 2 7 5 4 9 6 0 9 0 ...
##  $ HomeOwners                        : int  8 7 2 4 5 0 3 9 0 9 ...
##  $ X1car                             : int  8 7 7 9 6 5 8 4 5 6 ...
##  $ X2cars                            : int  0 1 0 0 2 3 0 4 2 1 ...
##  $ Nocar                             : int  1 2 2 0 1 3 1 2 3 2 ...
##  $ NationalHealthService             : int  8 6 9 7 5 9 9 6 7 6 ...
##  $ PrivateHealthInsurance            : int  1 3 0 2 4 0 0 3 2 3 ...
##  $ Income.30                         : int  0 2 4 1 0 5 4 2 7 2 ...
##  $ Income30.45.000                   : int  4 0 5 5 0 2 3 5 2 3 ...
##  $ Income45.75.000                   : int  5 5 0 3 9 3 3 3 1 3 ...
##  $ Income75.122.000                  : int  0 2 0 0 0 0 0 0 0 1 ...
##  $ Income.123.000                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AverageIncome                     : int  4 5 3 4 6 3 3 3 2 4 ...
##  $ PurchasingPowerClass              : int  3 4 4 4 3 3 5 3 3 7 ...
##  $ PrivateThirdPartyInsurance        : int  0 2 2 0 0 0 0 0 0 2 ...
##  $ ThirdPartyInsuranceFirms          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ThirdPartyInsuraneAgriculture     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ CarPolicies                       : int  6 0 6 6 0 6 6 0 5 0 ...
##  $ DeliveryVanPolicies               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MotorcycleScooterPolicies         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ LorryPolicies                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ TrailerPolicies                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ TractorPolicies                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AgriculturalMachinesPolicies      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MopedPolicies                     : int  0 0 0 0 0 0 0 3 0 0 ...
##  $ LifeInsurances                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PrivateAccidentPolicies           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FamilyAccidentPolicies            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ DisabilityInsurancePolicies       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FirePolicies                      : int  5 2 2 2 6 0 0 0 0 3 ...
##  $ SurfboardPolicies                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BoatPolicies                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BicyclePolicies                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PropertyInsurancePolicies         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SocialSecurityInsurancePolicies   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbPrivateThirdPartyInsurance      : int  0 2 1 0 0 0 0 0 0 1 ...
##  $ NbThirdPartyInsuranceFirms        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbThirdPartyInsuranceAgriculture : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbCarPolicies                     : int  1 0 1 1 0 1 1 0 1 0 ...
##  $ NbDeliveryVanPolicies             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbMotorcycleScooterPolicies       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbLorryPolicies                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbTrailerPolicies                 : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ NbTractorPolicies                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbAgriculturalMachinesPolicies     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbMopedPolicies                    : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ NbLifeInsurances                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbPrivateAccidentPolicies          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbFamilyAccidentsPolicies          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbDisabilityInsurancePolicies      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbFirePolicies                     : int  1 1 1 1 1 0 0 0 0 1 ...
##  $ NbSurfboardPolicies                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbBoatPolicies                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbBicyclePolicies                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbPropertyInsurancePolicies        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbSocialSecurityInsurancePolicies: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NbMobileHomePolicies               : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
# summarize columns to see possible values and observe if any scaling is need
ed
summary(InsData)
```

```
##  CustomerSubtype    NbHouses      AvgSizeHousehold     AvgAge
##  Min.   : 1.00   Min.   : 1.000   Min.   :1.000    Min.   :1.000
##  1st Qu.:10.00   1st Qu.: 1.000   1st Qu.:2.000    1st Qu.:2.000
##  Median :30.00   Median : 1.000   Median :3.000    Median :3.000
##  Mean   :24.25   Mean   : 1.111   Mean   :2.679    Mean   :2.991
##  3rd Qu.:35.00   3rd Qu.: 1.000   3rd Qu.:3.000    3rd Qu.:3.000
##  Max.   :41.00   Max.   :10.000   Max.   :5.000    Max.   :6.000
##  CustomerMainType RomanCatholic      Protestant    OtherReligion
##  Min.   : 1.000   Min.   :0.0000   Min.   :0.000   Min.   :0.00
##  1st Qu.: 3.000   1st Qu.:0.0000   1st Qu.:4.000   1st Qu.:0.00
##  Median : 7.000   Median :0.0000   Median :5.000   Median :1.00
##  Mean   : 5.774   Mean   :0.6965   Mean   :4.627   Mean   :1.07
##  3rd Qu.: 8.000   3rd Qu.:1.0000   3rd Qu.:6.000   3rd Qu.:2.00
##  Max.   :10.000   Max.   :9.0000   Max.   :9.000   Max.   :5.00
##    NoReligion       Married      LivingTogether   OtherRelation
##  Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   :0.00
##  1st Qu.:2.000   1st Qu.:5.000   1st Qu.:0.0000   1st Qu.:1.00
##  Median :3.000   Median :6.000   Median :1.0000   Median :2.00
##  Mean   :3.259   Mean   :6.183   Mean   :0.8835   Mean   :2.29
##  3rd Qu.:4.000   3rd Qu.:7.000   3rd Qu.:1.0000   3rd Qu.:3.00
##  Max.   :9.000   Max.   :9.000   Max.   :7.0000   Max.   :9.00
##     Singles      HouseholdNochildren HouseholdWithchildren HighLevelEducat
ion
##  Min.   :0.000   Min.   :0.00        Min.   :0.0           Min.   :0.000
##  1st Qu.:0.000   1st Qu.:2.00        1st Qu.:3.0           1st Qu.:0.000
##  Median :2.000   Median :3.00        Median :4.0           Median :1.000
##  Mean   :1.888   Mean   :3.23        Mean   :4.3           Mean   :1.461
##  3rd Qu.:3.000   3rd Qu.:4.00        3rd Qu.:6.0           3rd Qu.:2.000
##  Max.   :9.000   Max.   :9.00        Max.   :9.0           Max.   :9.000
##  MediumLevelEducation LowerLevelEducation   HighStatus      Entrepreneur
##  Min.   :0.000        Min.   :0.000       Min.   :0.000   Min.   :0.000
```

```
##  1st Qu.:2.000      1st Qu.:3.000      1st Qu.:0.000   1st Qu.:0.000  
##  Median :3.000      Median :5.000      Median :2.000   Median :0.000  
##  Mean   :3.351      Mean   :4.572      Mean   :1.895   Mean   :0.398  
##  3rd Qu.:4.000      3rd Qu.:6.000      3rd Qu.:3.000   3rd Qu.:1.000  
##  Max.   :9.000      Max.   :9.000      Max.   :9.000   Max.   :5.000  
##      Farmer      MiddleManagement SkilledLabourers UnskilledLabourers
##  Min.   :0.0000   Min.   :0.000    Min.   :0.00     Min.   :0.000    
##  1st Qu.:0.0000   1st Qu.:2.000    1st Qu.:1.00     1st Qu.:1.000    
##  Median :0.0000   Median :3.000    Median :2.00     Median :2.000    
##  Mean   :0.5223   Mean   :2.899    Mean   :2.22     Mean   :2.306    
##  3rd Qu.:1.0000   3rd Qu.:4.000    3rd Qu.:3.00     3rd Qu.:3.000    
##  Max.   :9.0000   Max.   :9.000    Max.   :9.00     Max.   :9.000    
##   SocialclassA   SocialclassB1   SocialclassB2    SocialclassC  
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:2.000  
##  Median :1.000   Median :2.000   Median :2.000   Median :4.000  
##  Mean   :1.621   Mean   :1.607   Mean   :2.203   Mean   :3.759  
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:5.000  
##  Max.   :9.000   Max.   :9.000   Max.   :9.000   Max.   :9.000  
##   SocialclassD    RentedHouse      HomeOwners        X1car          X2cars     
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.00   Min.   :0.
000  
##  1st Qu.:0.000   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:5.00   1st Qu.:0.
000  
##  Median :1.000   Median :4.000   Median :5.000   Median :6.00   Median :1.
000  
##  Mean   :1.067   Mean   :4.237   Mean   :4.772   Mean   :6.04   Mean   :1.
316  
##  3rd Qu.:2.000   3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:7.00   3rd Qu.:2.
000  
##  Max.   :9.000   Max.   :9.000   Max.   :9.000   Max.   :9.00   Max.   :7.
000  
##      Nocar       NationalHealthService PrivateHealthInsurance   Income.30   
##  Min.   :0.000   Min.   :0.000         Min.   :0.000          Min.   :0.00
0  
##  1st Qu.:1.000   1st Qu.:5.000         1st Qu.:1.000          1st Qu.:1.00
0  
##  Median :2.000   Median :7.000         Median :2.000          Median :2.00
0  
##  Mean   :1.959   Mean   :6.277         Mean   :2.729          Mean   :2.57
4  
##  3rd Qu.:3.000   3rd Qu.:8.000         3rd Qu.:4.000          3rd Qu.:4.00
0  
##  Max.   :9.000   Max.   :9.000         Max.   :9.000          Max.   :9.00
0  
##  Income30.45.000 Income45.75.000 Income75.122.000 Income.123.000  
##  Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :4.000   Median :3.000   Median :0.0000   Median :0.0000  
##  Mean   :3.536   Mean   :2.731   Mean   :0.7961   Mean   :0.2027  
```

```
##  3rd Qu.:5.000   3rd Qu.:4.000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :9.000   Max.   :9.000   Max.   :9.0000   Max.   :9.0000
##  AverageIncome   PurchasingPowerClass PrivateThirdPartyInsurance
##  Min.   :0.000   Min.   :1.000        Min.   :0.0000
##  1st Qu.:3.000   1st Qu.:3.000        1st Qu.:0.0000
##  Median :4.000   Median :4.000        Median :0.0000
##  Mean   :3.784   Mean   :4.236        Mean   :0.7712
##  3rd Qu.:4.000   3rd Qu.:6.000        3rd Qu.:2.0000
##  Max.   :9.000   Max.   :8.000        Max.   :3.0000
##  ThirdPartyInsuranceFirms ThirdPartyInsuraneAgriculture  CarPolicies
##  Min.   :0.00000          Min.   :0.00000               Min.   :0.00
##  1st Qu.:0.00000          1st Qu.:0.00000               1st Qu.:0.00
##  Median :0.00000          Median :0.00000               Median :5.00
##  Mean   :0.04002          Mean   :0.07162               Mean   :2.97
##  3rd Qu.:0.00000          3rd Qu.:0.00000               3rd Qu.:6.00
##  Max.   :6.00000          Max.   :4.00000               Max.   :8.00
##  DeliveryVanPolicies MotorcycleScooterPolicies LorryPolicies
##  Min.   :0.00000     Min.   :0.0000           Min.   :0.000000
##  1st Qu.:0.00000     1st Qu.:0.0000           1st Qu.:0.000000
##  Median :0.00000     Median :0.0000           Median :0.000000
##  Mean   :0.04827     Mean   :0.1754           Mean   :0.009447
##  3rd Qu.:0.00000     3rd Qu.:0.0000           3rd Qu.:0.000000
##  Max.   :7.00000     Max.   :7.0000           Max.   :9.000000
##  TrailerPolicies   TractorPolicies   AgriculturalMachinesPolicies
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.02096   Mean   :0.09258   Mean   :0.01305
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :5.00000   Max.   :6.00000   Max.   :6.00000
##  MopedPolicies   LifeInsurances   PrivateAccidentPolicies
##  Min.   :0.000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.000   Median :0.0000   Median :0.00000
##  Mean   :0.215   Mean   :0.1948   Mean   :0.01374
##  3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :6.000   Max.   :9.0000   Max.   :6.00000
##  FamilyAccidentPolicies DisabilityInsurancePolicies  FirePolicies
##  Min.   :0.00000        Min.   :0.00000             Min.   :0.000
##  1st Qu.:0.00000        1st Qu.:0.00000             1st Qu.:0.000
##  Median :0.00000        Median :0.00000             Median :2.000
##  Mean   :0.01529        Mean   :0.02353             Mean   :1.828
##  3rd Qu.:0.00000        3rd Qu.:0.00000             3rd Qu.:4.000
##  Max.   :3.00000        Max.   :7.00000             Max.   :8.000
##  SurfboardPolicies    BoatPolicies     BicyclePolicies
##  Min.   :0.0000000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.0000000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.0000000   Median :0.00000   Median :0.00000
##  Mean   :0.0008588   Mean   :0.01889   Mean   :0.02525
##  3rd Qu.:0.0000000   3rd Qu.:0.00000   3rd Qu.:0.00000
```

```
##  Max.   :3.0000000   Max.   :6.00000   Max.   :1.00000  
##  PropertyInsurancePolicies SocialSecurityInsurancePolicies
##  Min.   :0.00000           Min.   :0.00000                
##  1st Qu.:0.00000           1st Qu.:0.00000                
##  Median :0.00000           Median :0.00000                
##  Mean   :0.01563           Mean   :0.04758                
##  3rd Qu.:0.00000           3rd Qu.:0.00000                
##  Max.   :6.00000           Max.   :5.00000                
##  NbPrivateThirdPartyInsurance NbThirdPartyInsuranceFirms
##  Min.   :0.000                Min.   :0.00000           
##  1st Qu.:0.000                1st Qu.:0.00000           
##  Median :0.000                Median :0.00000           
##  Mean   :0.403                Mean   :0.01477           
##  3rd Qu.:1.000                3rd Qu.:0.00000           
##  Max.   :2.000                Max.   :5.00000           
##  NbThirdPartyInsuranceAgriculture NbCarPolicies    NbDeliveryVanPolicies
##  Min.   :0.00000                  Min.   :0.0000   Min.   :0.00000      
##  1st Qu.:0.00000                  1st Qu.:0.0000   1st Qu.:0.00000      
##  Median :0.00000                  Median :1.0000   Median :0.00000      
##  Mean   :0.02061                  Mean   :0.5622   Mean   :0.01048      
##  3rd Qu.:0.00000                  3rd Qu.:1.0000   3rd Qu.:0.00000      
##  Max.   :1.00000                  Max.   :7.0000   Max.   :4.00000      
##  NbMotorcycleScooterPolicies NbLorryPolicies    NbTrailerPolicies
##  Min.   :0.00000             Min.   :0.000000   Min.   :0.00000  
##  1st Qu.:0.00000             1st Qu.:0.000000   1st Qu.:0.00000  
##  Median :0.00000             Median :0.000000   Median :0.00000  
##  Mean   :0.04105             Mean   :0.002233   Mean   :0.01254  
##  3rd Qu.:0.00000             3rd Qu.:0.000000   3rd Qu.:0.00000  
##  Max.   :8.00000             Max.   :3.000000   Max.   :3.00000  
##  NbTractorPolicies NbAgriculturalMachinesPolicies NbMopedPolicies
##  Min.   :0.00000   Min.   :0.000000               Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.000000               1st Qu.:0.00000
##  Median :0.00000   Median :0.000000               Median :0.00000
##  Mean   :0.03367   Mean   :0.006183               Mean   :0.07042
##  3rd Qu.:0.00000   3rd Qu.:0.000000               3rd Qu.:0.00000
##  Max.   :4.00000   Max.   :6.000000               Max.   :2.00000
##  NbLifeInsurances  NbPrivateAccidentPolicies NbFamilyAccidentsPolicies
##  Min.   :0.00000   Min.   :0.000000          Min.   :0.000000         
##  1st Qu.:0.00000   1st Qu.:0.000000          1st Qu.:0.000000         
##  Median :0.00000   Median :0.000000          Median :0.000000         
##  Mean   :0.07661   Mean   :0.005325          Mean   :0.006527         
##  3rd Qu.:0.00000   3rd Qu.:0.000000          3rd Qu.:0.000000         
##  Max.   :8.00000   Max.   :1.000000          Max.   :1.000000         
##  NbDisabilityInsurancePolicies NbFirePolicies   NbSurfboardPolicies
##  Min.   :0.000000              Min.   :0.0000   Min.   :0.0000000  
##  1st Qu.:0.000000              1st Qu.:0.0000   1st Qu.:0.0000000  
##  Median :0.000000              Median :1.0000   Median :0.0000000  
##  Mean   :0.004638              Mean   :0.5701   Mean   :0.0005153  
##  3rd Qu.:0.000000              3rd Qu.:1.0000   3rd Qu.:0.0000000  
##  Max.   :2.000000              Max.   :7.0000   Max.   :1.0000000  
```

```
##  NbBoatPolicies     NbBicyclePolicies NbPropertyInsurancePolicies
##  Min.   :0.000000   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.000000
##  Median :0.000000   Median :0.00000   Median :0.000000
##  Mean   :0.006012   Mean   :0.03178   Mean   :0.007901
##  3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.000000
##  Max.   :2.000000   Max.   :3.00000   Max.   :2.000000
##  NbSocialSecurityInsurancePolicies NbMobileHomePolicies
##  Min.   :0.00000                   Min.   :0.00000
##  1st Qu.:0.00000                   1st Qu.:0.00000
##  Median :0.00000                   Median :0.00000
##  Mean   :0.01426                   Mean   :0.05977
##  3rd Qu.:0.00000                   3rd Qu.:0.00000
##  Max.   :2.00000                   Max.   :1.00000
```

```r
# Check number of rows and columns in each of the Train data and Test data
dim(InsData)
```

```
## [1] 5822   86
```

```r
dim(InsDataTest)
```

```
## [1] 4000   86
```

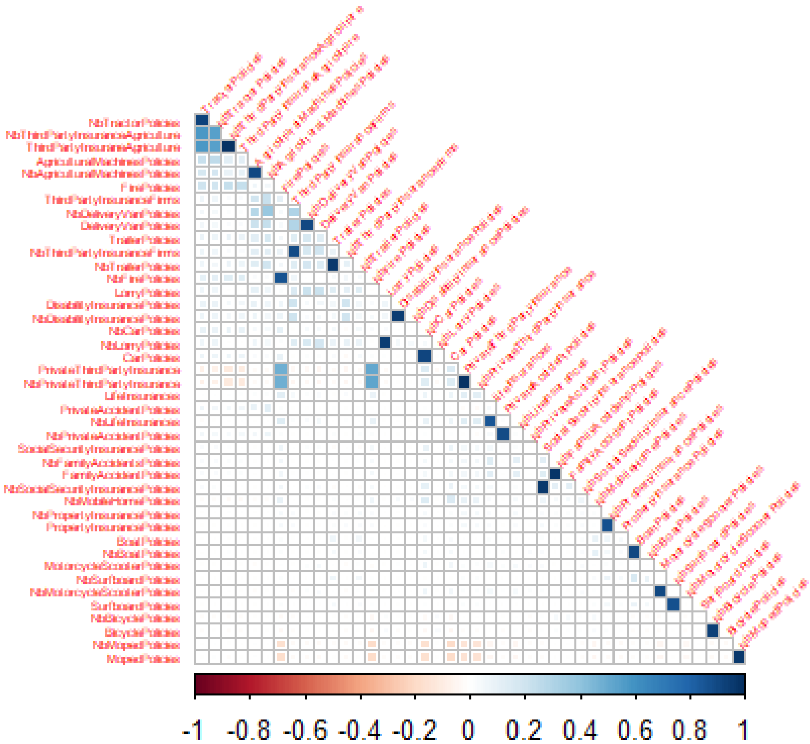
```r
# check the possible values in Target column #86 (NbMobileHomePolicies)
unique(InsData$NbMobileHomePolicies)
```

```
## [1] 0 1
```

This is the Dependent variable with possible values 0 or 1

Check the Correlation Matrix. Since the number of attributes is large, we will check the Attributes related to insurance product purchase first, then check the rest.

```r
#create correlation matrix
corInsData=cor(InsData[,c(44:86)], use="complete.obs")

## The below returns a nice looking correlation matrix with highly correlated
variables other than the diagonal attributes
corrplot(corInsData, method = 'square', order = 'FPC', type = 'lower', diag =
FALSE,tl.cex=0.4, tl.srt=45)
```

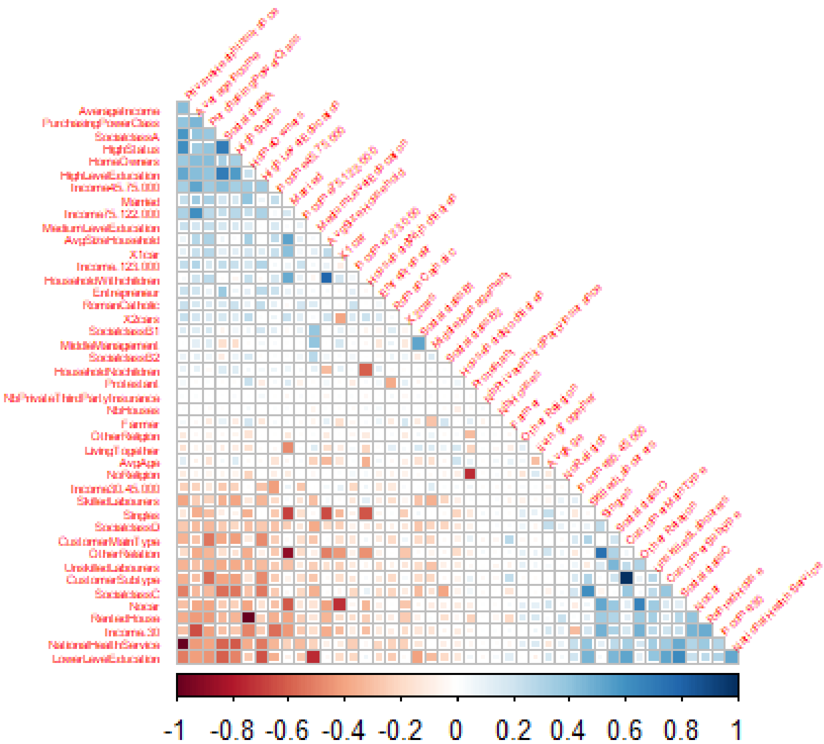


Highly Correlated attributes among the product purchase attributes exist. We can remove one from each of them.

```
# consider subset of the data where the Nb of Insurance Policies is removed a
nd we only keep if they have a certain type of insurance, yes/no
InsDataCut = InsData[,c(1:64,86)]
InsDataTestCut = InsDataTest[,c(1:64,86)]
```

Check next the correlation among the socio-demographic attributes

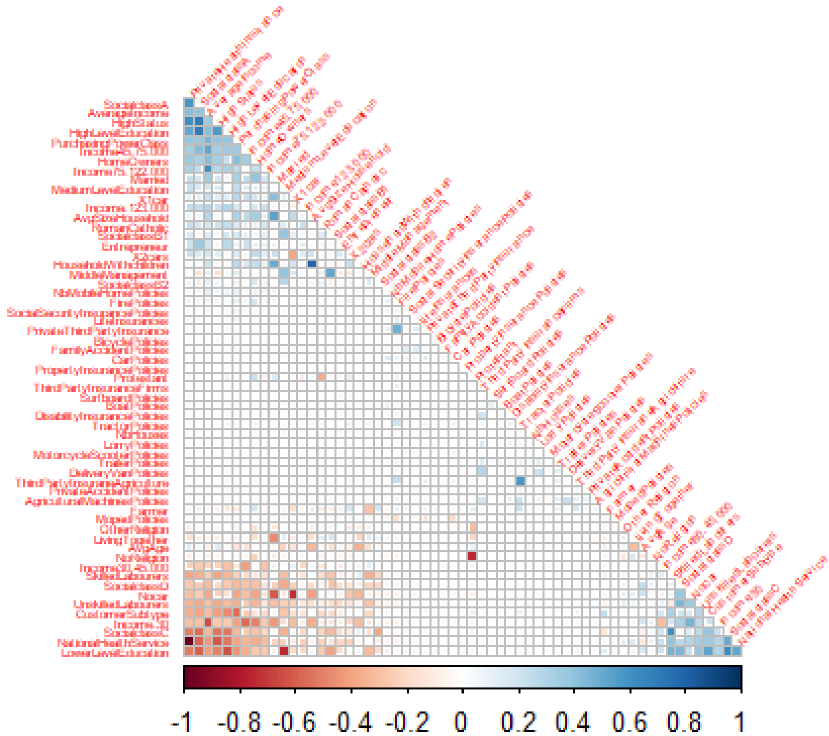
```
# check next the correlation among the socio-demographic attributes

#create correlation matrix
corInsData2=cor(InsData[,c(1:43,65)], use="complete.obs")

## The below returns a nice looking correlation matrix with highly correlated
variables other than the diagonal attributes
corrplot(corInsData2, method = 'square', order = 'FPC', type = 'lower', diag
= FALSE,tl.cex=0.4, tl.srt=45)
```



Here we can also find highly correlated variables that logically mean the same. We will keep one of each from the list below:

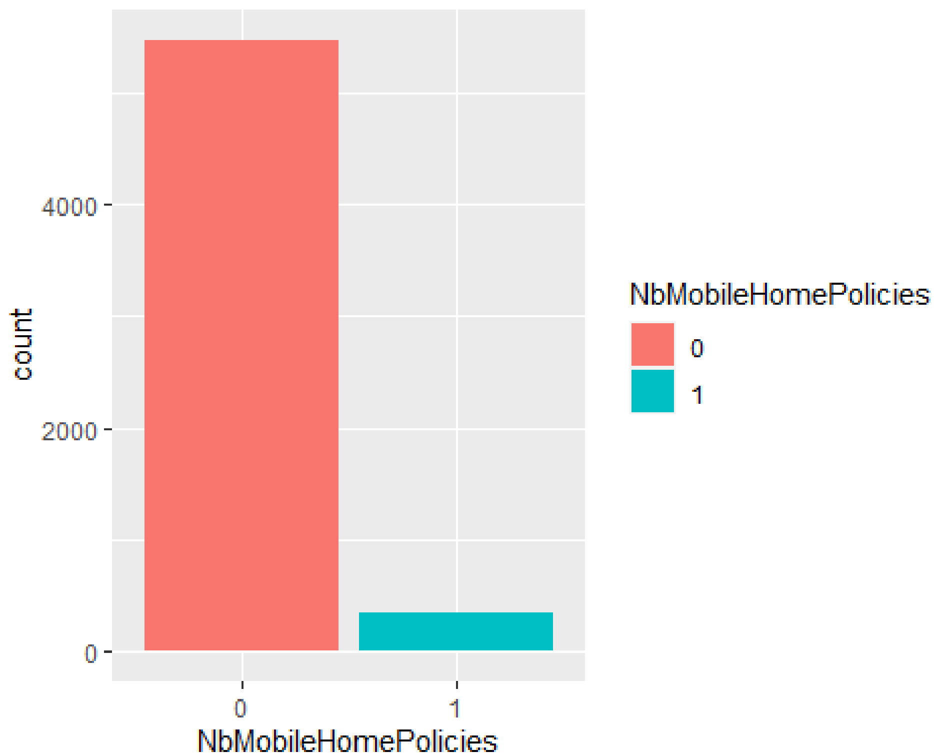
OtherRelation v.s. Married Singles v.s. Married RentedHouse v.s HomeOwners HouseholdNoChildren v.s. HouseholdWithChildren CustomerMainType v.s. CustomerSubType

```
# check next the correlation among all attributes after removing the highly correlated attributes above
InsDataCut = subset(InsDataCut, select = -c(OtherRelation, Singles, RentedHouse, HouseholdNochildren, CustomerMainType))
InsDataTestCut = subset(InsDataTestCut, select = -c(OtherRelation, Singles, RentedHouse, HouseholdNochildren, CustomerMainType))

#create correlation matrix
corInsData2=cor(InsDataCut, use="complete.obs")

## The below returns a nice looking correlation matrix with highly correlated variables other than the diagonal attributes
corrplot(corInsData2, method = 'square', order = 'FPC', type = 'lower', diag = FALSE,tl.cex=0.4, tl.srt=45)
```

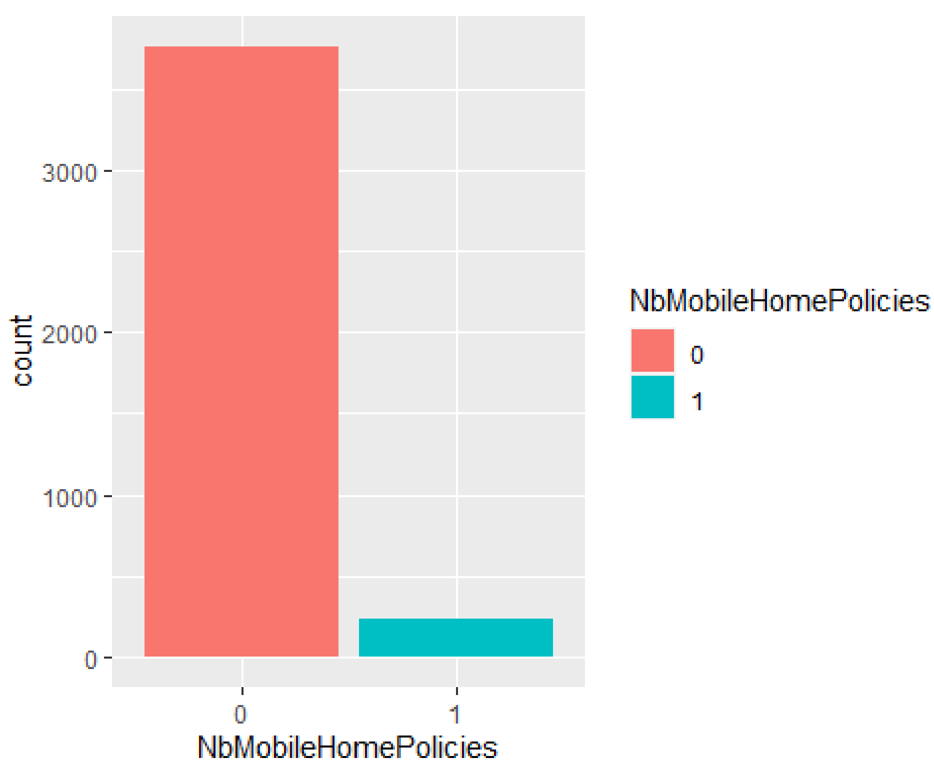


Check for Imbalanced Data in the original dataset

```
InsDataIM = InsData
InsDataTestIM = InsDataTest

# factor the dependent attribute
InsDataIM$NbMobileHomePolicies = as.factor(InsDataIM$NbMobileHomePolicies)
InsDataTestIM$NbMobileHomePolicies = as.factor(InsDataTestIM$NbMobileHomePoli
cies)

# To check if the dataset is balanced, count the number of observations we ha
ve for Targets 0 and 1
# 1 has insurance for mobile home
# 0 does not have insurance for mobile home
ggplot() +
  geom_bar(data=InsDataIM, aes(NbMobileHomePolicies, fill=NbMobileHomePolicie
s))
```




```
ggplot() +
  geom_bar(data=InsDataTestIM, aes(NbMobileHomePolicies, fill=NbMobileHomePol
icies))
```

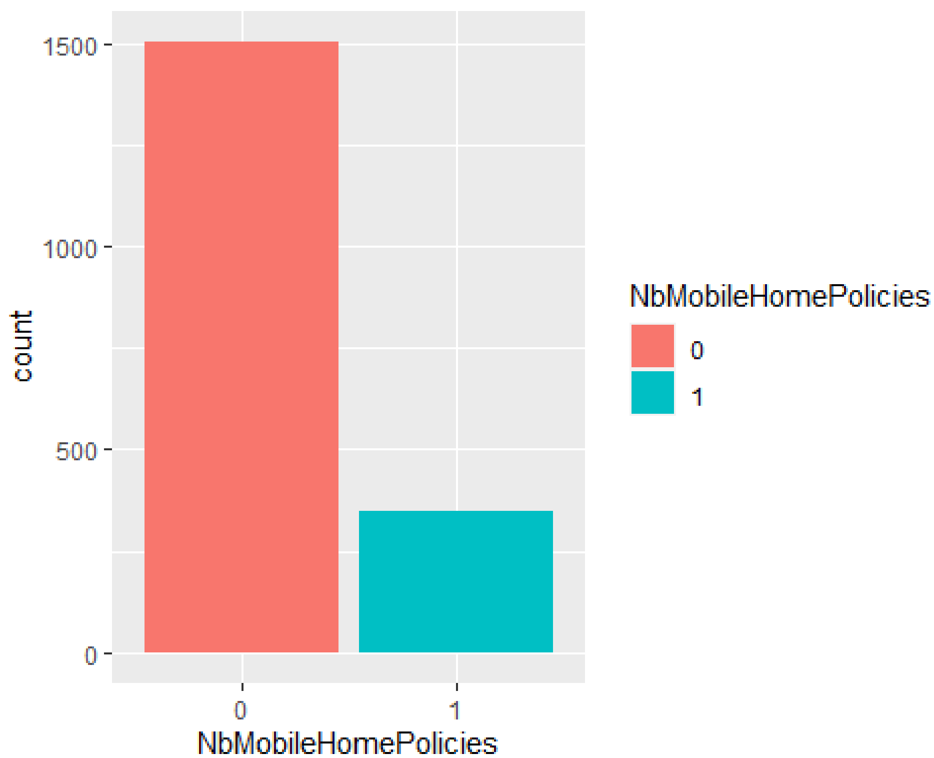


We have imbalanced data as number of 0's in target class are much higher than 1's. But if we check the % of 1's to 0's in each of the Training and Test datasets, they seem similar.

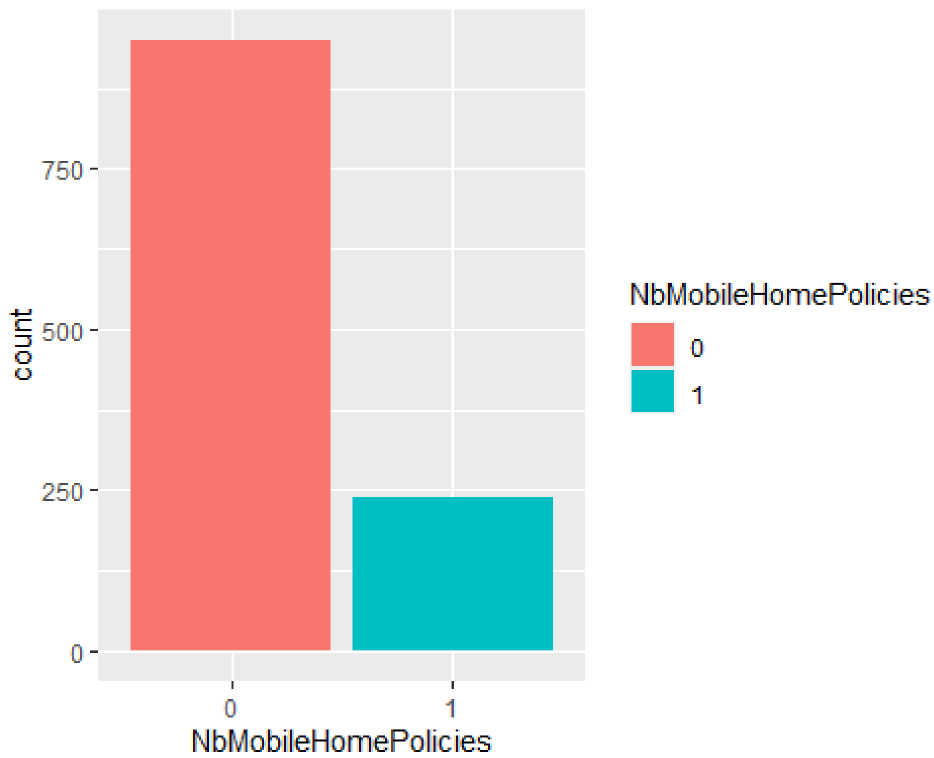
** We might try to undersize the sample data having 0's to be close to those having 1's

```
InsDataIM %>%
  group_by(NbMobileHomePolicies) %>%
  summarize(count = n())

## # A tibble: 2 × 2
##   NbMobileHomePolicies count
##   <fct>                <int>
## 1 0                     5474
## 2 1                      348

# 348 v.s. 5474 (~6.3%)

InsDataTestIM %>%
  group_by(NbMobileHomePolicies) %>%
  summarize(count = n())

## # A tibble: 2 × 2
##   NbMobileHomePolicies count
##   <fct>                <int>
## 1 0                     3762
## 2 1                      238

# 238 v.s. 3762 (~6.3%)

# We will read an edited version of the data where the 0 rows have been dimin
ished to have somewhat balanced data

InsDataIM = read.csv("InsuranceDataTrainIM.csv")
InsDataTestIM = read.csv("InsuranceDataTestIM.csv")


# Check number of rows and columns
dim(InsDataIM)

## [1] 1851   86

dim(InsDataTestIM)

## [1] 1188   86

# factor the dependent attribute
InsDataIM$NbMobileHomePolicies = as.factor(InsDataIM$NbMobileHomePolicies)
InsDataTestIM$NbMobileHomePolicies = as.factor(InsDataTestIM$NbMobileHomePoli
cies)

# To check if the dataset is balanced, count the number of observations we ha
ve for Targets 0 and 1
```

```r
# 1 has insurance for mobile home
# 0 does not have insurance for mobile home
ggplot() +
  geom_bar(data=InsDataIM, aes(NbMobileHomePolicies, fill=NbMobileHomePolicie
s))
```




```r
ggplot() +
  geom_bar(data=InsDataTestIM, aes(NbMobileHomePolicies, fill=NbMobileHomePol
icies))
```



Now we have prepared several datasets: Original Full Dataset (InsData, InsDataTest) Dataset without the Highly correlated Variables (InsDataCut, InsDataTestCut) Dataset treated the imbalanced data (InsDataIM, InsDataTestIM)

We will try 2 algorithms for Prediction: Random Forest and Logistic Regression We will start with the Random Forest, testing it on each of the 3 datasets above:

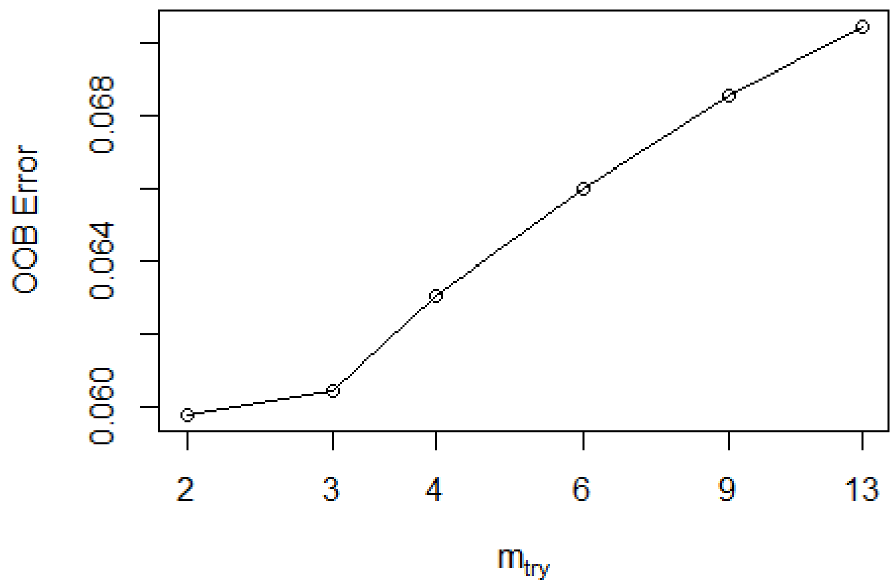
```
# factor the dependent attribute
InsData$NbMobileHomePolicies = as.factor(InsData$NbMobileHomePolicies)
InsDataTest$NbMobileHomePolicies = as.factor(InsDataTest$NbMobileHomePolicies
)

set.seed(71)
rf <-randomForest(NbMobileHomePolicies~.,data=InsData, ntree=500)
print(rf)

##
## Call:
##  randomForest(formula = NbMobileHomePolicies ~ ., data = InsData,      ntr
ee = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 6.85%
## Confusion matrix:
##      0  1 class.error
```

```
## 0 5413 61  0.01114359
## 1  338 10  0.97126437

#If a dependent variable is a factor, classification is assumed, otherwise re
gression is assumed.
```

check results

Find the optimal mtry value. Select mtry value with minimum out of bag(OOB) error. Two important input parameters for random forest: 1- Number of trees used in the forest (ntree) 2- Number of random variables used in each tree (mtry)

```
set.seed(71)
# search for the best mtry value
mtry <- tuneRF(InsData[,c(1:85)],InsData$NbMobileHomePolicies, ntreeTry=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)

## mtry = 9  OOB error = 6.85%
## Searching left ...
## mtry = 6     OOB error = 6.6%
## 0.03759398 0.01
## mtry = 4     OOB error = 6.3%
## 0.04427083 0.01
## mtry = 3     OOB error = 6.05%
## 0.04087193 0.01
## mtry = 2     OOB error = 5.98%
## 0.01136364 0.01
## Searching right ...
## mtry = 13    OOB error = 7.04%
## -0.1781609 0.01
```

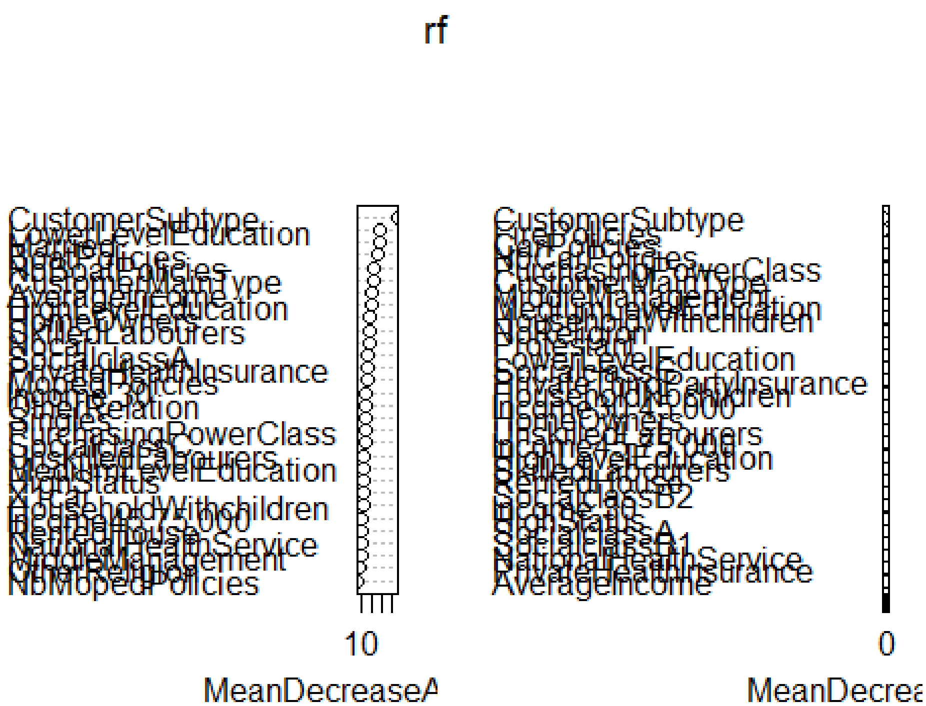



```
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)

##         mtry   OOBError
## 2.OOB      2 0.05977327
## 3.OOB      3 0.06046032
## 4.OOB      4 0.06303676
## 6.OOB      6 0.06595672
## 9.OOB      9 0.06853315
## 13.OOB    13 0.07042254

print(best.m)

## [1] 2
```

In this case, mtry = 2 is the best mtry as it has least OOB error. However, when running the model with mtry = 2, there were no True positives classified. After trying each mtry value, we noticed that the default = 9, was selecting the most true positives with OOB error 6.8, so we kept this value.

```
set.seed(71)
rf <-randomForest(NbMobileHomePolicies~.,data=InsData, mtry=9, importance=TRU
E,ntree=500)
print(rf)

##
## Call:
##  randomForest(formula = NbMobileHomePolicies ~ ., data = InsData,      mtr
```

```
y = 9, importance = TRUE, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 6.8%
## Confusion matrix:
##      0  1 class.error
## 0 5415 59  0.01077822
## 1  337 11  0.96839080

#Evaluate variable importance
#importance(rf)
varImpPlot(rf)
```

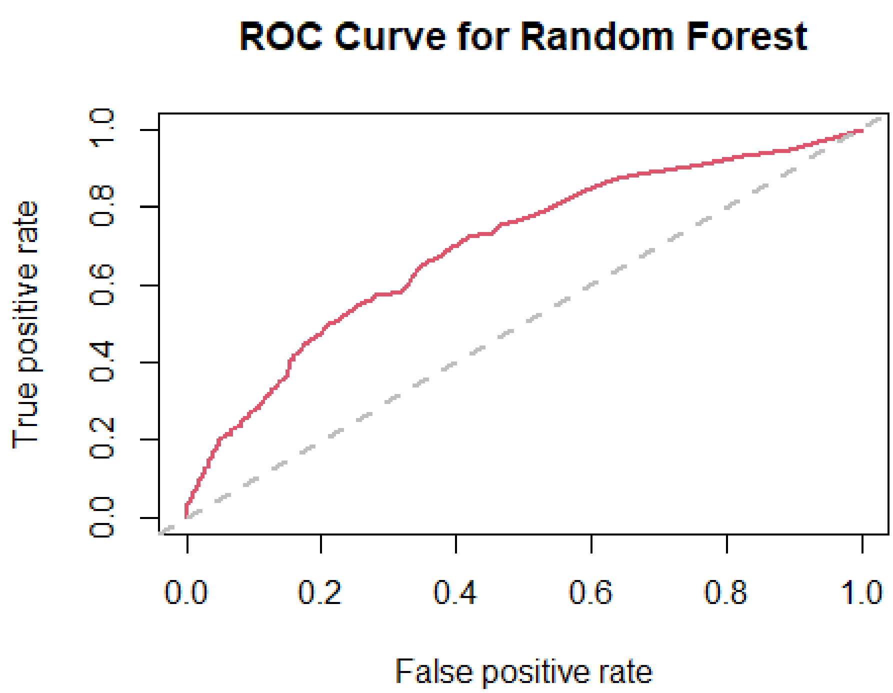


Higher the value of mean decrease accuracy or mean decrease gini score , higher the importance of the variable in the model. In the plot shown above, Customer Sub Type is most important variable. Other Important Variables are: FirePolicies, CarPolicies, NbCarPolicies, PurchasingPowerClass And also: NbBoatPolicies, BoatPolicies, Married, SocialClassC, LowerLevelEducation

Note: Mean Decrease Accuracy - How much the model accuracy decreases if we drop that variable. Mean Decrease Gini - Measure of variable importance based on the Gini impurity index used for the calculation of splits in trees.

Prediction and Calculate Performance Metrics

```
pred1R=predict(rf,type = "prob", newdata=InsDataTest[,c(1:85)])

perfR = prediction(pred1R[,2], InsDataTest$NbMobileHomePolicies)
# 1. Area under curve
auc = performance(perfR, "auc")
auc

## A performance instance
##   'Area under the ROC curve'

# 2. True Positive and Negative Rate
pred3R = performance(perfR, "tpr","fpr")
# 3. Plot the ROC curve
plot(pred3R,main="ROC Curve for Random Forest",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```



This means that the proportion of test samples correctly classified as Purchasing Caravan Insurance (true positive) is greater than the proportion of the sample incorrectly classified as Purchasing Caravan Insurance (false positives)

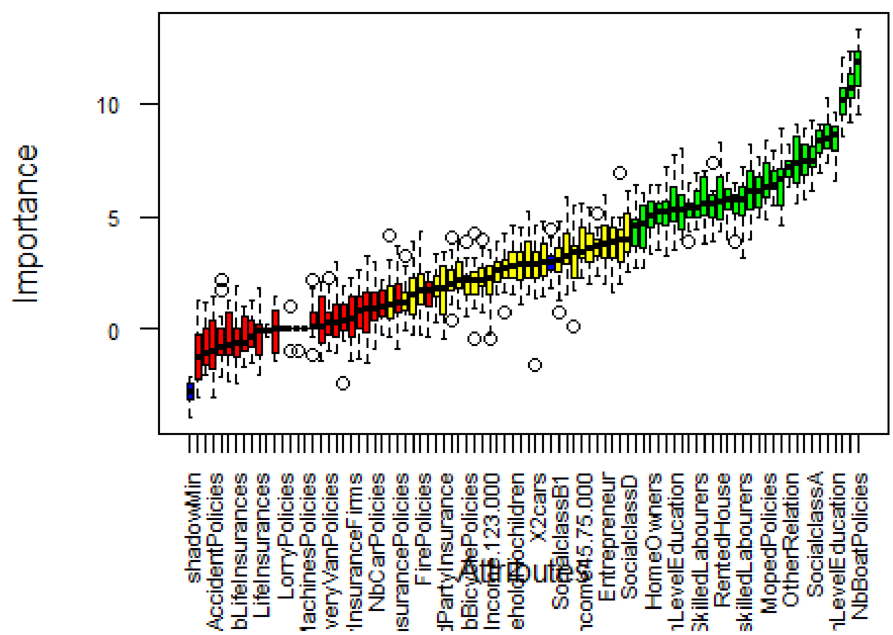
Another nice way to show feature selection based on random forest, using library(Boruta)

```
boruta <- Boruta(NbMobileHomePolicies~., data = InsData, doTrace = 2, maxRuns
= 20)

##  1. run of importance source...

##  2. run of importance source...
```

```
##  3. run of importance source...

##  4. run of importance source...

##  5. run of importance source...

##  6. run of importance source...

##  7. run of importance source...

##  8. run of importance source...

##  9. run of importance source...

##  10. run of importance source...

##  11. run of importance source...

##  12. run of importance source...

##  13. run of importance source...

##  14. run of importance source...

## After 14 iterations, +1.2 mins:

##  confirmed 26 attributes: AverageIncome, BoatPolicies, CustomerMainType, C
ustomerSubtype, HighLevelEducation and 21 more;

##  rejected 20 attributes: AgriculturalMachinesPolicies, DeliveryVanPolicies
, LifeInsurances, LorryPolicies, MotorcycleScooterPolicies and 15 more;

##  still have 39 attributes left.

##  15. run of importance source...

##  16. run of importance source...

##  17. run of importance source...

##  18. run of importance source...

## After 18 iterations, +1.3 mins:

##  confirmed 4 attributes: AvgSizeHousehold, Income75.122.000, UnskilledLabo
urers, X1car;

##  rejected 6 attributes: FamilyAccidentPolicies, NbCarPolicies, NbDeliveryV
anPolicies, NbFamilyAccidentsPolicies, NbHouses and 1 more;

##  still have 29 attributes left.

##  19. run of importance source...

print(boruta)
```

```
## Boruta performed 19 iterations in 1.359412 mins.
##  30 attributes confirmed important: AverageIncome, AvgSizeHousehold,
## BoatPolicies, CustomerMainType, CustomerSubtype and 25 more;
##  26 attributes confirmed unimportant: AgriculturalMachinesPolicies,
## DeliveryVanPolicies, FamilyAccidentPolicies, LifeInsurances,
## LorryPolicies and 21 more;
##  29 tentative attributes left: AvgAge, BicyclePolicies, CarPolicies,
## DisabilityInsurancePolicies, Entrepreneur and 24 more;
```

```
plot(boruta, las = 2, cex.axis = 0.7)
```



After 20 iterations: 22 attributes confirmed important, those are listed below.

```
getSelectedAttributes(boruta, withTentative = F)
```

```
##  [1] "CustomerSubtype"         "AvgSizeHousehold"         "CustomerMainType"
##  [4] "Married"                 "OtherRelation"            "Singles"
##  [7] "HouseholdWithchildren"   "HighLevelEducation"       "MediumLevelEducati
on"
## [10] "LowerLevelEducation"     "HighStatus"               "MiddleManagement"
## [13] "SkilledLabourers"        "UnskilledLabourers"       "SocialclassA"
## [16] "SocialclassC"            "RentedHouse"              "HomeOwners"
## [19] "X1car"                   "Nocar"                    "NationalHealthServ
ice"
## [22] "PrivateHealthInsurance" "Income.30"                "Income75.122.000"
## [25] "AverageIncome"           "PurchasingPowerClass"     "MopedPolicies"
## [28] "BoatPolicies"            "NbMopedPolicies"          "NbBoatPolicies"
```

IF we compare with the features based on random Forest MeanDecreaseAccuracy, we notice similarity in the top 5 attributes: CustomerSubType, NbBoatPolicies, BoatPolicies, Married, SocialClassC, LowerLevelEducation

We will build a 2nd Random Forest model after Treating the Imblanaced Data

```
InsDataIM$NbMobileHomePolicies = as.factor(InsDataIM$NbMobileHomePolicies)
InsDataTestIM$NbMobileHomePolicies = as.factor(InsDataTestIM$NbMobileHomePoli
cies)

set.seed(71)
rfIM <-randomForest(NbMobileHomePolicies~.,data=InsDataIM, ntree=500)
print(rfIM)

##
## Call:
##  randomForest(formula = NbMobileHomePolicies ~ ., data = InsDataIM,      n
tree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 19.72%
## Confusion matrix:
##      0  1 class.error
## 0 1426 77  0.05123087
## 1  288 60  0.82758621

#If a dependent variable is a factor, classification is assumed, otherwise re
gression is assumed.
```

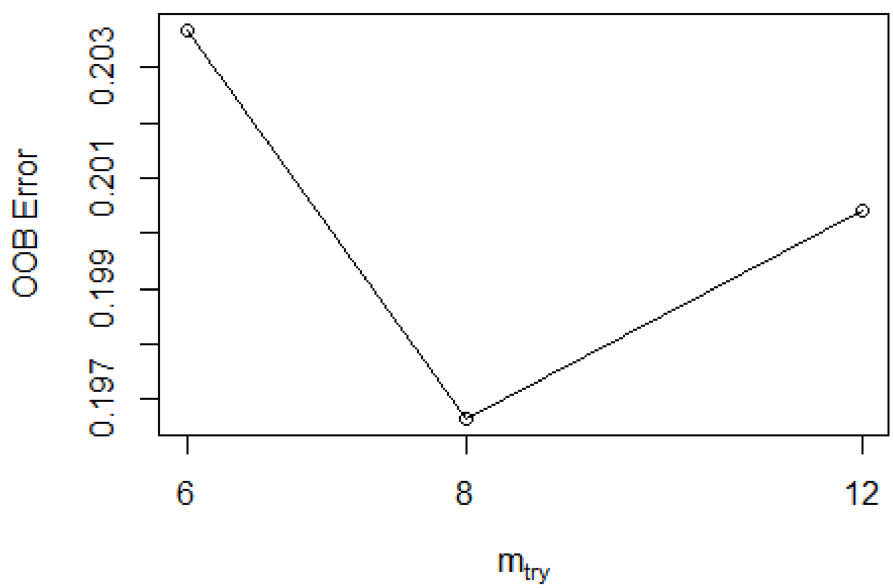
check results, we notice that by trying to balance the data “removed part of the 0’s from both training and test data” and also trying to keep only 1 of the variables that seem highly correlated (mainly whether they have a certain insurance type and the nb of insurance policies for that type). By doing the above the OOB estimate of error rate increased from 6.85% to 19.72% which might not sound good, but if we look at how much the True Positivies matched in the confusion matrix, 66 out of the 137 it is much better than the first model.

Find the optimal mtry value. Select mtry value with minimum out of bag(OOB) error. Two important input parameters for random forest: 1- Number of trees used in the forest (ntree) 2- Number of random variables used in each tree (mtry)

The out-of-bag (OOB) error is the average error for each calculated using predictions from the trees that do not contain in their respective bootstrap sample. This allows the RandomForestClassifier to be fit and validated whilst being trained.

```
# remove the dependent variable and search for the best mtry value
mtry <- tuneRF(InsDataIM[,c(1:64)],InsDataIM$NbMobileHomePolicies, ntreeTry=5
00,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
```

```
## mtry = 8  OOB error = 19.67%
## Searching left ...
## mtry = 6     OOB error = 20.37%
## -0.03571429 0.01
## Searching right ...
## mtry = 12    OOB error = 20.04%
## -0.01923077 0.01
```



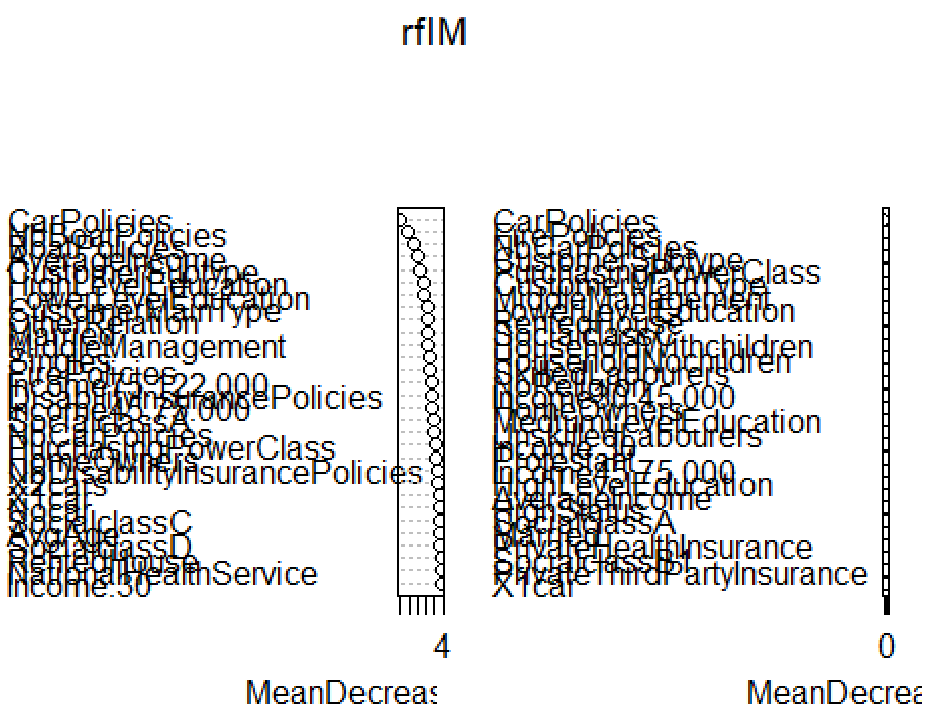

```
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)

##         mtry  OOBError
## 6.OOB      6 0.2036737
## 8.OOB      8 0.1966505
## 12.OOB    12 0.2004322

print(best.m)

## [1] 8
```

In this case, mtry = 8 is the best mtry as it has least OOB error. However, when mtry = 12 it has more number of True Positives.

Build model again using mtry =12 value.

```
set.seed(71)
rfIM <-randomForest(NbMobileHomePolicies~.,data=InsDataIM, mtry=12, importanc
```

```
e=TRUE,ntree=500)
print(rfIM)

##
## Call:
##  randomForest(formula = NbMobileHomePolicies ~ ., data = InsDataIM,      m
try = 12, importance = TRUE, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##         OOB estimate of  error rate: 20.1%
## Confusion matrix:
##      0  1 class.error
## 0 1416 87  0.05788423
## 1  285 63  0.81896552

#Evaluate variable importance
#importance(rf)
varImpPlot(rfIM)
```



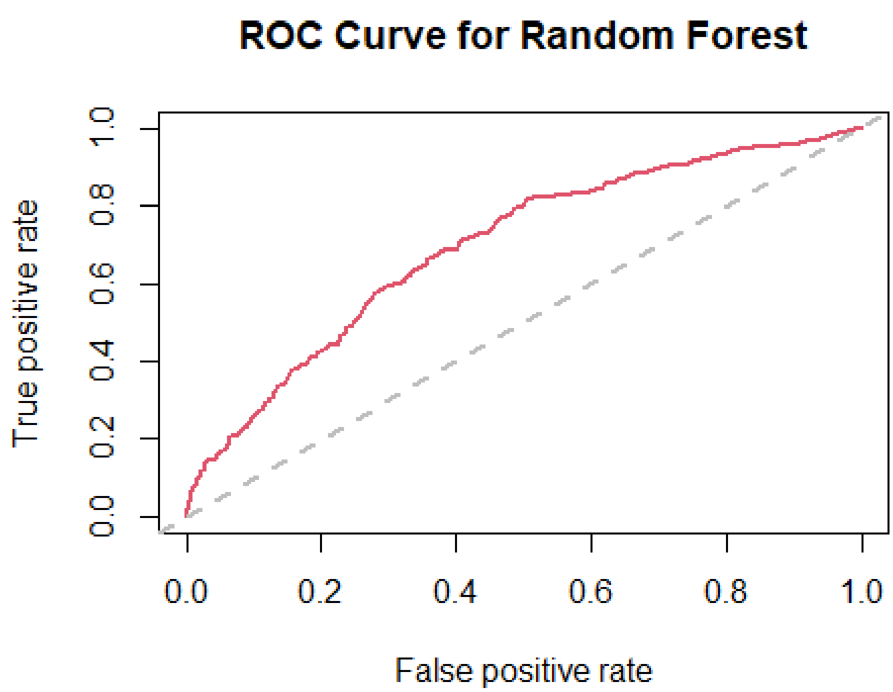
Higher the value of mean decrease accuracy or mean decrease gini score , higher the importance of the variable in the model. In the plot shown above, Customer Sub Type is most important variable. Other Important Variables are: FirePolicies, CarPolicies, NbofCarPolicies, PurchasingPowerClass And also: NbBoatPolicies, BoatPolicies, Married, SocialClassC, LowerLevelEducation

Note: Mean Decrease Accuracy - How much the model accuracy decreases if we drop that variable. Mean Decrease Gini - Measure of variable importance based on the Gini impurity index used for the calculation of splits in trees.

If we look at the Confusion Matrix for the Treated imbalanced data and after using the best mtry value = 12, we get an OOB rate of 20.1% with an increased number of matching True Postivies of 63 out of 150.

Prediction and Calculate Performance Metrics

```
pred1IM=predict(rfIM,type = "prob", newdata=InsDataTestIM[,c(1:85)])

perfIM = prediction(pred1IM[,2], InsDataTestIM$NbMobileHomePolicies)
# 1. Area under curve
aucIM = performance(perfIM, "auc")
aucIM

## A performance instance
##   'Area under the ROC curve'

# 2. True Positive and Negative Rate
pred3IM = performance(perfIM, "tpr","fpr")
# 3. Plot the ROC curve
plot(pred3IM,main="ROC Curve for Random Forest",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```



We will build a third Random Forest model after removing the Highly Correlated Variables

```
InsDataCut$NbMobileHomePolicies = as.factor(InsDataCut$NbMobileHomePolicies)
InsDataTestCut$NbMobileHomePolicies = as.factor(InsDataTestCut$NbMobileHomePo
licies)

set.seed(71)
rfCut <-randomForest(NbMobileHomePolicies~.,data=InsDataCut, ntree=500)
print(rfCut)

##
## Call:
##  randomForest(formula = NbMobileHomePolicies ~ ., data = InsDataCut,
ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 6.68%
## Confusion matrix:
##      0  1 class.error
## 0 5424 50 0.009134088
## 1  339  9 0.974137931

#If a dependent variable is a factor, classification is assumed, otherwise re
gression is assumed.
```

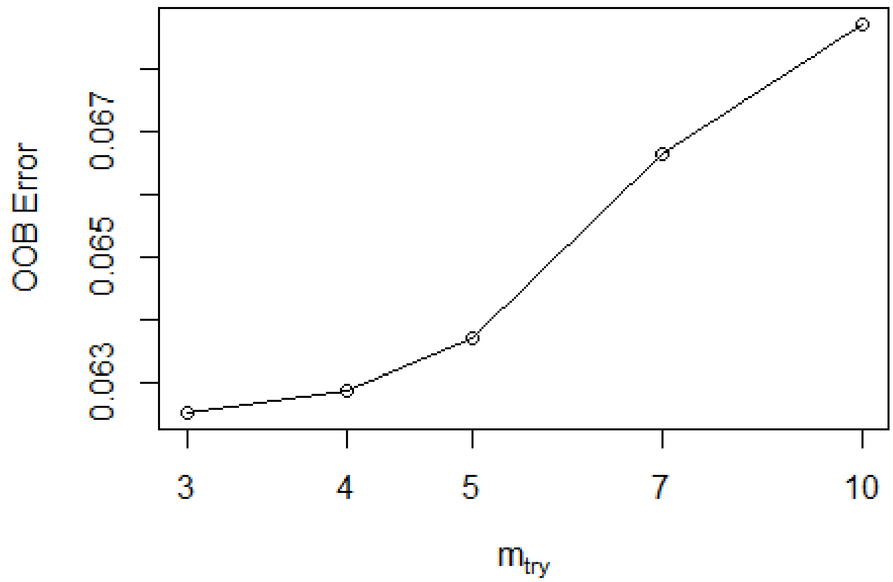
check results, we notice that by trying to balance the data “removed part of the 0’s from both training and test data” and also trying to keep only 1 of the variables that seem highly correlated (mainly whether they have a certain insurance type and the nb of insurance policies for that type). By doing the above the OOB estimate of error rate decreased from 6.85% to 6.68% which means a better model is achieved.

Find the optimal mtry value. Select mtry value with minimum out of bag(OOB) error. Two important input parameters for random forest: 1- Number of trees used in the forest (ntree) 2- Number of random variables used in each tree (mtry)

The out-of-bag (OOB) error is the average error for each calculated using predictions from the trees that do not contain in their respective bootstrap sample. This allows the RandomForestClassifier to be fit and validated whilst being trained.

```
# remove the dependent variable and search for the best mtry value
mtry <- tuneRF(InsDataCut[,c(1:59)],InsDataCut$NbMobileHomePolicies, ntreeTry
=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)

## mtry = 7  OOB error = 6.66%
## Searching left ...
## mtry = 5     OOB error = 6.37%
## 0.04381443 0.01
## mtry = 4     OOB error = 6.29%
```

```
## 0.01347709 0.01
## mtry = 3     OOB error = 6.25%
## 0.005464481 0.01
## Searching right ...
## mtry = 10    OOB error = 6.87%
## -0.09289617 0.01
```




```
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)
```

```
##          mtry   OOBError
## 3.OOB       3 0.06252147
## 4.OOB       4 0.06286499
## 5.OOB       5 0.06372381
## 7.OOB       7 0.06664377
## 10.OOB     10 0.06870491
```

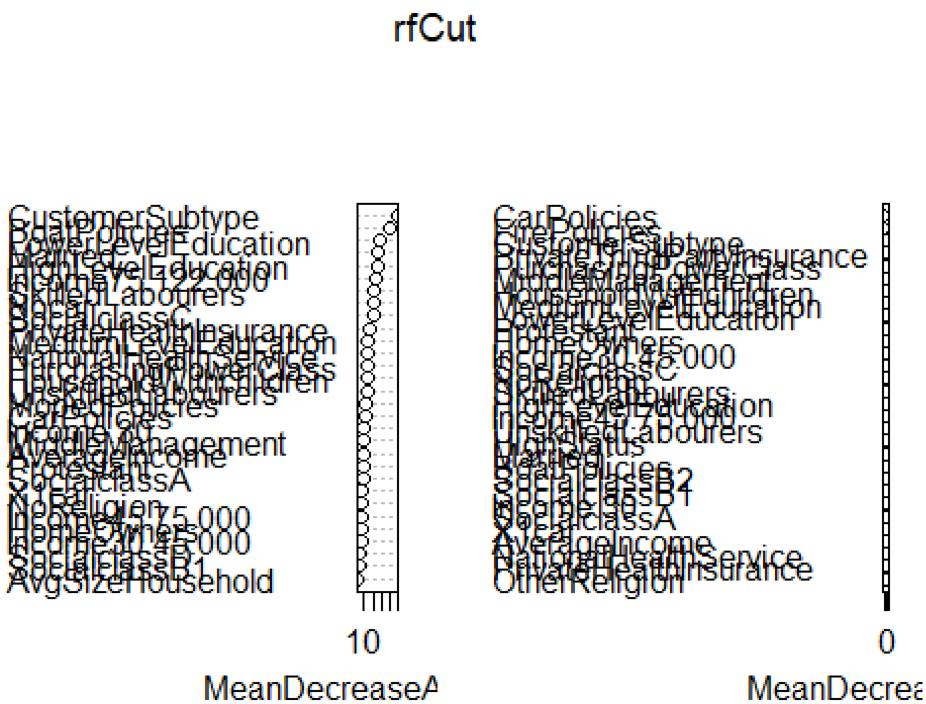
```
print(best.m)
```

```
## [1] 3
```

In this case, mtry = 3 is the best mtry as it has least OOB error. However this didn't return any true positives. So we kept the mtry = 7.

```
set.seed(71)
rfCut <-randomForest(NbMobileHomePolicies~.,data=InsDataCut, mtry=7, importan
ce=TRUE,ntree=500)
print(rfCut)
```

```
##
## Call:
##  randomForest(formula = NbMobileHomePolicies ~ ., data = InsDataCut,
mtry = 7, importance = TRUE, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 6.58%
## Confusion matrix:
##      0  1 class.error
## 0 5429 45  0.00822068
## 1  338 10  0.97126437

#Evaluate variable importance
#importance(rf)
varImpPlot(rfCut)
```



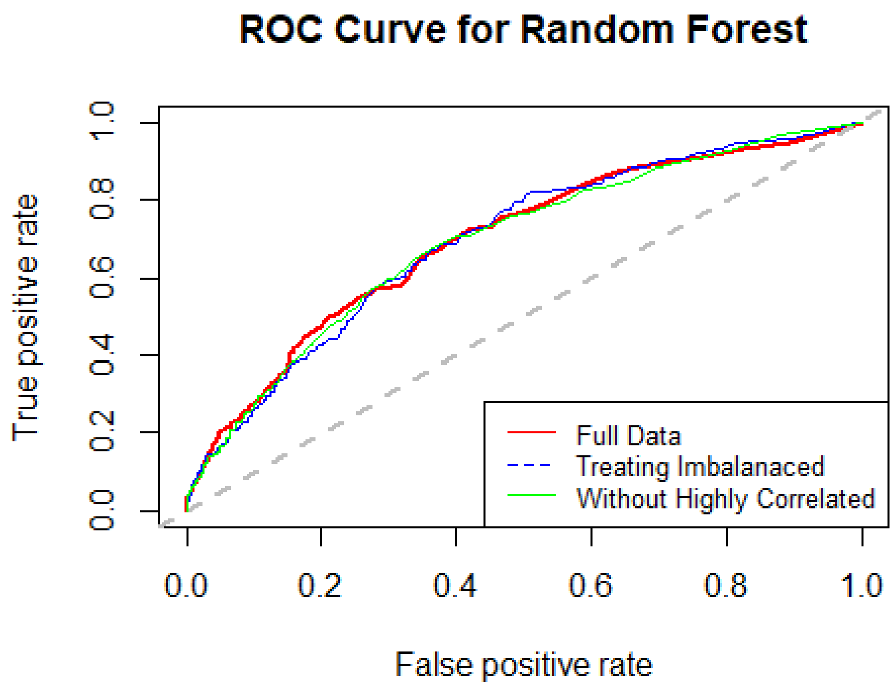
Higher the value of mean decrease accuracy or mean decrease gini score , higher the importance of the variable in the model. In the plot shown above, Customer Sub Type is most important variable. Other Important Variables are: FirePolicies, CarPolicies, NbofCarPolicies, PurchasingPowerClass And also: NbBoatPolicies, BoatPolicies, Married, SocialClassC, LowerLevelEducation

Note: Mean Decrease Accuracy - How much the model accuracy decreases if we drop that variable. Mean Decrease Gini - Measure of variable importance based on the Gini impurity index used for the calculation of splits in trees.

Looking at the Confusion Matrix after taking the best mtry variable = 3 with lowest OOB error rate = 6.25% seems no True Positives were detected, so better to keep the default split at mtry = 7 with OOB rate 6.58 with 10 matching true positives.

Prediction and Calculate Performance Metrics

```
pred1=predict(rfCut,type = "prob", newdata=InsDataTestCut[,c(1:59)])

perf = prediction(pred1[,2], InsDataTestCut$NbMobileHomePolicies)
# 1. Area under curve
auc = performance(perf, "auc")
auc

## A performance instance
##   'Area under the ROC curve'

# 2. True Positive and Negative Rate
pred3 = performance(perf, "tpr","fpr")


# 3. Plot the ROC curve
# add First curve
plot(pred3R,main="ROC Curve for Random Forest",lwd=2, col = "red")
abline(a=0,b=1,lwd=2,lty=2,col="gray")

# add second curve After treating imbalanced data
#plot(pred3,add = TRUE, colorize = TRUE,col=2,lwd=2)
lines(pred3IM@x.values[[1]], pred3IM@y.values[[1]], col = "blue")
#points(pred3IM@x.values[[1]], pred3IM@y.values[[1]], col="blue", pch="*")

# add 3rd curve After removing highly correlated variables
lines(pred3@x.values[[1]], pred3@y.values[[1]], col = "green")

# Add a legend
legend("bottomright", legend=c("Full Data", "Treating Imbalanaced", "Without
Highly Correlated"),
       col=c("red", "blue", "green"), lty=1:2, cex=0.8)
```



This means that the proportion of test samples correctly classified as Purchasing Caravan Insurance (true positive) is greater than the proportion of the sample incorrectly classified as Purchasing Caravan Insurance (false positives)

Now we will continue with the Prediction using Logistic Regression and try it on the Full Dataset + compare it with Random Forest.

```
lr.fits <- glm(NbMobileHomePolicies~.,data=InsDataIM, family=binomial)

lr.probs = predict(lr.fits, InsDataTestIM[,c(1:85)], type="response")
lr.pred = rep("No",4000)
lr.pred[lr.probs >.5]=" Yes"
table(lr.pred , InsDataTest$NbMobileHomePolicies)

##
## lr.pred    0    1
##      Yes  185   16
##     No   3577  222

# 16/238 = 7%

lr.pred=rep("No",4000)
lr.pred[lr.probs >.25]=" Yes"
table(lr.pred ,InsDataTest$NbMobileHomePolicies)
```

```
##
## lr.pred    0    1
##     Yes 1009   80
##    No   2753  158
```

```
# 80 / 238 = 34%
```

When using 50% as classifying probability, only 16 of the test observations are predicted to purchase insurance. Though the positive rate is ~7%. When using 25% as classifier, we get better results, we have 80 predicted to purchase insurance with a true positive rate of ~34%.

We also tried to apply Logistic Regression on the data that has Treated Imbalanced as well as removing highly correlated attributes with better number on true postives

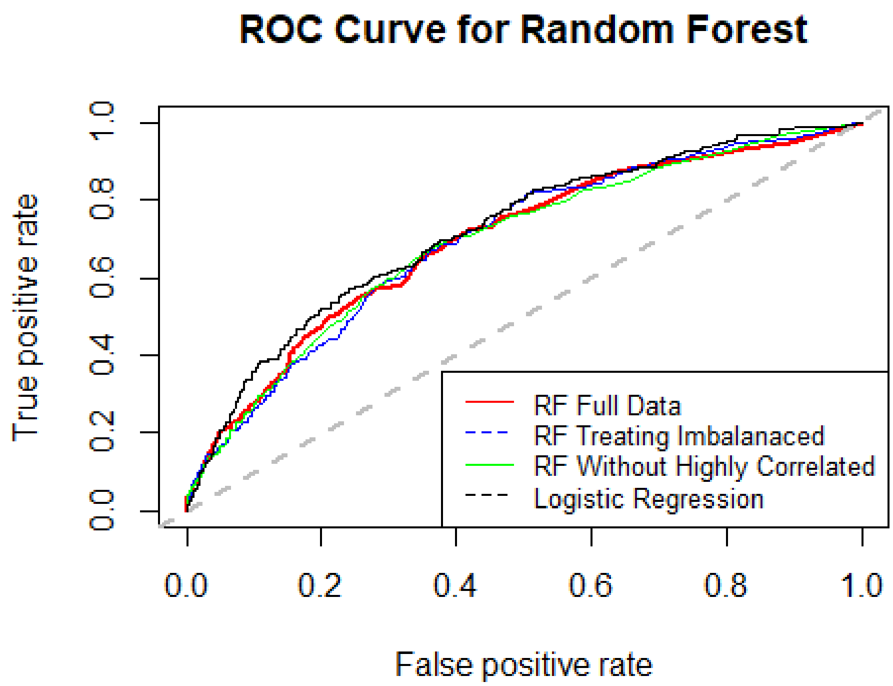
```
InsDataIMCut = InsDataIM[,c(1:64,86)]
InsDataTestIMCut = InsDataTestIM[,c(1:64,86)]

lr2.fits <- glm(NbMobileHomePolicies~.,data=InsDataIMCut, family=binomial)

lr2.probs = predict(lr2.fits, InsDataTestIMCut[,c(1:64)], type="response")
lr2.pred = rep("No",1188)
lr2.pred[lr2.probs >.5]=" Yes"
table(lr2.pred , InsDataTestIMCut$NbMobileHomePolicies)

##
## lr2.pred   0   1
##      Yes  22  25
##     No   928 213

# 25/238 = 11%

lr2.pred=rep("No",1188)
lr2.pred[lr2.probs >.25]=" Yes"
table(lr2.pred ,InsDataTestIMCut$NbMobileHomePolicies)

##
## lr2.pred   0   1
##      Yes 193 124
##     No   757 114

# 124 / 238 = 52%
```

We notice the when using the cut of probability as 25% we got True Potisitive matching 52%

Calculate Performance Metrics

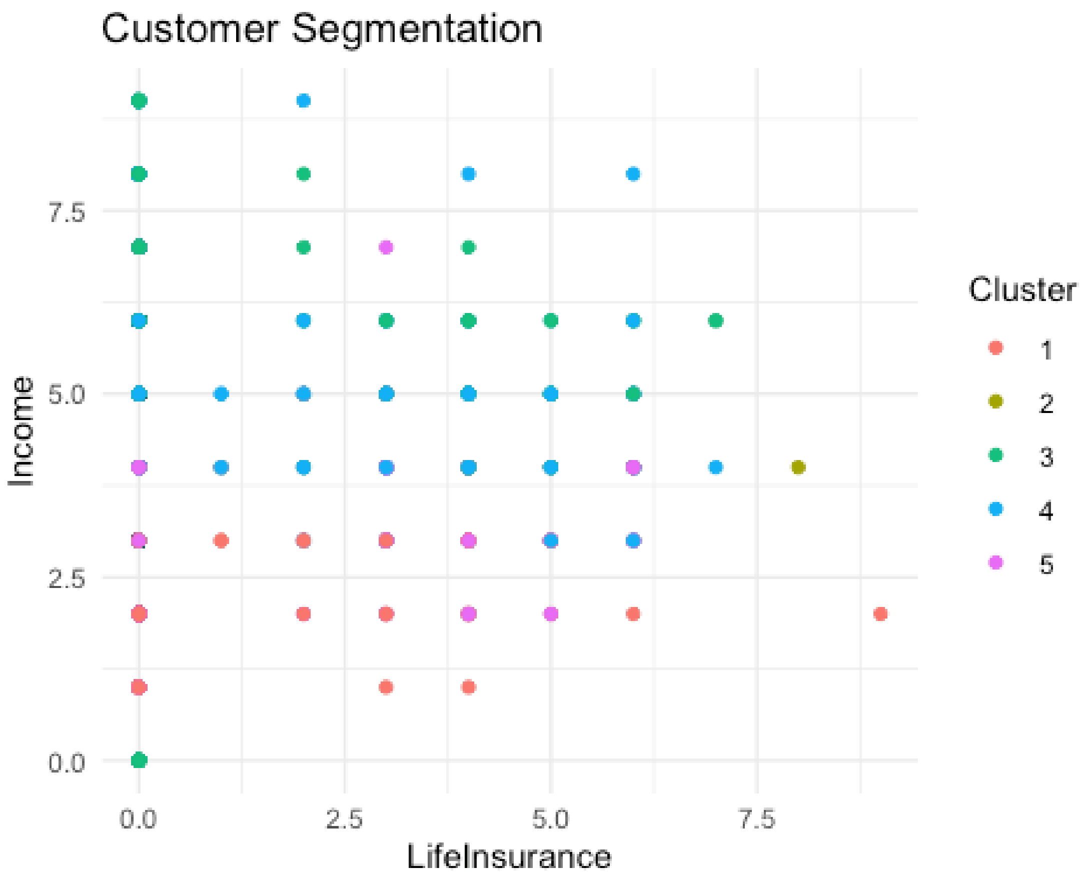
```
pred1LR=predict(lr2.fits, type = "response", newdata=InsDataTestIMCut[,c(1:64
)])
```

```r
perfLR = prediction(pred1LR, InsDataTestIMCut$NbMobileHomePolicies)
# 1. Area under curve
auc = performance(perfLR, "auc")
auc

## A performance instance
##   'Area under the ROC curve'

# 2. True Positive and Negative Rate
pred3LR = performance(perfLR, "tpr","fpr")

# 3. Plot the ROC curve
# add First curve
plot(pred3R,main="ROC Curve for Random Forest",lwd=2, col = "red")
abline(a=0,b=1,lwd=2,lty=2,col="gray")

# add second curve After treating imbalanced data
#plot(pred3,add = TRUE, colorize = TRUE,col=2,lwd=2)
lines(pred3IM@x.values[[1]], pred3IM@y.values[[1]], col = "blue")
#points(pred3IM@x.values[[1]], pred3IM@y.values[[1]], col="blue", pch="*")

# add 3rd curve After removing highly correlated variables
lines(pred3@x.values[[1]], pred3@y.values[[1]], col = "green")

# add 3rd curve After removing highly correlated variables
lines(pred3LR@x.values[[1]], pred3LR@y.values[[1]], col = "black")


# Add a legend
legend("bottomright", legend=c("RF Full Data", "RF Treating Imbalanaced", "RF
Without Highly Correlated", "Logistic Regression"),
       col=c("red", "blue", "green", "black"), lty=1:2, cex=0.8)
```

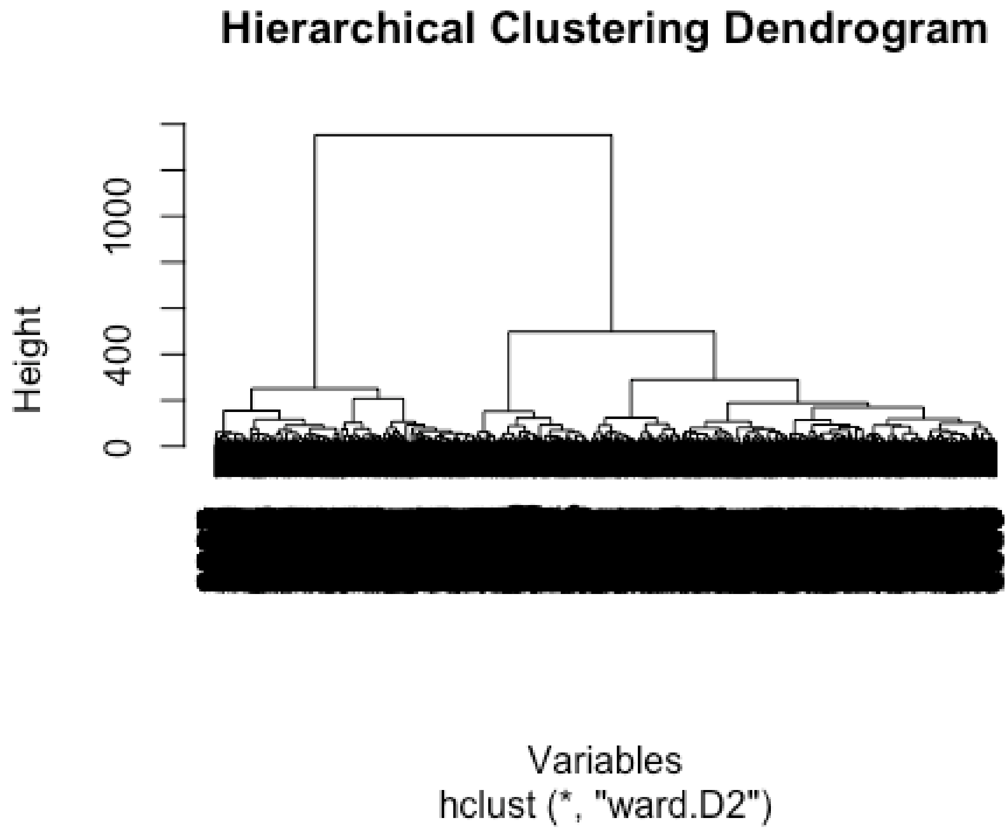

ROC Curve for Random Forest
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
RF Full Data
RF Treating Imbalanaced
RF Without Highly Correlated
Logistic Regression

Question 2: Customer segmentation with K-means Clustering and Heirarchial clustering algorithm

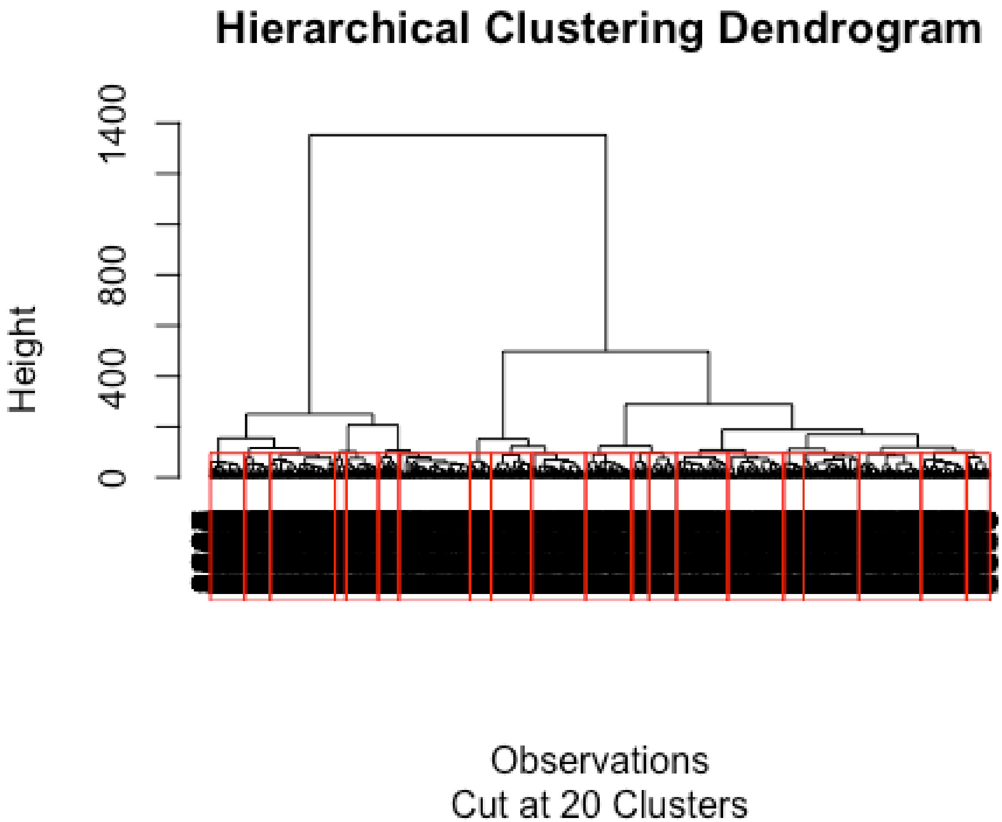
```r
library('caret')

## Loading required package: ggplot2

## Loading required package: lattice

library('dplyr')

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(class)
train_data = read.csv("/Users/hadoop/Downloads/STAT515-005-Team2-FinalProject
/Code/InsuranceDataTrain.csv", header = TRUE)

#Scale the data
scaled_data <- scale(train_data)

#Perform K-means clustering
k <- 5  # Number of clusters
kmeans_model <- kmeans(scaled_data, centers = k)

#Extract cluster assignments
cluster_assignments <- kmeans_model$cluster

#Plot the clusters
ggplot(data = train_data, aes(x = LifeInsurances, y = AverageIncome, color =
as.factor(cluster_assignments))) +
  geom_point() +
  labs(title = "Customer Segmentation",
       x = "LifeInsurance",
       y = "Income",
       color = "Cluster") +
  theme_minimal()
```




```
distance_matrix <- dist(train_data, method = "euclidean")

#hierarchical clustering
hierarchical_model <- hclust(distance_matrix, method = "ward.D2")

#dendrogram
plot(hierarchical_model, main = "Hierarchical Clustering Dendrogram", xlab =
"Variables")
```



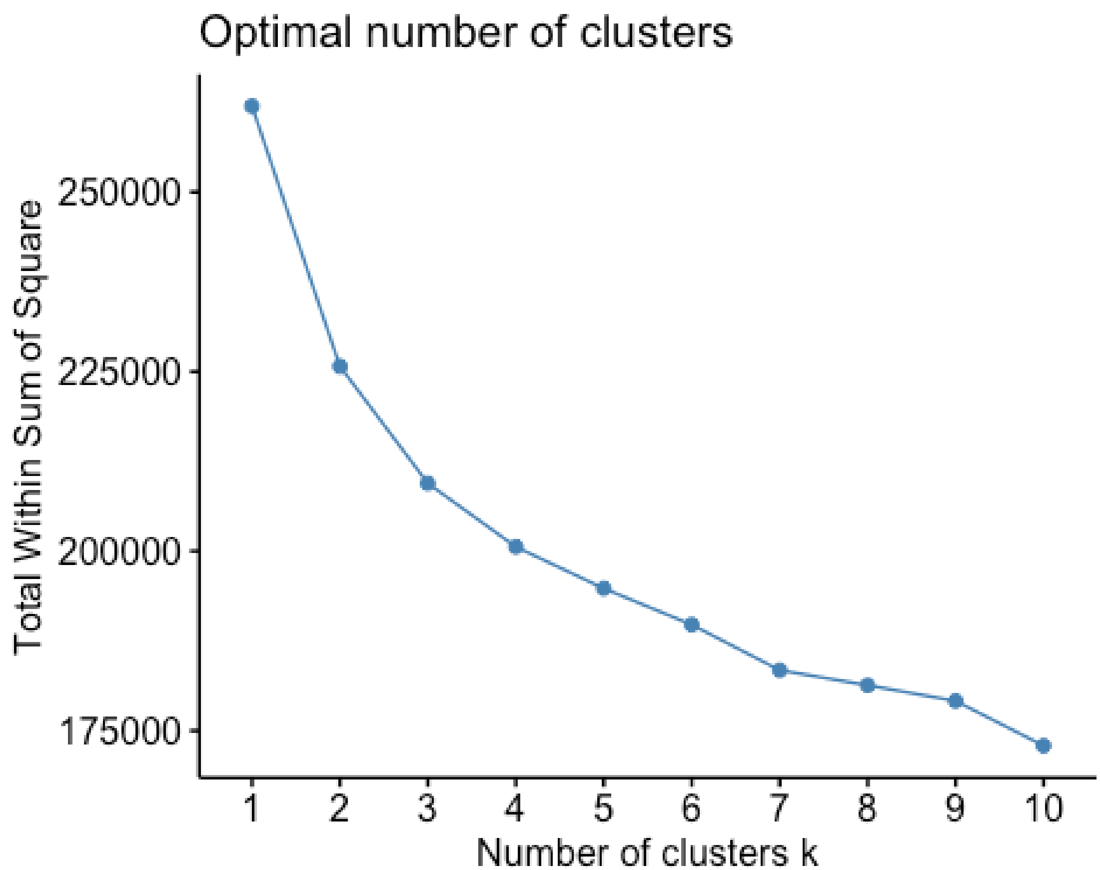

```r
num_clusters <- 20  # Number of clusters
cluster_assignments <- cutree(hierarchical_model, k = num_clusters)

# Create a data frame with cluster assignments
data_with_clusters <- data.frame(train_data, cluster = as.factor(cluster_assi
gnments))

plot(hierarchical_model, hang = -1, main = "Hierarchical Clustering Dendrogra
m",
     xlab = "Observations", sub = paste("Cut at", num_clusters, "Clusters"))
rect.hclust(hierarchical_model, k = num_clusters, border = "red")
```



We do not have clear cluster seperations in the above plots , so we went ahead and tried PCA dimensionality reduction alongside with K means Clustering algorithm

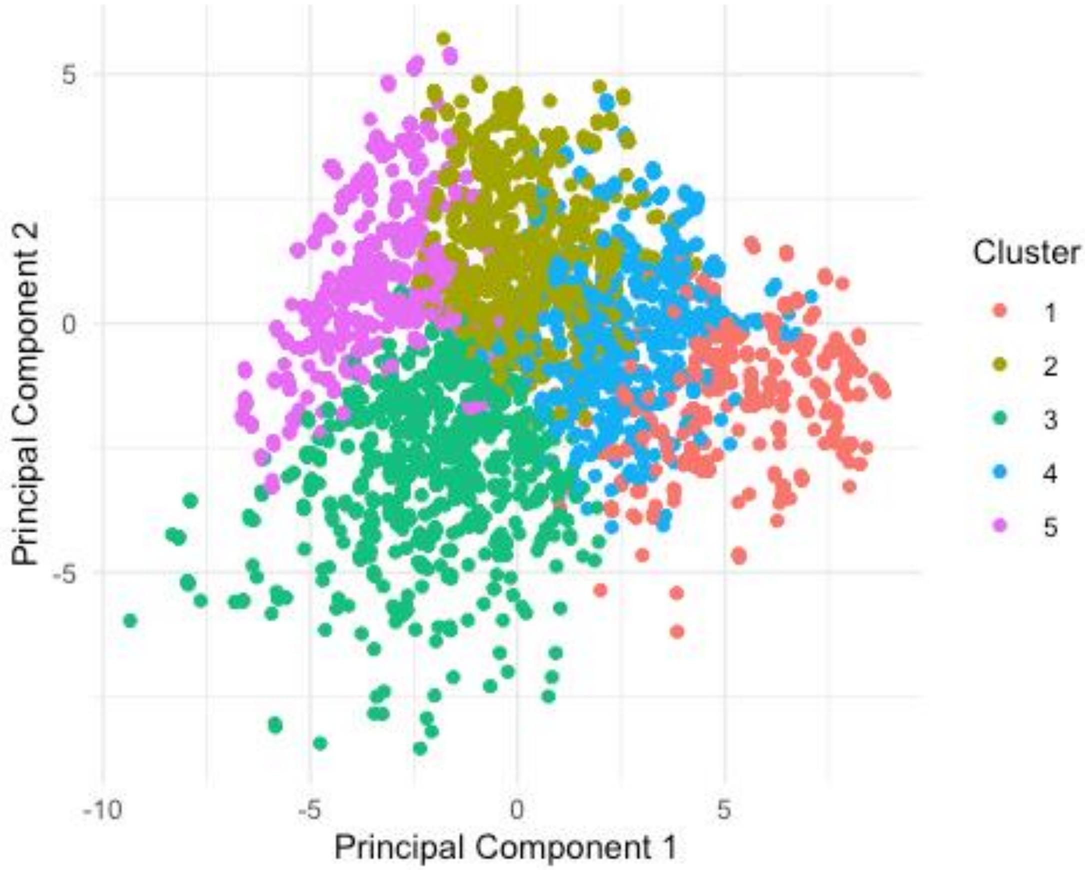
to find optimal K value

```
# Compute and plot the within-cluster sum of squares (WCSS) for different k v
alues
fviz_nbclust(pca_scores, kmeans, method = "wss", k.max = 10)
```



PCA + K-means clustering algorithm

```
# Set the number of clusters based on the "elbow" point
k <- 5

# Perform k-means clustering
kmeans_result <- kmeans(pca_scores, centers = k, nstart = 25)

# Add the cluster assignments to the PCA plot
pca_clusters <- as.data.frame(pca_scores)
pca_clusters$Cluster <- as.factor(kmeans_result$cluster)

# Plot the PCA plot with colored clusters
pca_plot_clusters <- ggplot(pca_clusters, aes(x = PC1, y = PC2, color = Clust
er)) +
  geom_point() +
  labs(x = "Principal Component 1", y = "Principal Component 2", color = "Clu
ster") +
  theme_minimal()
pca_plot_clusters
```

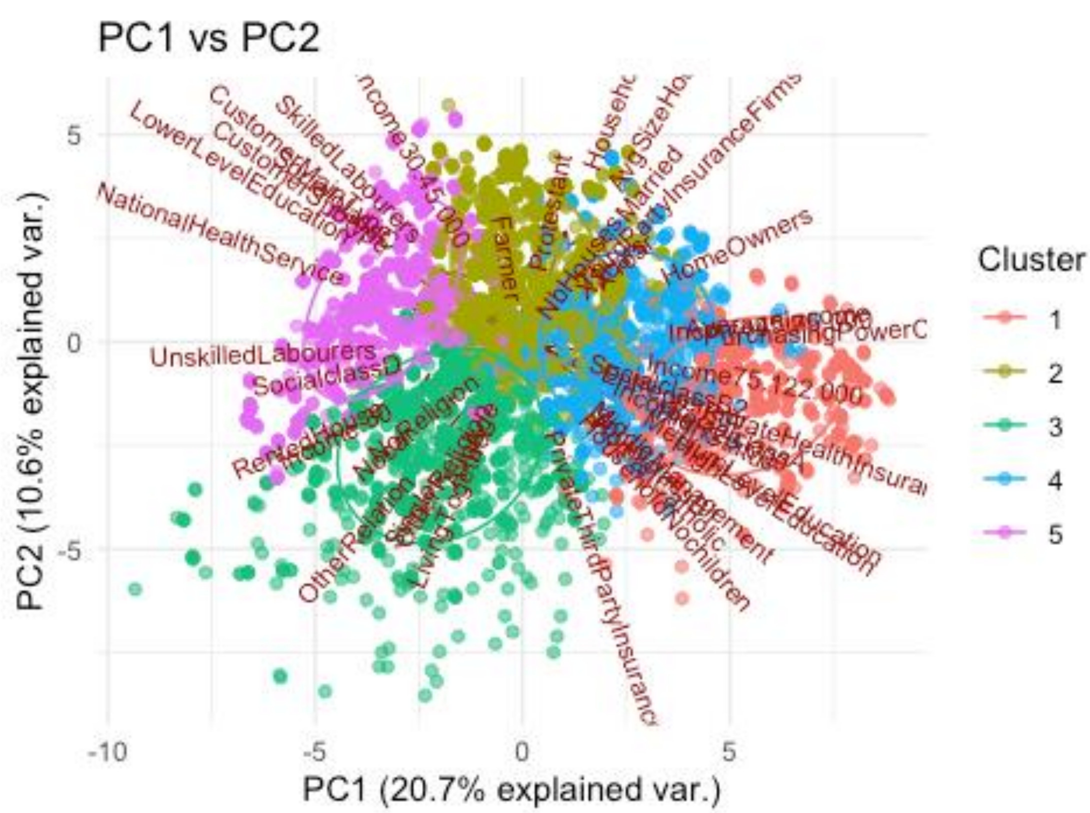


Here we can clearly see the separation between the clusters

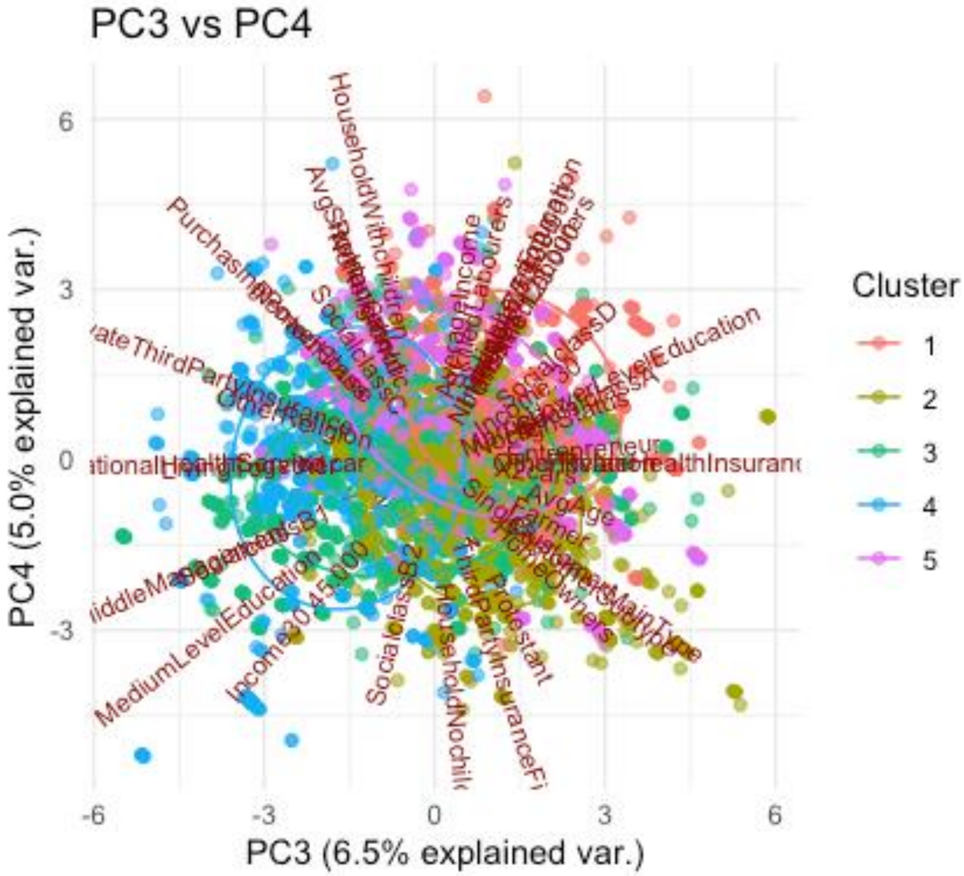
Understand Customer Segmentation.

```
# Filter the data based on selected clusters
filtered_df <- pca_clusters[pca_clusters$Cluster %in% c("1", "4", "5"), ]

# Perform the merge using row indexes
merged_df <- merge(filtered_df, train_data[, 1:45], by.x = "row.names", by.y
= "row.names")

# Remove the ID column from the merged data frame
merged_df$row.names <- NULL

# Convert Cluster variable to character
merged_df$Cluster <- as.character(merged_df$Cluster)

# PC1 Vs PC2
p1 <- ggbiplot(pca_result, obs.scale = 1, var.scale = 1, groups = as.characte
r(kmeans_result$cluster),
         ellipse = TRUE, circle = FALSE, alpha = 0.6) +
  scale_color_discrete(name = "Cluster") +
  labs(title = "PC1 vs PC2") +
  theme_minimal()
```

```
p1 <- p1 %>%
  add_trace(x = pca_result$PC1, y = pca_result$PC2,
            text = rownames(pca_result),
            hoverinfo = "text",
            mode = "text",
            textfont = list(color = "black", size = 12),
            textposition = "top center")

# PC3 Vs PC4
p2 <- ggbiplot(pca_result, obs.scale = 1, var.scale = 1, groups = as.characte
r(kmeans_result$cluster),
         choices = c(3, 4), ellipse = TRUE, circle = FALSE, alpha = 0.6) +
  scale_color_discrete(name = "Cluster") +
  labs(title = "PC3 vs PC4") +
  theme_minimal()

p1
```




```
p2
```


PC3 vs PC4
PC4 (5.0% explained var.)
PC3 (6.5% explained var.)
6
3
0
-3
-6
-3
0
3
6
MediumLevelEducation
Protestant
AvgAge
Farmer
Cluster
1
2
3
4
5

Question 3: Is there a relationship between buying a caravan insurance and having other insurance policy types?

```r
library(ggplot2)
library(plotly)
data <- read.csv("InsuranceDataTrain.csv", header = TRUE)
data1 <-read.csv("InsuranceDataTest.csv", header = TRUE)
#create a data frame with rows 65 through 86

df <- data[,c(44:64, 86)]

# Split dataframe into two parts: the first 21 columns, and the 22nd column

x <- as.matrix(df[1:21])
y <- as.factor(df[,22])

# Perform chi-squared test for each column in `x` with respect to `y`
results <- apply(x, 2, function(x) chisq.test(x, y))

# View the results
results
```

```
## $PrivateThirdPartyInsurance
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 57.476, df = 3, p-value = 2.034e-12
##
##
## $ThirdPartyInsuranceFirms
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 3.9124, df = 6, p-value = 0.6885
```

```
##
##
## $ThirdPartyInsuraneAgriculture
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 2.8469, df = 3, p-value = 0.4158
##
##
## $CarPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 194.69, df = 5, p-value < 2.2e-16
##
##
## $DeliveryVanPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 0.83309, df = 3, p-value = 0.8415
##
##
## $MotorcycleScooterPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 23.685, df = 5, p-value = 0.0002496
##
##
```

```
## $LorryPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 0.57305, df = 3, p-value = 0.9026
##
##
## $TrailerPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 4.0259, df = 5, p-value = 0.5457
##
##
## $TractorPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 3.9146, df = 4, p-value = 0.4177
##
##
## $AgriculturalMachinesPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 1.3399, df = 4, p-value = 0.8546
##
##
## $MopedPolicies
##
```

```
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 12.87, df = 5, p-value = 0.02463
##
##
## $LifeInsurances
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 10.337, df = 9, p-value = 0.3239
##
##
## $PrivateAccidentPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 0.83439, df = 6, p-value = 0.9911
##
##
## $FamilyAccidentPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 14.442, df = 2, p-value = 0.000731
##
##
## $DisabilityInsurancePolicies
##
##  Pearson's Chi-squared test
##
```

```
## data:  x and y
## X-squared = 7.9587, df = 4, p-value = 0.0931
##
##
## $FirePolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 140.39, df = 8, p-value < 2.2e-16
##
##
## $SurfboardPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 6.9624, df = 2, p-value = 0.03077
##
##
## $BoatPolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 80.941, df = 6, p-value = 2.284e-15
##
##
## $BicyclePolicies
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  x and y
## X-squared = 4.0535, df = 1, p-value = 0.04408
```

```
##
##
## $PropertyInsurancePolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 5.5866, df = 6, p-value = 0.4711
##
##
## $SocialSecurityInsurancePolicies
##
##  Pearson's Chi-squared test
##
## data:  x and y
## X-squared = 29.362, df = 4, p-value = 6.598e-06
```

```r
# Create a data frame to store the results
result_table <- data.frame(Variable = colnames(x), p_value = sapply(results, function(x) x$p.value), stringsAsFactors = FALSE)

# Define the p-value thresholds for low, medium, and high
p_value_thresholds <- c(0.01, 0.05)

# Group the results based on p-value thresholds
result_table$Group <- cut(result_table$p_value, c(0, p_value_thresholds, 1), labels = c("High", "Medium", "Low"), right = FALSE)

# Display the result table
result_table
```

```
##                                                         Variable      p_val
ue
## PrivateThirdPartyInsurance           PrivateThirdPartyInsurance 2.033526e-
12
## ThirdPartyInsuranceFirms               ThirdPartyInsuranceFirms 6.885267e-
01
## ThirdPartyInsuraneAgriculture     ThirdPartyInsuraneAgriculture 4.158361e-
01
```

```
## CarPolicies                                           CarPolicies 3.888906e-
40
## DeliveryVanPolicies                           DeliveryVanPolicies 8.415373e-
01
## MotorcycleScooterPolicies               MotorcycleScooterPolicies 2.495952e-
04
## LorryPolicies                                       LorryPolicies 9.025744e-
01
## TrailerPolicies                                   TrailerPolicies 5.456894e-
01
## TractorPolicies                                   TractorPolicies 4.176891e-
01
## AgriculturalMachinesPolicies         AgriculturalMachinesPolicies 8.545754e-
01
## MopedPolicies                                       MopedPolicies 2.462775e-
02
## LifeInsurances                                     LifeInsurances 3.239193e-
01
## PrivateAccidentPolicies                   PrivateAccidentPolicies 9.911198e-
01
## FamilyAccidentPolicies                     FamilyAccidentPolicies 7.309731e-
04
## DisabilityInsurancePolicies           DisabilityInsurancePolicies 9.310246e-
02
## FirePolicies                                         FirePolicies 1.965861e-
26
## SurfboardPolicies                               SurfboardPolicies 3.077105e-
02
## BoatPolicies                                         BoatPolicies 2.283642e-
15
## BicyclePolicies                                   BicyclePolicies 4.408044e-
02
## PropertyInsurancePolicies               PropertyInsurancePolicies 4.710528e-
01
## SocialSecurityInsurancePolicies SocialSecurityInsurancePolicies 6.598398e-
06
##                                   Group
## PrivateThirdPartyInsurance         High
## ThirdPartyInsuranceFirms            Low
## ThirdPartyInsuraneAgriculture       Low
## CarPolicies                        High
```

```
## DeliveryVanPolicies                 Low
## MotorcycleScooterPolicies         High
## LorryPolicies                       Low
## TrailerPolicies                     Low
## TractorPolicies                     Low
## AgriculturalMachinesPolicies        Low
## MopedPolicies                    Medium
## LifeInsurances                      Low
## PrivateAccidentPolicies             Low
## FamilyAccidentPolicies            High
## DisabilityInsurancePolicies         Low
## FirePolicies                      High
## SurfboardPolicies                Medium
## BoatPolicies                      High
## BicyclePolicies                  Medium
## PropertyInsurancePolicies           Low
## SocialSecurityInsurancePolicies   High
```

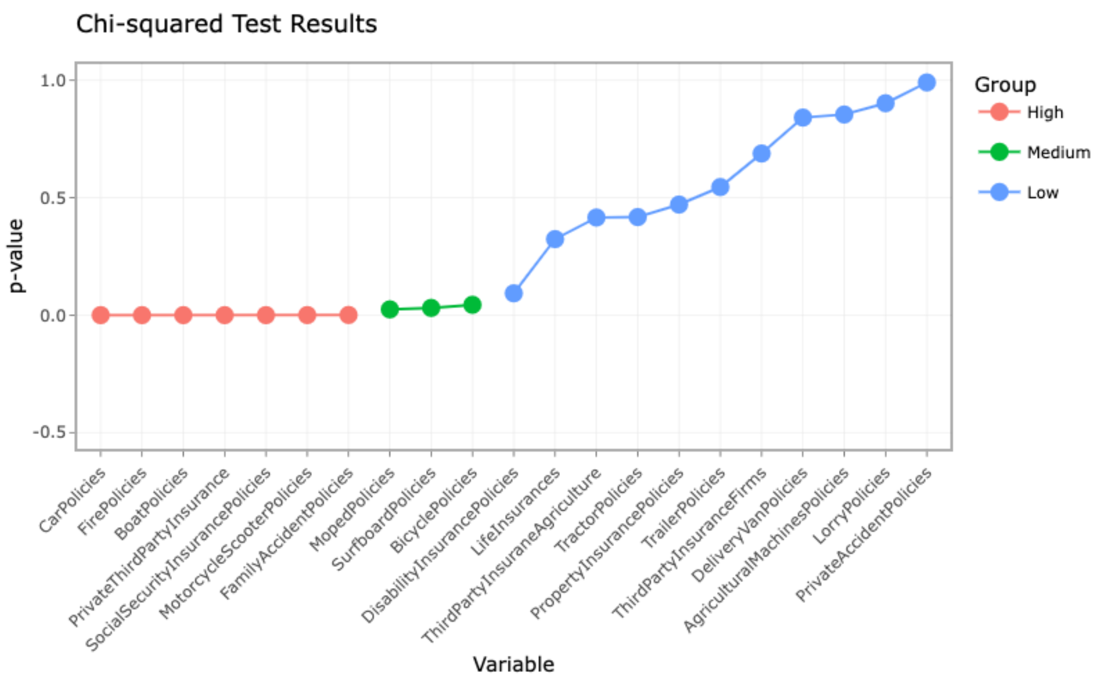
```
# Sort the result_table by p-value in ascending order
result_table_sorted <- result_table[order(result_table$p_value), ]

# Reorder the levels of the "Variable" factor variable
result_table_sorted$Variable <- factor(result_table_sorted$Variable,
                                       levels = result_table_sorted$Variable)


# Print the sorted result_table
result_table_sorted
```

```
##                                                         Variable      p_val
ue
## CarPolicies                                       CarPolicies 3.888906e-
40
## FirePolicies                                     FirePolicies 1.965861e-
26
## BoatPolicies                                     BoatPolicies 2.283642e-
15
## PrivateThirdPartyInsurance           PrivateThirdPartyInsurance 2.033526e-
12
```

```
## SocialSecurityInsurancePolicies SocialSecurityInsurancePolicies 6.598398e-
06
## MotorcycleScooterPolicies           MotorcycleScooterPolicies 2.495952e-
04
## FamilyAccidentPolicies                 FamilyAccidentPolicies 7.309731e-
04
## MopedPolicies                                   MopedPolicies 2.462775e-
02
## SurfboardPolicies                           SurfboardPolicies 3.077105e-
02
## BicyclePolicies                               BicyclePolicies 4.408044e-
02
## DisabilityInsurancePolicies       DisabilityInsurancePolicies 9.310246e-
02
## LifeInsurances                                 LifeInsurances 3.239193e-
01
## ThirdPartyInsuraneAgriculture   ThirdPartyInsuraneAgriculture 4.158361e-
01
## TractorPolicies                               TractorPolicies 4.176891e-
01
## PropertyInsurancePolicies           PropertyInsurancePolicies 4.710528e-
01
## TrailerPolicies                               TrailerPolicies 5.456894e-
01
## ThirdPartyInsuranceFirms             ThirdPartyInsuranceFirms 6.885267e-
01
## DeliveryVanPolicies                       DeliveryVanPolicies 8.415373e-
01
## AgriculturalMachinesPolicies     AgriculturalMachinesPolicies 8.545754e-
01
## LorryPolicies                                   LorryPolicies 9.025744e-
01
## PrivateAccidentPolicies               PrivateAccidentPolicies 9.911198e-
01
##                                 Group
## CarPolicies                      High
## FirePolicies                     High
## BoatPolicies                     High
## PrivateThirdPartyInsurance       High
## SocialSecurityInsurancePolicies  High
## MotorcycleScooterPolicies        High
```

```
## FamilyAccidentPolicies              High
## MopedPolicies                     Medium
## SurfboardPolicies                 Medium
## BicyclePolicies                   Medium
## DisabilityInsurancePolicies          Low
## LifeInsurances                       Low
## ThirdPartyInsuraneAgriculture        Low
## TractorPolicies                      Low
## PropertyInsurancePolicies            Low
## TrailerPolicies                      Low
## ThirdPartyInsuranceFirms             Low
## DeliveryVanPolicies                  Low
## AgriculturalMachinesPolicies         Low
## LorryPolicies                        Low
## PrivateAccidentPolicies              Low
```

```
# Create the plot using line plot with the original axes
p <- ggplot(result_table_sorted, aes(x = Variable, y = p_value, group = Group
, color = Group)) +
  geom_line() +
  geom_point(size = 3) +
  labs(title = "Chi-squared Test Results",
       x = "Variable",
       y = "p-value",
       color = "Group") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "top") +
  ylim(c(-0.5, 1))


# Convert the ggplot object to an interactive plotly object
p <- ggplotly(p)


# Display the interactive plot
p
```


Chi-squared Test Results
Group
High
Medium
Low
p-value
1.0
0.5
0.0
-0.5
CarPolicies
FirePolicies
BoatPolicies
PrivateThirdPartyInsurance
SocialSecurityInsurancePolicies
MotorcycleScooterPolicies
FamilyAccidentPolicies
MopedPolicies
SurfboardPolicies
BicyclePolicies
DisabilityInsurancePolicies
LifeInsurances
ThirdPartyInsuraneAgriculture
TractorPolicies
PropertyInsurancePolicies
TrailerPolicies
ThirdPartyInsuranceFirms
DeliveryVanPolicies
AgriculturalMachinesPolicies
LorryPolicies
PrivateAccidentPolicies
Variable