ML Homework 3

Anil Akyildirim

5/3/2020

Introduction

Our new intern Kansales Ruffio analyzed a dataset car_eval_test and car_eval_train datasets test and training datasets we provided. The purpose of this document is to review his work, output of his analysis and provide feedback on how he can improve his approach to data science, identify the choice of his classifiers and compare the performance metrices.

```
library(VGAM)
```

```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
library(caret)
```

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

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:VGAM':
##
##     predictors
```

```
library(ggplot2)
library(gridExtra)
library(caret)
library(MASS)
library(MASS)
library(pROC)
```

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

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

```
## Loading required package: rpart
```

```
library(randomForest)
```

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

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

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

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

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

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

```
library(rpart)
library(RColorBrewer)
library(rattle)
```

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

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
## The following object is masked from 'package:VGAM':
##
##     wine
```

```
library(rpart.plot)
library(ipred)
library(gbm)
```

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

```
## Loaded gbm 2.1.5
```

```
library(randomForest)
library(xgboost)
```

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

```
##
## Attaching package: 'xgboost'
```

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

```
library(Matrix)
```

```
traindatafile<-'car_eval_train.csv'
trdata<-read.csv(traindatafile,head=T,sep=',')
car_eval<-trdata
names(car_eval)<-c("buying","maint","doors","persons","lug_boot","safety","class")
names(trdata)<-c("buying","maint","doors","persons","lug_boot","safety","class")
tstdatafile<-'car_eval_test.csv'
tstdata<-read.csv(tstdatafile,head=T,sep=',')
names(tstdata)<-names(car_eval)
x<-tstdata[,1:6]
y<-tstdata[[7]]
```

We will first quickly look at the dataset to understand predictor and response variables.

```
head(trdata)
```

```
##   buying maint doors persons lug_boot safety class
## 1   high   low 5more       2    small    med unacc
## 2    low   med     4       2      big    low unacc
## 3  vhigh vhigh 5more    more    small    low unacc
## 4    med   low     3       2    small    med unacc
## 5   high  high     2       2    small    low unacc
## 6    med vhigh     4       4      big   high   acc
```

```
str(trdata)
```

```
## 'data.frame':    1210 obs. of  7 variables:
##  $ buying  : Factor w/ 4 levels "high","low","med",..: 1 2 4 3 1 3 3 3 4 2 ...
##  $ maint   : Factor w/ 4 levels "high","low","med",..: 2 3 4 2 1 4 3 4 3 1 ...
##  $ doors   : Factor w/ 4 levels "2","3","4","5more": 4 3 4 2 1 3 4 1 4 4 ...
##  $ persons : Factor w/ 3 levels "2","4","more": 1 1 3 1 1 2 1 1 2 2 ...
##  $ lug_boot: Factor w/ 3 levels "big","med","small": 3 1 3 3 3 1 1 1 1 2 ...
##  $ safety  : Factor w/ 3 levels "high","low","med": 3 2 2 3 2 1 1 1 1 1 ...
##  $ class   : Factor w/ 4 levels "acc","good","unacc",..: 3 3 3 3 3 1 3 3 1 4 ...
```

```
unique(trdata$class)
```

```
## [1] unacc acc   vgood good
## Levels: acc good unacc vgood
```

```
colSums(is.na(trdata))
```

```
##   buying    maint    doors  persons lug_boot   safety    class
##        0        0        0        0        0        0        0
```

Multinomial Logistic Regression (vglm)

All variables are categorical, "class" is the response variable, our label where our models will tell us, based on the predictor variables we use, if the car is "unacc", "acc", "vgood", "good" condition. There are 4 levels in the label categorical variable.

```
vglm_model<-vglm(class~buying+maint+doors+persons+lug_boot+safety,family = "multinomial",data=car_eval)
```

```
## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 27 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 224 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 371 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 373 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 375 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 375 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred
```

```
## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 376 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred
```

```
## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 376 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred
```

```
## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 376 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred
```

```
## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 376 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred
```

```
## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon =
## control$wzepsilon): 376 diagonal elements of the working weights variable
## 'wz' have been replaced by 1.819e-12
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred
```

```
## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1
```

```
## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted
## probabilities numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra =
## extra): fitted values close to 0 or 1

## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, :
## iterations terminated because half-step sizes are very small

## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 =
## Xm2, : some quantities such as z, residuals, SEs may be inaccurate due to
## convergence at a half-step
```

The first model our intern Kansales Ruffio chose is multinomial logistic regression classification. He used all the independent variables. As always, we are choosing a function to fit our model and prediction, based on certain assumptions. In this case Ruffio chose below possible assumptions;

** The dependent variable (class) is a categorical variable and unordered variable. Unfortunately this may not be correct, If we look at the class variable, the categories can be ordered based on the car evaluation class. Such as a car that is evaluated as "vgood" might be higher or lower in the order than a car that is evaluated as "good". This does not mean that we can not use multiple logistic regression, but maybe ordinal logistic regression might have been preferred over multinomial.

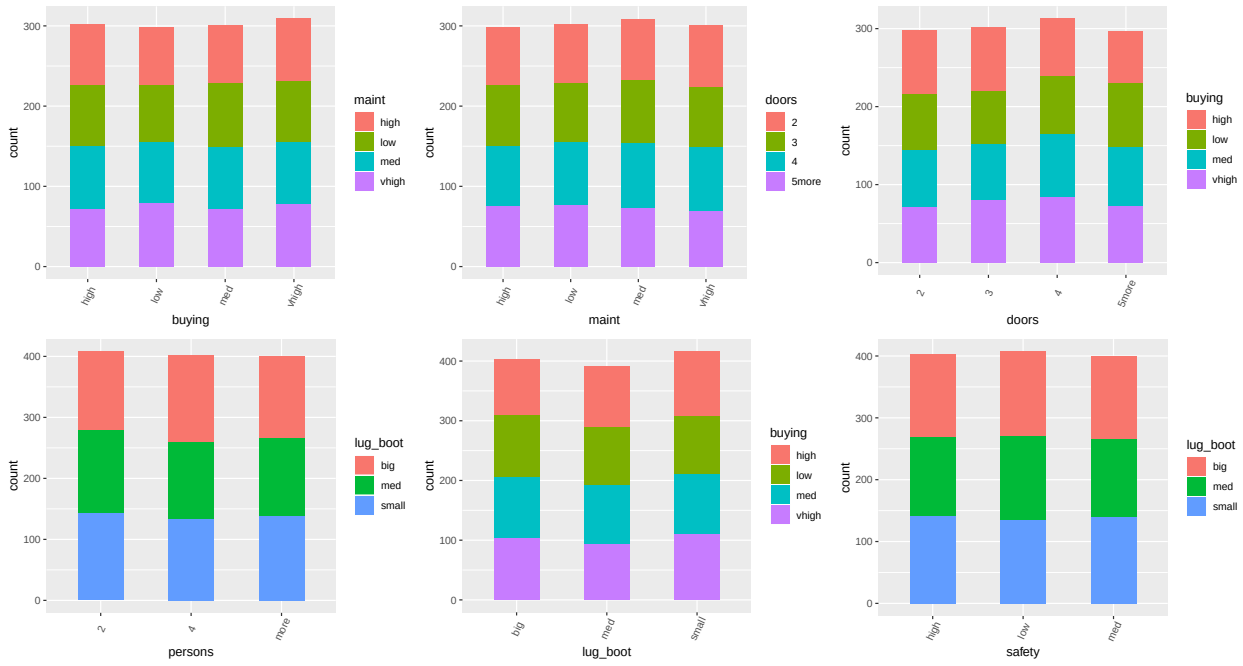
** A car can be evaulated only one of these categorical variable. This assumption is correct as we can only assign one value to the car evaluation. It will be either "acc", "good" "unacc", "vgood".

** The dependent(response) variable are typically coded as j=1,2,. . . .m. They dont have to be coded with numbers, they can be coded with their decriptions. But for convinience purposes we could have coded as numbers. If we coded numbers we would also assume that these codes and their manitude wouldnt be interpreted. (wouldnt get means to summarize and etc. . .)

**There needs to be independence of observeations and dependent variable(class) should have mutually exclusive.. We are assuming that the cars are completely independent and acc, good, unacc and vgood evaluations are mutually exclusive. This assumption can be acceptable.

** There should be no multicollinearity. Our possible independent variables; buying, maint, doors, persons, lug_boot and safety should be independent from each other. We dont see any analysis of distribution, skewness or outliers in the analysis. Since this is logistic regression, we dont need data to be normaly distributed but assuming he ran Chi-Squared test and confirmed independence and no multicollinearity.

```
s1 <- ggplot(trdata, aes(buying))+
  geom_bar(aes(fill=maint), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

s2 <- ggplot(trdata, aes(maint))+
  geom_bar(aes(fill=doors), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

s3 <- ggplot(trdata, aes(doors))+
  geom_bar(aes(fill=buying), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

s4 <- ggplot(trdata, aes(persons))+
  geom_bar(aes(fill=lug_boot), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

s5 <- ggplot(trdata, aes(lug_boot))+
  geom_bar(aes(fill=buying), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

s6 <- ggplot(trdata, aes(safety))+
  geom_bar(aes(fill=lug_boot), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))


grid.arrange(s1, s2, s3, s4, s5, s6,nrow=2)
```



Let's look at the summary of Ruffio's vglm model.

```r
summary(vglm_model)
```

```
## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred

## Warning in temp1@family@linkinv(eta = temp1@predictors, extra =
## temp1@extra): fitted probabilities numerically 0 or 1 occurred
```

```
##
## Call:
## vglm(formula = class ~ buying + maint + doors + persons + lug_boot +
##     safety, family = "multinomial", data = car_eval)
##
## Pearson residuals:
##                      Min        1Q     Median         3Q    Max
## log(mu[,1]/mu[,4]) -3.800 -0.025852 -9.558e-04 -1.957e-07  2.188
## log(mu[,2]/mu[,4]) -3.962 -0.010204 -7.031e-05 -2.611e-07  1.465
## log(mu[,3]/mu[,4]) -2.187 -0.001533  3.521e-04  1.028e-02 19.458
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept):1    15.5707     3.8411   4.054 5.04e-05 ***
## (Intercept):2     0.5564     4.1446   0.134 0.893204
## (Intercept):3    28.3663     3.4304   8.269  < 2e-16 ***
## buyinglow:1     -10.1854     1.9551  -5.210 1.89e-07 ***
## buyinglow:2      -1.9084     2.4860  -0.768 0.442697
## buyinglow:3     -14.2855     2.0469  -6.979 2.97e-12 ***
## buyingmed:1      -7.6260     1.8127  -4.207 2.59e-05 ***
## buyingmed:2      -1.4077     2.4451  -0.576 0.564821
```

```
## buyingmed:3     -11.4324     1.9015  -6.012 1.83e-09 ***
## buyingvhigh:1     0.2940     2.3291   0.126 0.899533
## buyingvhigh:2     0.0540     3.4628   0.016 0.987559
## buyingvhigh:3     2.4934     2.3670   1.053 0.292165
## maintlow:1       -5.0619     1.0706  -4.728 2.27e-06 ***
## maintlow:2        2.5082     2.0739   1.209 0.226511
## maintlow:3       -7.7709     1.1776  -6.599 4.14e-11 ***
## maintmed:1       -3.5801     0.9777  -3.662 0.000250 ***
## maintmed:2        2.0748     2.0874   0.994 0.320254
## maintmed:3       -6.3679     1.0986  -5.796 6.78e-09 ***
## maintvhigh:1      5.6087     1.8224      NA       NA
## maintvhigh:2      4.6580     3.1947   1.458 0.144823
## maintvhigh:3      8.4069     1.8888   4.451 8.55e-06 ***
## doors3:1         -0.9764     0.9297  -1.050 0.293611
## doors3:2         -0.4874     0.8953  -0.544 0.586196
## doors3:3         -2.5582     1.0317  -2.480 0.013150 *
## doors4:1         -2.7918     0.9714  -2.874 0.004054 **
## doors4:2         -1.6956     0.9516  -1.782 0.074782 .
## doors4:3         -4.9413     1.0730  -4.605 4.12e-06 ***
## doors5more:1     -2.7409     0.9885  -2.773 0.005557 **
## doors5more:2     -1.6924     0.9528  -1.776 0.075706 .
## doors5more:3     -4.9125     1.0845  -4.530 5.90e-06 ***
## persons4:1       -5.4929     2.8045  -1.959 0.050157 .
## persons4:2       -2.0599     2.3315  -0.884 0.376958
## persons4:3      -19.9138     2.2207  -8.967  < 2e-16 ***
## personsmore:1    -6.2285     2.8493  -2.186 0.028816 *
## personsmore:2    -2.8407     2.3794  -1.194 0.232522
## personsmore:3   -20.4966     2.2714  -9.024  < 2e-16 ***
## lug_bootmed:1     1.9964     0.7414   2.693 0.007088 **
## lug_bootmed:2     1.1262     0.7212   1.562 0.118399
## lug_bootmed:3     3.0709     0.8388   3.661 0.000251 ***
## lug_bootsmall:1   8.5329     1.7158   4.973 6.58e-07 ***
## lug_bootsmall:2   5.6730     1.6215   3.499 0.000468 ***
## lug_bootsmall:3  12.4169     1.7870   6.948 3.69e-12 ***
## safetylow:1       6.3158     2.9808   2.119 0.034107 *
## safetylow:2       2.6934     2.5855   1.042 0.297541
## safetylow:3      21.3749     2.3467   9.108  < 2e-16 ***
## safetymed:1       8.3315     1.6866   4.940 7.81e-07 ***
## safetymed:2       6.2937     1.5981   3.938 8.21e-05 ***
## safetymed:3      11.1417     1.7350   6.422 1.35e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Names of linear predictors: log(mu[,1]/mu[,4]), log(mu[,2]/mu[,4]),
## log(mu[,3]/mu[,4])
##
## Residual deviance: 346.5269 on 3582 degrees of freedom
##
## Log-likelihood: -173.2635 on 3582 degrees of freedom
##
## Number of Fisher scoring iterations: 15
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## '(Intercept):3', 'maintvhigh:1', 'lug_bootsmall:1', 'lug_bootsmall:2', 'safetylow:3', 'safetymed:1',
```

```
##
##
## Reference group is level  4  of the response
```

When we ran "vglm_model<-vglm(class~buying+maint+doors+persons+lug_boot+safety,family ="multinomial",data=car_eval)", we see a message that states certain vectors are replaced with other ones and "probabilities numerically 0 or 1" occured.

This might mean that, the classification problem we are dealing with will predict absolute probabilities such as 0 and 1. This also means that, for the particular value of the explanatory variables, there can be only two outcomes. My suggestion here would be, instead of performing the vglm multonimial logistic regression, maybe we could try stepwise logit to see, out of buying, maint, doors, persons, lug_boot and safety, which variable is creating this perfect separation.

Other option would be to increase a parameter to change the value of the iterations for the fishing scoring. (We are also getting this "SEs may be inaccurate due to convergence at a half-step." so fishing score is not enough for convergence). In this case the Number of Fisher scoring iterations is 15 , we can try to increase this to 50 maybe.

We can also see this in Warning "Warning: Hauck-Donner effect detected in the following estimate(s): '(Intercept):3', 'maintvhigh:1', 'lug_bootsmall:1', 'lug_bootsmall:2', 'safetylow:3', 'safetymed:1', 'safetymed:2' " This also means "Perfect seperation"

The idea to fix this issue is to add penalization to the regression model.

With Multinomila Logistic Regression, we are making little assumptions on the probability and applying discriminative algorithm(discriminative counterpart of Naive Bayes). We are using maximum likelyhod estimate, which we want to choose parameters that we maximize the conditional likelihood. This is the probability of the observered response variable values (class) in the training data trdata on the dependent varailes (Features, buying, maint, doors, persons, lug_boot, safety). In this case, we are not looking at the data distributions, we model the P(class|dependent variables) directly. Let's see how Ruffio found the probabilities, predicted values and confusion matrix for the predictions.

```
vglm_class_probabilities<-predict(vglm_model,tstdata[,1:6],type="response")

vglm_predicted_class<-apply(vglm_class_probabilities,1,which.max)

vglm_pred<-c()
vglm_pred[which(vglm_predicted_class=="1")]<-levels(y)[1]
vglm_pred[which(vglm_predicted_class=="2")]<-levels(y)[2]
vglm_pred[which(vglm_predicted_class=="3")]<-levels(y)[3]
vglm_pred[which(vglm_predicted_class=="4")]<-levels(y)[4]

vglm_mtab<-table(vglm_pred,tstdata[[7]])
(vglm_cmx<-confusionMatrix(table(vglm_pred,tstdata[[7]])))
```

```
## Confusion Matrix and Statistics
##
##
## vglm_pred acc good unacc vgood
##     acc   106    4    15     0
##     good    3   17     0     0
##     unacc  10    0   337     0
##     vgood   2    2     0    22
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9305
##                  95% CI : (0.9051, 0.9509)
##     No Information Rate : 0.6795
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8566
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: acc Class: good Class: unacc Class: vgood
## Sensitivity              0.8760     0.73913       0.9574      1.00000
## Specificity              0.9521     0.99394       0.9398      0.99194
## Pos Pred Value           0.8480     0.85000       0.9712      0.84615
## Neg Pred Value           0.9618     0.98795       0.9123      1.00000
## Prevalence               0.2336     0.04440       0.6795      0.04247
## Detection Rate           0.2046     0.03282       0.6506      0.04247
## Detection Prevalence     0.2413     0.03861       0.6699      0.05019
## Balanced Accuracy        0.9141     0.86653       0.9486      0.99597
```

```
(vglm_accuracy<-sum(diag(vglm_mtab))/sum(vglm_mtab))
```

```
## [1] 0.9305019
```

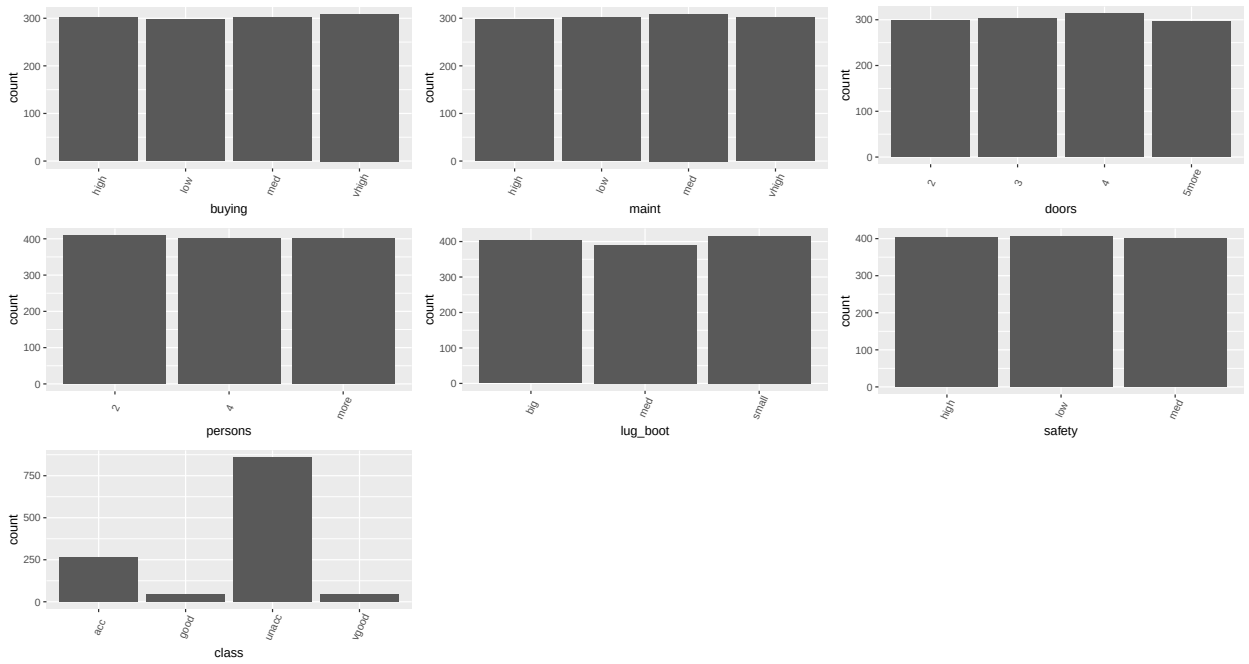
We see that the accuracy of the model is high with 93% accurate. P-value is really low. Since our output is more than two classes (multinomial) we see that we have 4 different combinations of predicted and actual values. Out of all the classed the model predicts 93% correctly. True positive for acc is 106, True Positive for good is 17, True positive for unacc is 337 and True positive for vgood is 22. True positive rate (sensitivity) is 100% for vgood explanatory variables. As previously mentioned, we might want to add penalization to the model. Kappa score is 85% which indicates our model performed well compared to how it would have performed by chance. This also indicates there is a big difference between the accuracy and the null error rate. When we look at the specifity , we also see high True-Negative Rate for each classs.The proportion of observed negatives that were predicted to be negatives(in other class for each class). Again, we are seeing very high (99%) rate for vgood.

Since we have more than 2 classes, the true positive and negative measures may not be that meaningful, so we focus on accuracy and error in this case.

Linear Discriminant Analysis (lda)

Ruffio's second model is using Linear Discriminant Analysis. This linear classification rule is preferred if the response categorical variable has more than two classes which is the case in our data set. In order to apply linear discriminant classification rule Ruffio made certain assumptions. The first one is that, the data is Gaussian(mean and variance). Each variable is shaped like a bell curve when we plot them. The second one is that, each variable has the same variance and the values of each variable vary around the same mean. The assumption of that everything must belong to one part or the other and nothing can belong simulataneously to both parts in the car class evaluation is met. Let's look at the distribution of each variable. (We have done a similar one earlier prior to find correlation betweeen independent variables).

```
a1 <- ggplot(trdata, aes(buying))+
  geom_bar() +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

a2 <- ggplot(trdata, aes(maint))+
  geom_bar() +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

a3 <- ggplot(trdata, aes(doors))+
  geom_bar() +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

a4 <- ggplot(trdata, aes(persons))+
  geom_bar() +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

a5 <- ggplot(trdata, aes(lug_boot))+
  geom_bar() +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

a6 <- ggplot(trdata, aes(safety))+
  geom_bar() +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

a7 <- ggplot(trdata, aes(class))+
  geom_bar() +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))

grid.arrange(a1, a2, a3, a4, a5, a6, a7, nrow=3)
```



We see the distribution of the explanatory variable is uniformly distributed (not a bell shape at all), however

the class variable distribution is not uniform or normaly distributed. We might want to consider transformation (logm root, box-cox), remove any outliers.

His purpose is to find the linear combinations of the explanatory variables (buying, maint, doors, persons, lug_boot, safety, class) that gives the best possible seperation between the class(acc, good, unacc, vgood). We have 4 levels in our class that we are trying to predict. The maximum number of discriminant functions that can separate the classes by the evaluation is the minimum of 3 and 6(explanatory variables. The minimum between 2 and 6 is 2. We can find the most 3 useful discriminant functions to seperate the classes using 6 explanatory variables. Let's see Ruffio's model.

```
lda_model<-lda(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval)
lda_model
```

```
## Call:
## lda(class ~ buying + maint + doors + persons + lug_boot + safety,
##     data = car_eval)
##
## Prior probabilities of groups:
##        acc       good      unacc      vgood
## 0.21735537 0.03801653 0.70909091 0.03553719
##
## Group means:
##       buyinglow buyingmed buyingvhigh  maintlow  maintmed maintvhigh
## acc   0.2129278 0.3003802   0.1787072 0.2357414 0.3117871  0.1901141
## good  0.6086957 0.3913043   0.0000000 0.6739130 0.3260870  0.0000000
## unacc 0.2191142 0.2179487   0.3053613 0.2214452 0.2284382  0.2925408
## vgood 0.6046512 0.3953488   0.0000000 0.4418605 0.3720930  0.0000000
##          doors3    doors4 doors5more  persons4 personsmore lug_bootmed
## acc   0.2699620 0.2471483  0.2813688 0.5247148   0.4752852   0.3307985
## good  0.2173913 0.2391304  0.3260870 0.6086957   0.3913043   0.3043478
## unacc 0.2482517 0.2599068  0.2261072 0.2505828   0.2738928   0.3170163
## vgood 0.1860465 0.3255814  0.3255814 0.4883721   0.5116279   0.4186047
##       lug_bootsmall safetylow safetymed
## acc       0.2813688  0.000000 0.4676806
## good      0.2826087  0.000000 0.5869565
## unacc     0.3834499  0.474359 0.2913753
## vgood     0.0000000  0.000000 0.0000000
##
## Coefficients of linear discriminants:
##                       LD1         LD2         LD3
## buyinglow     -0.7899038 -1.98704214 -0.36737114
## buyingmed     -0.6133282 -1.04180394 -0.16238537
## buyingvhigh    0.4355932 -0.59176194 -0.01144474
## maintlow      -0.6119818 -1.27754172 -1.31356931
## maintmed      -0.5299456 -0.42268878 -0.52860177
## maintvhigh     0.4591778 -0.01551135 -0.46145125
## doors3        -0.2624840  0.24330051  0.03049248
## doors4        -0.3186862 -0.01944816  0.25508499
## doors5more    -0.3906429  0.08827206  0.03051766
## persons4      -1.9705677  0.65221321 -0.37056331
## personsmore   -1.8686358  0.66391132  0.07943286
## lug_bootmed    0.2224485  0.12296904 -0.05970846
## lug_bootsmall  0.6744915  0.29261653 -0.72451690
## safetylow      2.2901725 -0.26578060 -0.82858685
```

```
## safetymed      0.7271670  0.72059048 -1.97971049
##
## Proportion of trace:
##    LD1    LD2    LD3
## 0.8846 0.0768 0.0386
```

Basically we can create the discriminant functions using these 3 LDvalues for each class. (Example: -0.7899 * buyyinglow -0.7899 * buyingmed. . . ..) When we look at the proportion of trace for each possible function, we can say that the percentage seperation achived for LDA1 (88%) is the highest. With LDA we are estimating probabilities that each test observation belongs to a class. The class that gets the highest probability is the output class and prediction is made. Let's look at Ruffio's probability estimates and predictions.

```
lda_class_probabilities<-predict(lda_model,tstdata[,1:6],type="response")

(lda_cmx<-table(lda_class_probabilities$class,tstdata[[7]]))
```

```
##
##         acc good unacc vgood
##   acc   107   12    25     9
##   good    5    9     0     0
##   unacc   9    0   327     0
##   vgood   0    2     0    13
```

```
lda_mtab<-table(lda_class_probabilities$class,tstdata[[7]])
(lda_accuracy<-sum(diag(lda_cmx))/sum(lda_cmx))
```

```
## [1] 0.8803089
```

```
lda_cmx<-confusionMatrix(table(lda_class_probabilities$class,tstdata[[7]]))
lda_cmx
```

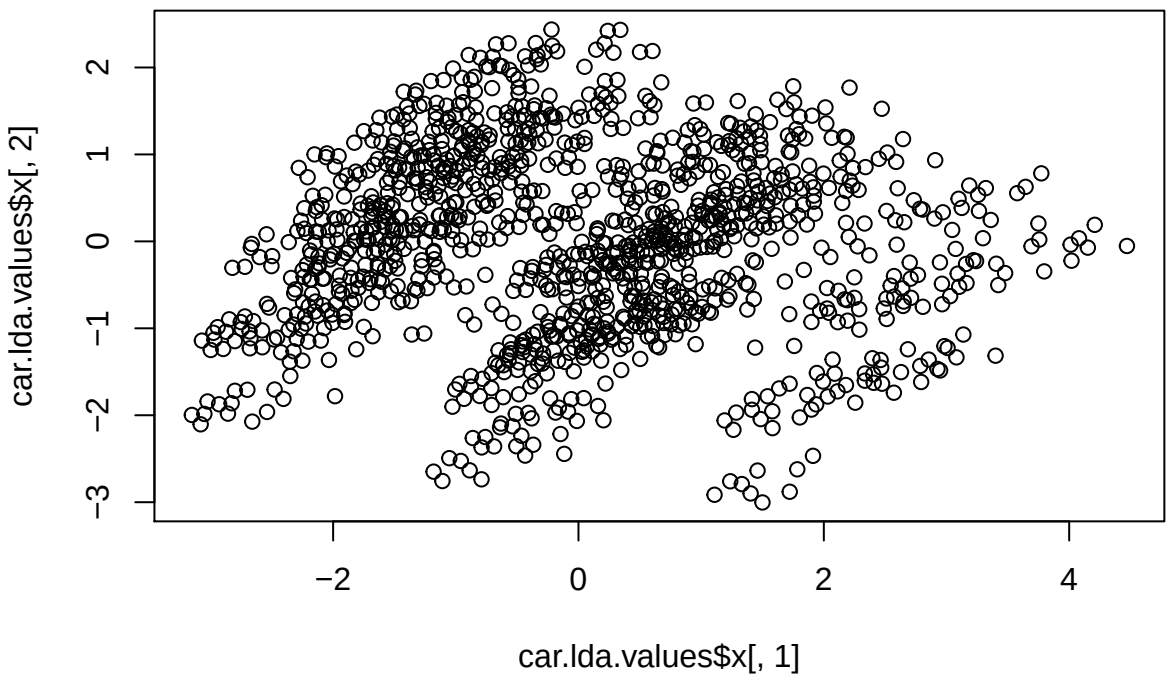
```
## Confusion Matrix and Statistics
##
##
##         acc good unacc vgood
##   acc   107   12    25     9
##   good    5    9     0     0
##   unacc   9    0   327     0
##   vgood   0    2     0    13
##
## Overall Statistics
##
##                Accuracy : 0.8803
##                  95% CI : (0.8492, 0.907)
##     No Information Rate : 0.6795
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7546
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: acc Class: good Class: unacc Class: vgood
## Sensitivity              0.8843     0.39130       0.9290      0.59091
## Specificity              0.8841     0.98990       0.9458      0.99597
## Pos Pred Value           0.6993     0.64286       0.9732      0.86667
## Neg Pred Value           0.9616     0.97222       0.8626      0.98211
## Prevalence               0.2336     0.04440       0.6795      0.04247
## Detection Rate           0.2066     0.01737       0.6313      0.02510
## Detection Prevalence     0.2954     0.02703       0.6486      0.02896
## Balanced Accuracy        0.8842     0.69060       0.9374      0.79344
```

We have an accuracy of 83%. The model predicts 83% correctly. True positive for acc is 107 for, True Positive for good is 9, True positive for unacc is 327 and True positive for vgood is 13. True positive rate (sensitivity) is really low for good (39%). Kappa score is 75% which is lower than the first model. Again, since we have more than 2 classes, the true positive and negative measures may not be that meaningful, so we focus on accuracy and error in this case. Accuracy is lower than the first model, however, as we mentioned earlier, we might want to apply a penalty to the multinomial logistic regression model we created earlier.

We can also suggest Ruffio to take the predictions and plot them to see the results in different groups. For example, create a scatter plot between groups of predictions.

```
car.lda.values <- predict(lda_model)
plot(car.lda.values$x[,1],car.lda.values$x[,2]) # make a scatterplot
```



Decision Tree(rpart)

In his third model, Ruffio is using decision tree classification model with rpart function. If we remember KNN's main assumption which is similar inputs have similar similar neighbours where it would imply that the data points are sprinkled across the spave, but instead they are in cluster or more or less homogenous class assignments, Decision tree exploits this assumption and takes further. In Decision tree assumptions, we are building a tree like structure that divides the space into regions with similar labels. Begining of the tree is the root. Another assumption is that we would like the feature set to be all categorical. (Which is the case in our dataset). We want the variables to be distributed recursively based on the explanatory feature set.

```
rpart_model<-rpart(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval)
rpart_class_probabilities<-predict(rpart_model,tstdata[,1:6],type="class")

(rpart_mtab<-table(rpart_class_probabilities,tstdata[[7]]))
```

```
##
## rpart_class_probabilities acc good unacc vgood
##                     acc   106    0    14     5
##                     good   10   20     1     0
##                     unacc   4    0   337     0
##                     vgood   1    3     0    17
```

```
rpart_cmx<-confusionMatrix(rpart_mtab)
(rpart_accuracy<-sum(diag(rpart_mtab))/sum(rpart_mtab))
```

```
## [1] 0.9266409
```

```
rpart_cmx
```

```
## Confusion Matrix and Statistics
##
##
## rpart_class_probabilities acc good unacc vgood
##                     acc   106    0    14     5
##                     good   10   20     1     0
##                     unacc   4    0   337     0
##                     vgood   1    3     0    17
##
## Overall Statistics
##
##                Accuracy : 0.9266
##                  95% CI : (0.9007, 0.9476)
##     No Information Rate : 0.6795
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8509
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

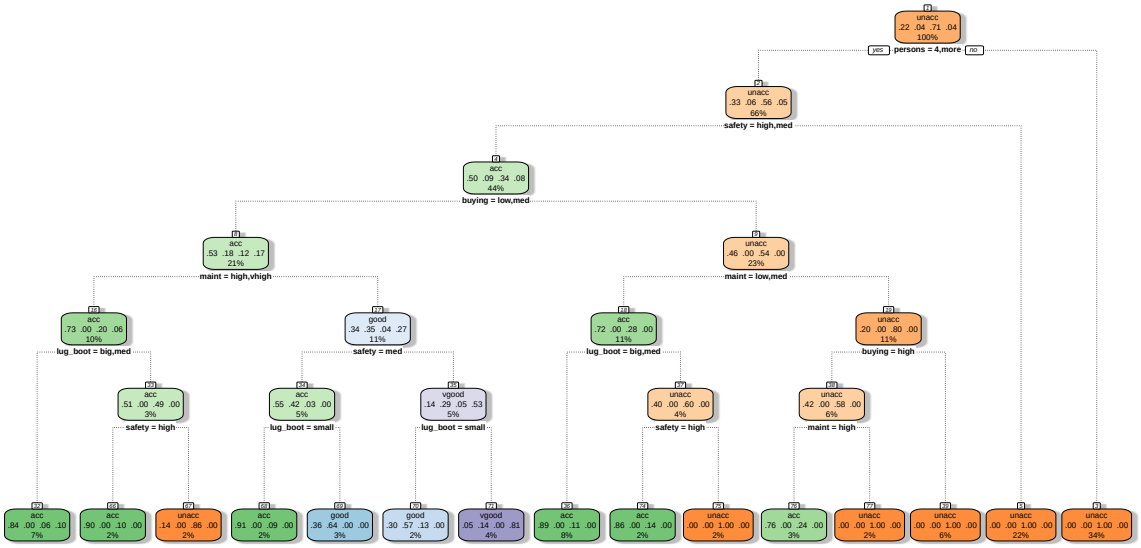
```
##                      Class: acc Class: good Class: unacc Class: vgood
## Sensitivity              0.8760     0.86957       0.9574      0.77273
## Specificity              0.9521     0.97778       0.9759      0.99194
## Pos Pred Value           0.8480     0.64516       0.9883      0.80952
## Neg Pred Value           0.9618     0.99384       0.9153      0.98994
## Prevalence               0.2336     0.04440       0.6795      0.04247
## Detection Rate           0.2046     0.03861       0.6506      0.03282
## Detection Prevalence     0.2413     0.05985       0.6583      0.04054
## Balanced Accuracy        0.9141     0.92367       0.9666      0.88233
```

```
rpart_model
```

```
## n= 1210
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1210 352 unacc (0.21735537 0.03801653 0.70909091 0.03553719)
##    2) persons=4,more 802 352 unacc (0.32793017 0.05735661 0.56109726 0.05361596)
##      4) safety=high,med 531 268 acc (0.49529190 0.08662900 0.33709981 0.08097928)
##        8) buying=low,med 254 119 acc (0.53149606 0.18110236 0.11811024 0.16929134)
##         16) maint=high,vhigh 124  33 acc (0.73387097 0.00000000 0.20161290 0.06451613)
##           32) lug_boot=big,med 83  13 acc (0.84337349 0.00000000 0.06024096 0.09638554) *
##           33) lug_boot=small 41  20 acc (0.51219512 0.00000000 0.48780488 0.00000000)
##             66) safety=high 20   2 acc (0.90000000 0.00000000 0.10000000 0.00000000) *
##             67) safety=med 21   3 unacc (0.14285714 0.00000000 0.85714286 0.00000000) *
##         17) maint=low,med 130  84 good (0.33846154 0.35384615 0.03846154 0.26923077)
##           34) safety=med 64  29 acc (0.54687500 0.42187500 0.03125000 0.00000000)
##             68) lug_boot=small 22   2 acc (0.90909091 0.00000000 0.09090909 0.00000000) *
##             69) lug_boot=big,med 42  15 good (0.35714286 0.64285714 0.00000000 0.00000000) *
##           35) safety=high 66  31 vgood (0.13636364 0.28787879 0.04545455 0.53030303)
##             70) lug_boot=small 23  10 good (0.30434783 0.56521739 0.13043478 0.00000000) *
##             71) lug_boot=big,med 43   8 vgood (0.04651163 0.13953488 0.00000000 0.81395349) *
##        9) buying=high,vhigh 277 128 unacc (0.46209386 0.00000000 0.53790614 0.00000000)
##         18) maint=low,med 138  38 acc (0.72463768 0.00000000 0.27536232 0.00000000)
##           36) lug_boot=big,med 91  10 acc (0.89010989 0.00000000 0.10989011 0.00000000) *
##           37) lug_boot=small 47  19 unacc (0.40425532 0.00000000 0.59574468 0.00000000)
##             74) safety=high 22   3 acc (0.86363636 0.00000000 0.13636364 0.00000000) *
##             75) safety=med 25   0 unacc (0.00000000 0.00000000 1.00000000 0.00000000) *
##         19) maint=high,vhigh 139  28 unacc (0.20143885 0.00000000 0.79856115 0.00000000)
##           38) buying=high 67  28 unacc (0.41791045 0.00000000 0.58208955 0.00000000)
##             76) maint=high 37   9 acc (0.75675676 0.00000000 0.24324324 0.00000000) *
##             77) maint=vhigh 30   0 unacc (0.00000000 0.00000000 1.00000000 0.00000000) *
##           39) buying=vhigh 72   0 unacc (0.00000000 0.00000000 1.00000000 0.00000000) *
##      5) safety=low 271   0 unacc (0.00000000 0.00000000 1.00000000 0.00000000) *
##    3) persons=2 408   0 unacc (0.00000000 0.00000000 1.00000000 0.00000000) *
```

We have 6 features (n=6) in our model. It looks like we selected unacc as the root of our tree, we split the first time based on persons value, further we split the same data set based on safety feature in the second node. Our accuracy is 92%, which is pretty good. However we should keep in mind that one of the drawbacks of decision trees is probability of overfitting. Our Kappa value is high as well.

```
fancyRpartPlot(rpart_model, caption = NULL)
```



We are using rpart package and by default this uses gini impurity to select the splits. We might also suggest Ruffio that he can use information gain when specifying the splits in the parms parameter. rpart also has a complexity of a tree measure which is defined with cp() parameter. Since decision trees tend to produce over-fitting, we might want to ensure that this parameter is not set to negative.

Bagging

The assumption for Ruffio to perform baggins is that we want to reduce the error and with that reduce the variance term. (Error = Variance + Bias + Noise). In order to do that, he is applying bagging which is easy to implement, reduces the variance.

```
bag_model<-bagging(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval)
bag_model
```

```
##
## Bagging classification trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = class ~ buying + maint + doors +
##     persons + lug_boot + safety, data = car_eval)
```

```
bag_class_probabilities<-predict(bag_model,tstdata[,1:6])#,type="response")
(bag_mtab<-table(bag_class_probabilities,tstdata[[7]]))
```

```
##
## bag_class_probabilities acc good unacc vgood
##                     acc   119    1     5     1
```

```
##                   good    1   22     1     1
##                   unacc   1    0   346     0
##                   vgood   0    0     0    20
```

```
(bag_cmx<-confusionMatrix(bag_mtab))
```

```
## Confusion Matrix and Statistics
##
##
## bag_class_probabilities acc good unacc vgood
##                   acc   119    1     5     1
##                   good    1   22     1     1
##                   unacc   1    0   346     0
##                   vgood   0    0     0    20
##
## Overall Statistics
##
##                Accuracy : 0.9788
##                  95% CI : (0.9623, 0.9894)
##     No Information Rate : 0.6795
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9561
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: acc Class: good Class: unacc Class: vgood
## Sensitivity              0.9835     0.95652       0.9830      0.90909
## Specificity              0.9824     0.99394       0.9940      1.00000
## Pos Pred Value           0.9444     0.88000       0.9971      1.00000
## Neg Pred Value           0.9949     0.99797       0.9649      0.99598
## Prevalence               0.2336     0.04440       0.6795      0.04247
## Detection Rate           0.2297     0.04247       0.6680      0.03861
## Detection Prevalence     0.2432     0.04826       0.6699      0.03861
## Balanced Accuracy        0.9829     0.97523       0.9885      0.95455
```

```
(bag_accuracy<-sum(diag(bag_mtab))/sum(bag_mtab))
```

```
## [1] 0.9787645
```

The model accuracy is higher than any of the other models we created. 98% accurate. And Kappa is also 95%. We are not seeing any specific method that is added to the model such as method or trControl. We might want to consider adding these. The assumption is the Resampling happened as Cross-Validation(10 folds). However, we can say that so far this is the best model Ruffio created.

Gradient Boosting Model

With the gradient boosting model, Ruffio wants to convert weak learners into strong learners. But, can weak learners be combined to generate strong learner with low bias? (Yes we can https://www.cs.princeton.edu/

~schapire/papers/strengthofweak.pdf). With boosting we create a ensemble classifier with iteration similar to gradient descent. Instead of updating the model parameters in each step, we are adding a function to the ensemble. With Gradient Boosted Regression Tree, we are applying classification rule.

With each new tree is a fit on a modified version of the original training data set (trdata)

```
nlev<-4 # number of classes
gbm_model<-gbm(class~buying+maint+doors+persons+lug_boot+safety,
data=car_eval,n.trees=5000,interaction.depth=nlev, shrinkage=0.001,bag.fraction=0.8,distribution="multi
```

In this model, Ruffio used all the explanatory variables with 5000 tree splits and for 4 classes (as class has 4 levels)

```
gbm_class_probabilities<-predict(gbm_model,tstdata[,1:6],n.trees=5000,type="response")
gbm_pred<-apply(gbm_class_probabilities,1,which.max)

gbm_predicted_class<-unlist(lapply(gbm_pred,FUN=function(x)levels(tstdata[[7]])[[x]]))

(gbm_mtab<-table(gbm_predicted_class,tstdata[[7]]))
```

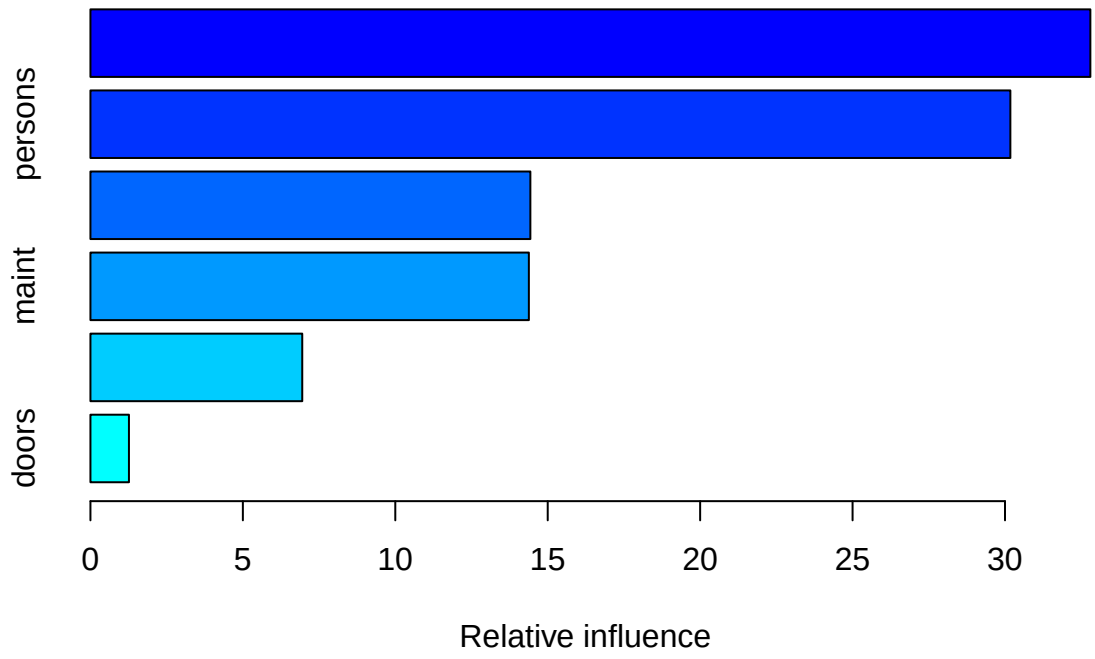
```
##
## gbm_predicted_class acc good unacc vgood
##               acc   115    0     5     1
##               good    4   20     1     0
##               unacc   2    0   346     0
##               vgood   0    3     0    21
```

```
(gbm_accuracy<-sum(diag(gbm_mtab))/sum(gbm_mtab))
```

```
## [1] 0.969112
```

This model generates 5000 trees and shrinkage parameter lambda is 0.001 which is also can be seen as the learning rate. Depth is the total splits we want to do which is 4 splits.

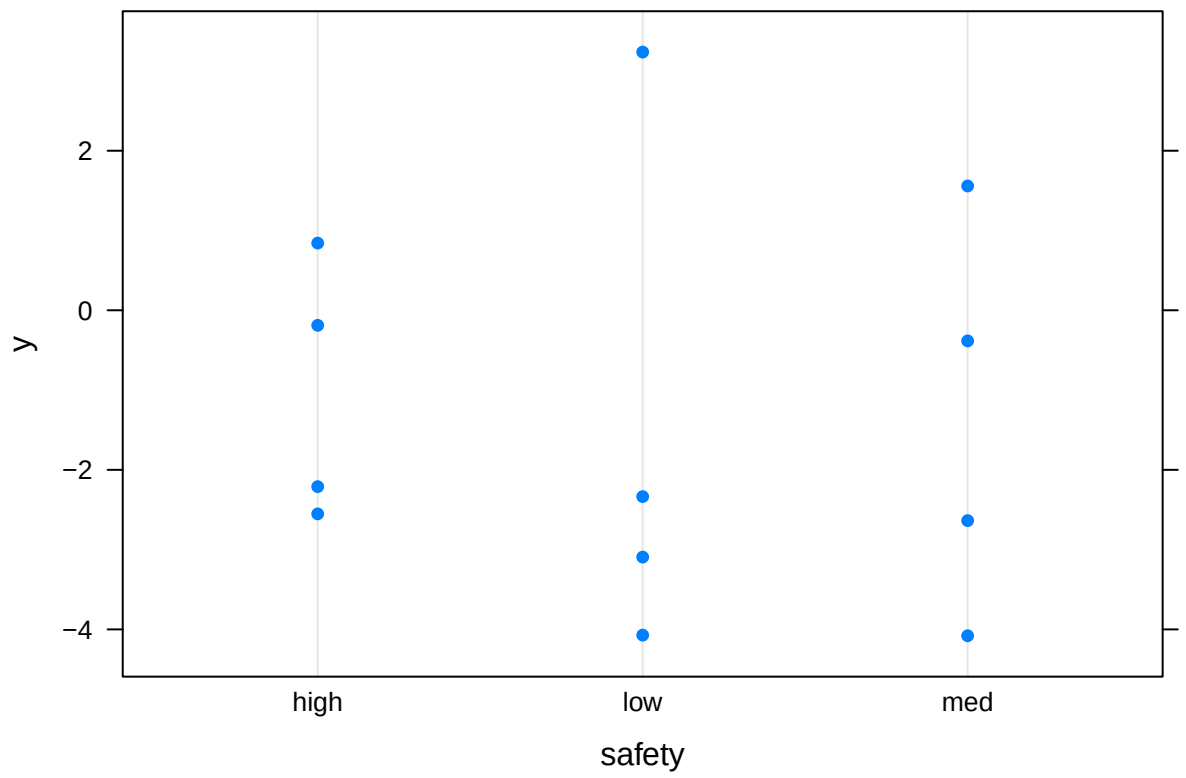
```
summary(gbm_model)
```



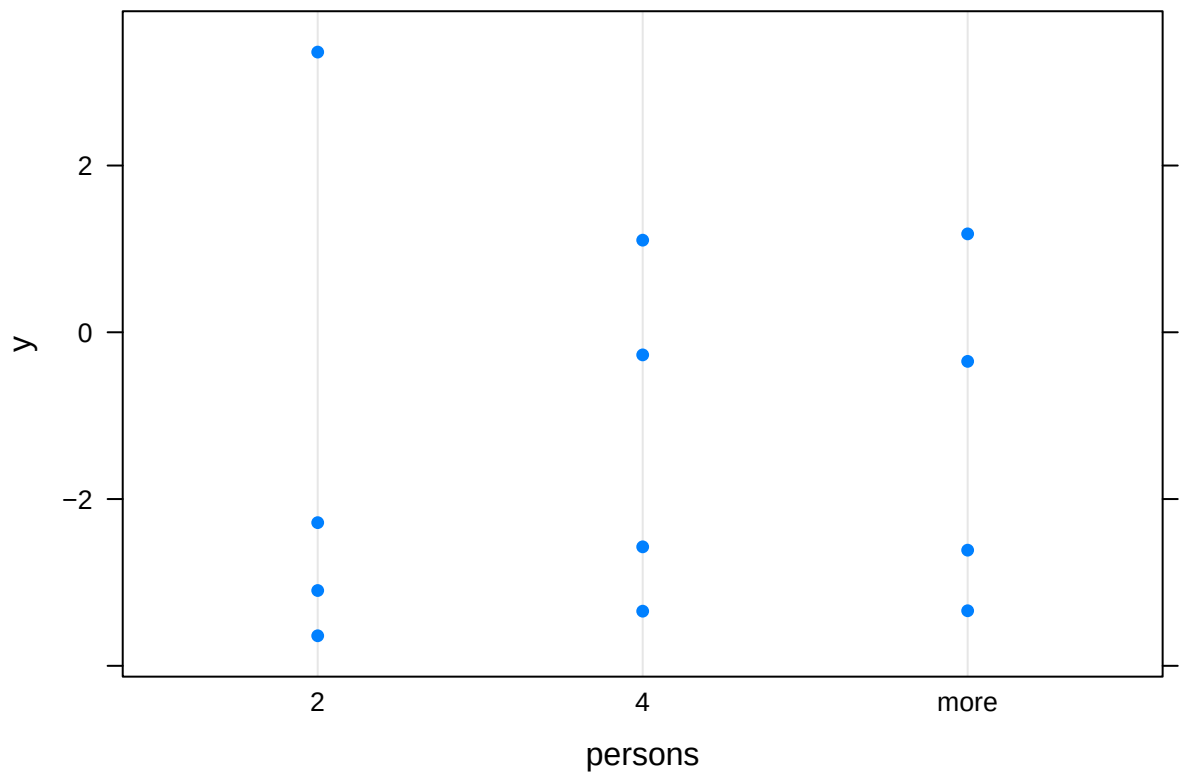

```
##                var   rel.inf
## safety     safety 32.800497
## persons   persons 30.175971
## buying     buying 14.430574
## maint       maint 14.381781
## lug_boot lug_boot  6.948451
## doors       doors  1.262727
```

As we can see the importance of the features, we see doors is not that important and doesnt influence the classification that much.

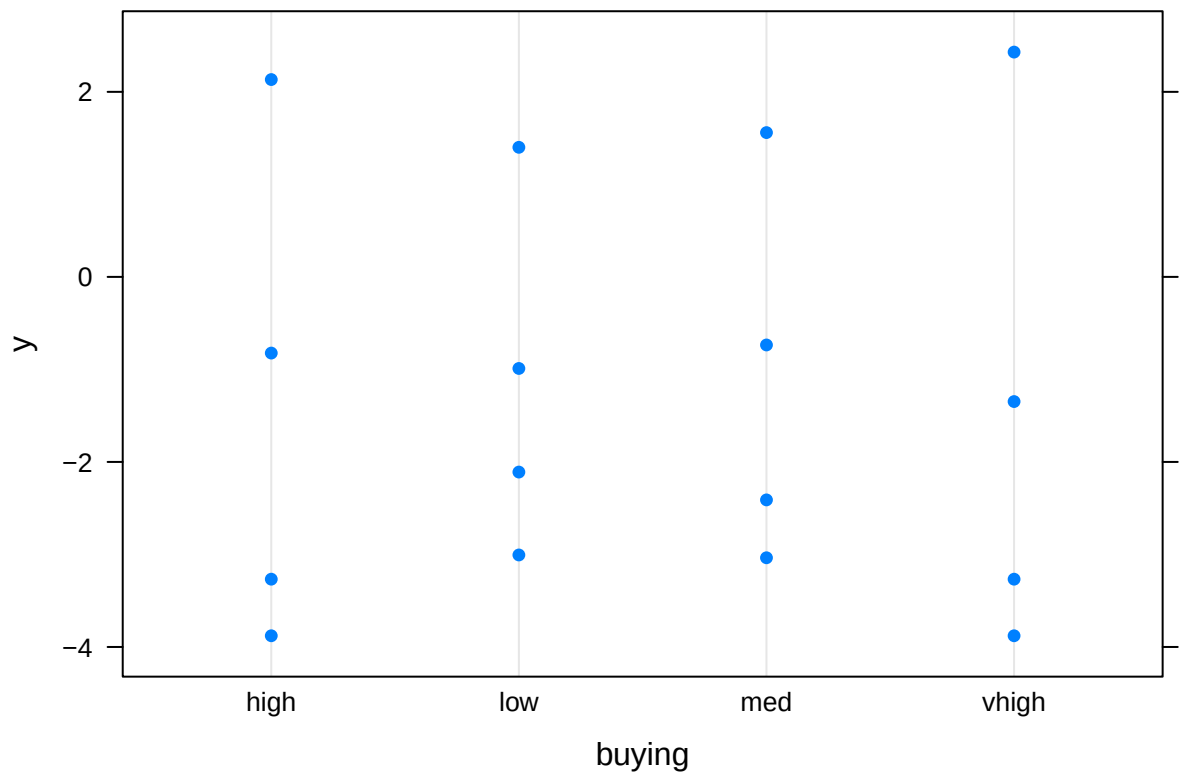
```
plot(gbm_model, i="safety")
```



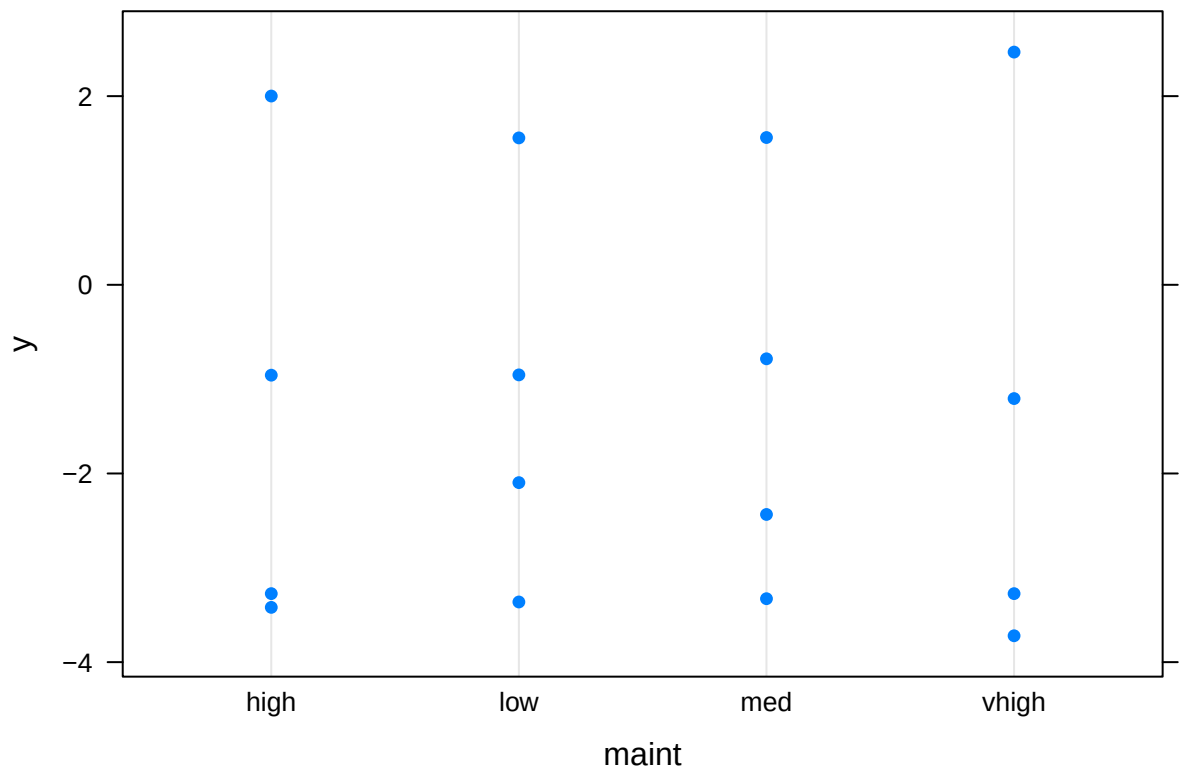

```
plot(gbm_model, i="persons")
```



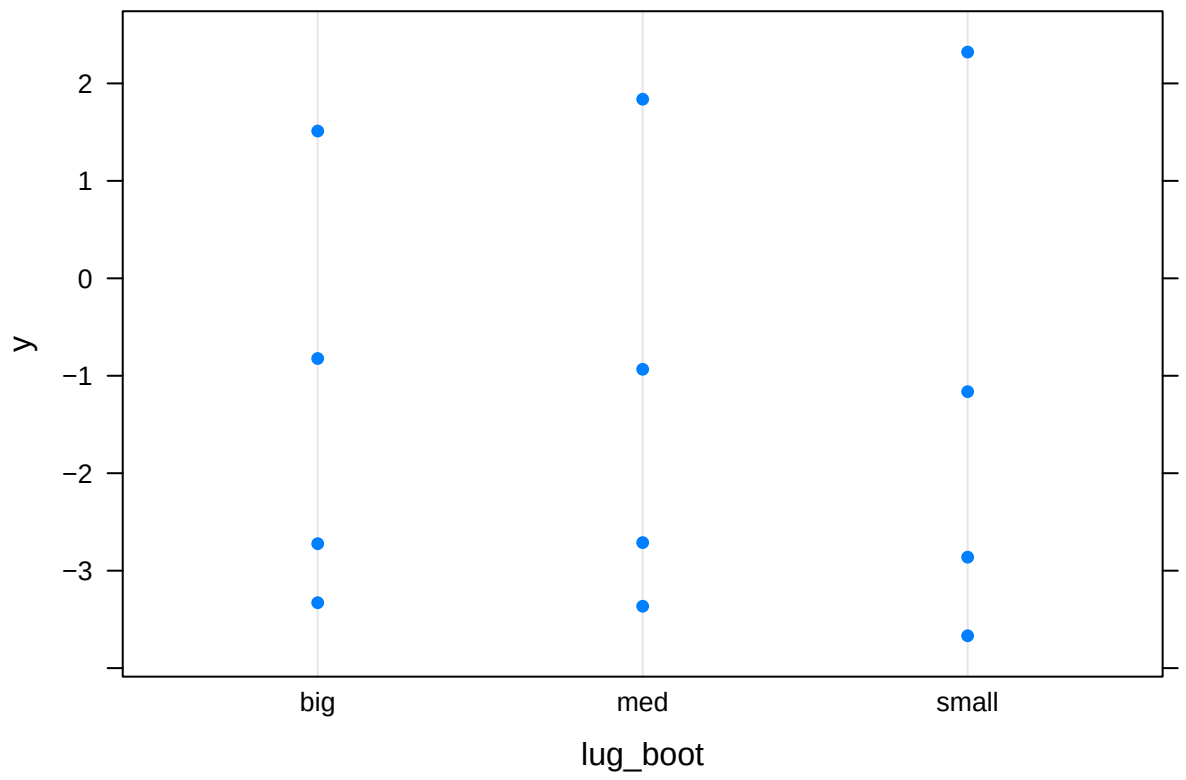

```
plot(gbm_model, i="buying")
```



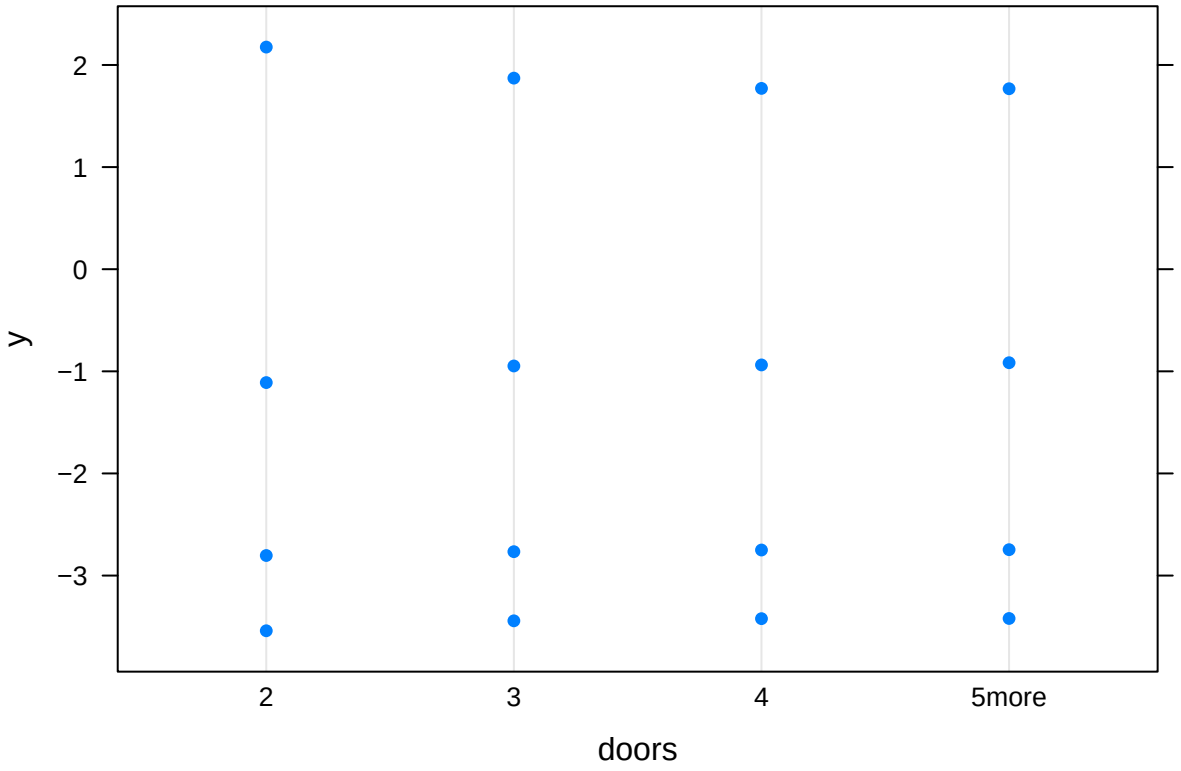

```
plot(gbm_model, i="maint")
```




```
plot(gbm_model, i="lug_boot")
```




```
plot(gbm_model, i="doors")
```



Accuracy of the model is the highest so far 96%. As we increase the trees , we will notice that the accuracy will increase.

Gradient Boosting Model 2

```
gbm_model2<-gbm(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval,n.trees=5000,interaction
shrinkage=0.001,bag.fraction=0.8,distribution="multinomial",
verbose=FALSE,n.cores=4)
gbm_class_probabilities2<-predict(gbm_model2,tstdata[,1:6],n.trees=5000,type="response")
gbm_pred2<-apply(gbm_class_probabilities2,1,which.max)
gbm_pred2[which(gbm_pred2=="1")]<-levels(tstdata[[7]])[1]
gbm_pred2[which(gbm_pred2=="2")]<-levels(tstdata[[7]])[2]
gbm_pred2[which(gbm_pred2=="3")]<-levels(tstdata[[7]])[3]
gbm_pred2[which(gbm_pred2=="4")]<-levels(tstdata[[7]])[4]
gbm_pred2<-as.factor(gbm_pred2)
l<-union(gbm_pred2,tstdata[[7]])
(gbm_mtab2<-table(factor(gbm_pred2,l),factor(tstdata[[7]],l)))
```

```
##
##          unacc acc vgood good
##   unacc    345   2     0    0
##   acc        6 115     1    0
##   vgood      0   0    21    3
##   good       1   4     0   20
```

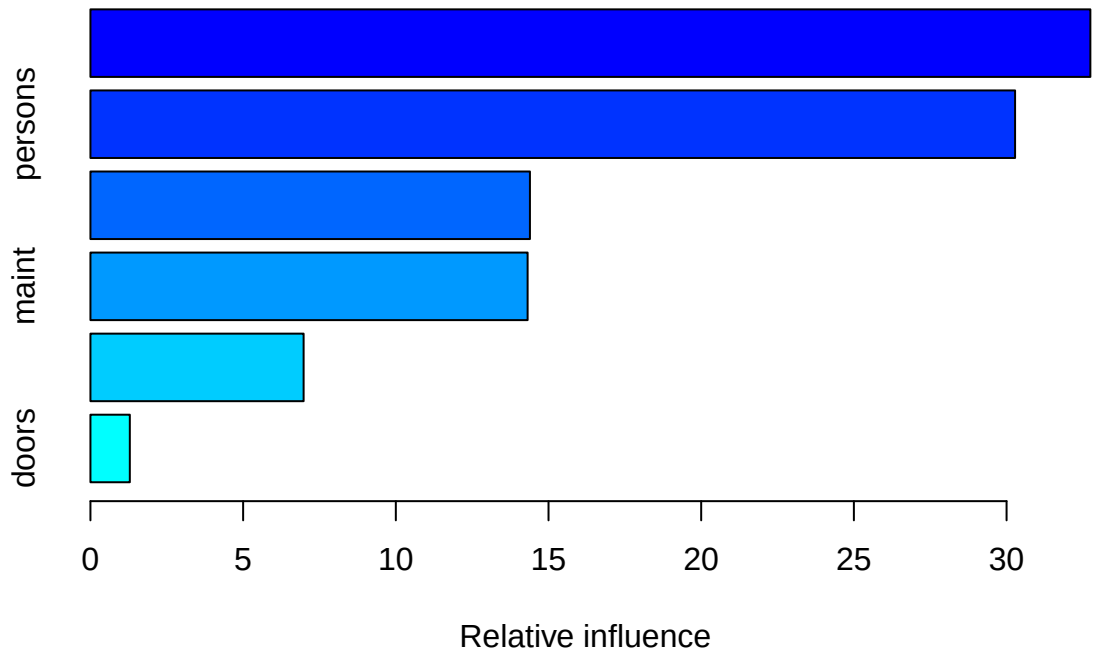
```
(gbm_accuracy2<-sum(diag(gbm_mtab2))/sum(gbm_mtab2))
```

```
## [1] 0.9671815
```

```
(gbm_cmx2<-confusionMatrix(gbm_mtab2))
```

```
## Confusion Matrix and Statistics
##
##
##          unacc acc vgood good
##   unacc   345   2     0    0
##   acc       6 115     1    0
##   vgood     0   0    21    3
##   good      1   4     0   20
##
## Overall Statistics
##
##                Accuracy : 0.9672
##                  95% CI : (0.948, 0.9808)
##     No Information Rate : 0.6795
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9324
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: unacc Class: acc Class: vgood Class: good
## Sensitivity                0.9801     0.9504      0.95455     0.86957
## Specificity                0.9880     0.9824      0.99395     0.98990
## Pos Pred Value             0.9942     0.9426      0.87500     0.80000
## Neg Pred Value             0.9591     0.9848      0.99798     0.99391
## Prevalence                 0.6795     0.2336      0.04247     0.04440
## Detection Rate             0.6660     0.2220      0.04054     0.03861
## Detection Prevalence       0.6699     0.2355      0.04633     0.04826
## Balanced Accuracy          0.9840     0.9664      0.97425     0.92973
```

```
summary(gbm_model2)
```




```
##                var   rel.inf
## safety      safety 32.743468
## persons    persons 30.278710
## buying      buying 14.391931
## maint        maint 14.315719
## lug_boot lug_boot  6.980317
## doors        doors  1.289855
```

We see the model accuracy is similar to the first GBM model. In this case we added "multinomial" distribution. The assumption is after evaluating the first tree in the model, we increase the weights of the obversvations and increase the ones that are difficult to classif the car evaluation class, and give lower weights to the ones that are easy to classify. This creates the imprvement.

The GBM Model is proine to overfitting so we do want to applu hyperparameter tunning. We want to tune the number of iterations to stop early if required. We can suggest Ruffino to find the test error for certain number of trees in sequence such as 100, 1000, up to 10000 to see the performance pattern. We can suggest him to apply specific arguement called "c.v folds" while training the model. We can also define the number of iterations (gbm.perf()).

Gradient Boosting Model 3

```
nlev<-5 # number of classes+1
gbm_model3<-gbm(class~buying+maint+doors+persons+lug_boot+safety,
```

```
data=car_eval,n.trees=5000,interaction.depth=nlev,
    shrinkage=0.001,bag.fraction=0.8,distribution="multinomial",verbose=FALSE,n.cores=4)
gbm_class_probabilities3<-predict(gbm_model3,tstdata[,1:6],n.trees=5000,type="response")
gbm_pred3<-apply(gbm_class_probabilities3,1,which.max)
##############
gbm_pred3[which(gbm_pred3=="1")]<-levels(tstdata[[7]])[1]
gbm_pred3[which(gbm_pred3=="2")]<-levels(tstdata[[7]])[2]
gbm_pred3[which(gbm_pred3=="3")]<-levels(tstdata[[7]])[3]
gbm_pred3[which(gbm_pred3=="4")]<-levels(tstdata[[7]])[4]
gbm_pred3<-as.factor(gbm_pred3)
l<-union(gbm_pred3,tstdata[[7]])
(gbm_mtab3<-table(factor(gbm_pred3,l),factor(tstdata[[7]],l)))
```

```
##
##         unacc acc vgood good
##   unacc   347   1     0    0
##   acc       4 117     1    1
##   vgood     0   0    21    3
##   good      1   3     0   19
```

```
(gbm_accuracy3<-sum(diag(gbm_mtab3))/sum(gbm_mtab3))
```

```
## [1] 0.972973
```

```
(gbm_cmx3<-confusionMatrix(gbm_mtab3))
```

```
## Confusion Matrix and Statistics
##
##
##         unacc acc vgood good
##   unacc   347   1     0    0
##   acc       4 117     1    1
##   vgood     0   0    21    3
##   good      1   3     0   19
##
## Overall Statistics
##
##                Accuracy : 0.973
##                  95% CI : (0.9551, 0.9851)
##     No Information Rate : 0.6795
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9442
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: unacc Class: acc Class: vgood Class: good
## Sensitivity                0.9858     0.9669      0.95455     0.82609
## Specificity                0.9940     0.9849      0.99395     0.99192
## Pos Pred Value             0.9971     0.9512      0.87500     0.82609
```

```
## Neg Pred Value              0.9706     0.9899      0.99798     0.99192
## Prevalence                  0.6795     0.2336      0.04247     0.04440
## Detection Rate              0.6699     0.2259      0.04054     0.03668
## Detection Prevalence        0.6718     0.2375      0.04633     0.04440
## Balanced Accuracy           0.9899     0.9759      0.97425     0.90900
```

```
###################
#gbm_predicted_class3<-unlist(lapply(gbm_pred3,FUN=function(x)levels(tstdata[[7]])[[x]]))

#(gbm_mtab3<-table(gbm_predicted_class3,tstdata[[7]]))
#(gbm_accuracy3<-sum(diag(gbm_mtab3))/sum(gbm_mtab3))
#(gbm_cmx3<-confusionMatrix(gbm_mtab3))
```

With the 3rth Gradient Boosting Model, the accuracy went down slightly to 97% , along with the Kappa value(94%). The difference in this model is that we are defining number of cores as 4. We dont know if we neccessarily have to define this. We re asking cross-validation loop to attemp different CV folds to different cores.

Random Forest

The idea is that we will aggregate the predictions made by multiple different decision trees of varying depth. It is extension of classification trees. Ruffio didnt need to make much assumptions as random forests are non parametrci and can handle skewness and multi modal data as well as categorical data that are ordinal and non-ordinal. These not complicated assumptions met by our data set. Random Fores tend to be very accurate but can overfit data sets.

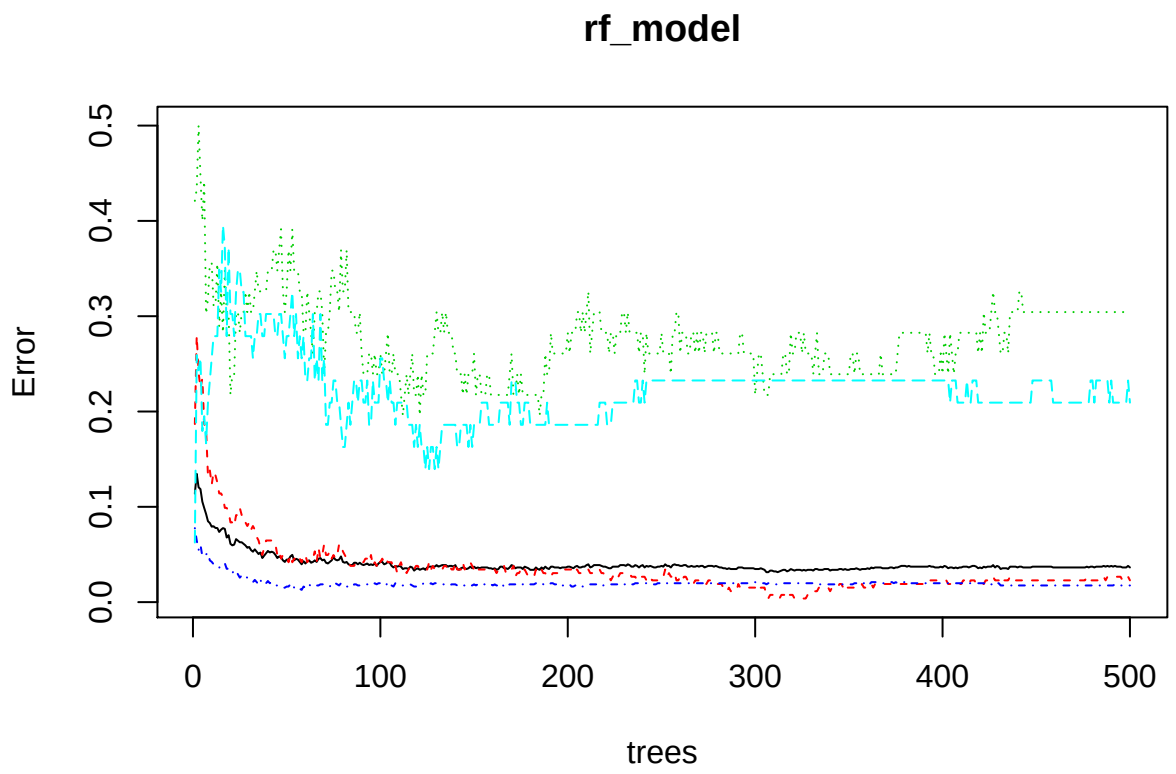
```
rf_model<-randomForest(class~buying+maint+doors+persons+lug_boot+safety, data=car_eval)
summary(rf_model)
```

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted       1210   factor numeric
## err.rate        2500   -none- numeric
## confusion         20   -none- numeric
## votes           4840   matrix numeric
## oob.times       1210   -none- numeric
## classes            4   -none- character
## importance         6   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y               1210   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
rf_model
```

```
##
## Call:
##  randomForest(formula = class ~ buying + maint + doors + persons +      lug_boot + safety, data = ca
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.64%
## Confusion matrix:
##       acc good unacc vgood class.error
## acc   257    2     2     2  0.02281369
## good   11   32     0     3  0.30434783
## unacc  15    0   843     0  0.01748252
## vgood   9    0     0    34  0.20930233
```

```
plot(rf_model)
```



As we increase the trees, we see for each class the error tends to go down. He defined the number of trees as 500 which seems to be ok.

```
rf_pred<-predict(rf_model,tstdata[,1:6])
rf_mtab<-table(rf_pred,tstdata[[7]])
```

```
rf_cmx<-confusionMatrix(rf_mtab)
rf_cmx$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.633205e-01   9.240375e-01   9.433114e-01   9.777747e-01   6.795367e-01
## AccuracyPValue  McnemarPValue
##   1.829776e-59            NaN
```

```
rf_cmx$byClass
```

```
##              Sensitivity Specificity Pos Pred Value Neg Pred Value
## Class: acc     0.9834711   0.9647355      0.8947368      0.9948052
## Class: good    0.7391304   1.0000000      1.0000000      0.9880240
## Class: unacc   0.9772727   0.9879518      0.9942197      0.9534884
## Class: vgood   0.8636364   0.9939516      0.8636364      0.9939516
##              Precision    Recall        F1 Prevalence Detection Rate
## Class: acc   0.8947368 0.9834711 0.9370079 0.23359073     0.22972973
## Class: good  1.0000000 0.7391304 0.8500000 0.04440154     0.03281853
## Class: unacc 0.9942197 0.9772727 0.9856734 0.67953668     0.66409266
## Class: vgood 0.8636364 0.8636364 0.8636364 0.04247104     0.03667954
##              Detection Prevalence Balanced Accuracy
## Class: acc              0.25675676         0.9741033
## Class: good             0.03281853         0.8695652
## Class: unacc            0.66795367         0.9826123
## Class: vgood            0.04247104         0.9287940
```

We see the accuracy is 96% which is slightly lower than earlier models. Sensitivity and Specifity is lower but we dont really care on the True Positve and True Negative values due to multiple classes. We will suggest to find the Out of Bag Error Rate with Cross Valuidation to make sure the predicted results are not over estimated. We can also suggest to define the optimal mtry parameter, number of candidates to feed the algrorithm as by default this is the square of the number of columns(49). We can also ask the model to asess the importance of the each independent variable by adding importance=True parameter.

We can also suggest Ruffio to apply optimization(tunning) of the model by Random Search and Grid Search methods.

XGBoost (Extreme Gradient Boost) 1

He made the same assumptions as the Gradient Boost Models, but wanted to increase speed of the prediction and density of the input. There is no sparsity in the data which tree boster can be applied if it existed. The idea is similarly to reduce variance with resampled data that increases the generalization. My assumption is that with XGBoost he basically wanted to optimize the GBM Models he created.

```
trdatamx<-sparse.model.matrix(class~.-1,data=trdata)
tstdatamx<-sparse.model.matrix(class~.-1,data=tstdata)

xgb_model<-xgboost(data=trdatamx,label=trdata$class,max_depth = 2,
eta = 1, nrounds = 2,nthread = 2, objective = "multi:softmax",num_class=5)
```

```
## [1]  train-merror:0.249587
## [2]  train-merror:0.221488
```

```
xgb_pred <- predict(xgb_model,tstdatamx)
xgb_tab<-table( xgb_pred)
xgb_mtab<-table(xgb_pred,tstdata[[7]])
#xgb_cmx<-confusionMatrix(xgb_mtab)
#xgb_cmx$overall
#xgb_cmx$byClass
```

```
summary(xgb_model)
```

```
##                Length Class              Mode
## handle            1   xgb.Booster.handle externalptr
## raw            3847   -none-             raw
## niter             1   -none-             numeric
## evaluation_log    2   data.table         list
## call             18   -none-             call
## params            6   -none-             list
## callbacks         2   -none-             list
## feature_names    16   -none-             character
## nfeatures         1   -none-             numeric
```

```
xgb_mtab
```

```
##
## xgb_pred acc good unacc vgood
##        1 121   23    71    22
##        3   0    0   281     0
```

XGBoost (Extreme Gradient Boost) 4

```
xgb_model4<-xgboost(data=trdatamx,label=trdata$class,max_depth = 4,
eta = 1, nrounds = 3,nthread = 2, objective = "multi:softmax",num_class=5)
```

```
## [1]  train-merror:0.166116
## [2]  train-merror:0.126446
## [3]  train-merror:0.087603
```

```
 xgb_pred4 <- predict(xgb_model4,tstdatamx)
xgb_tab4<-table( xgb_pred4)
temp_xgb_tab4<-xgb_tab4

xgb_pred4[which(xgb_pred4=="1")]<-levels(y)[1]
xgb_pred4[which(xgb_pred4=="2")]<-levels(y)[2]
xgb_pred4[which(xgb_pred4=="3")]<-levels(y)[3]
xgb_pred4[which(xgb_pred4=="4")]<-levels(y)[4]
xgb_mtab4<-table(xgb_pred4,tstdata[[7]])
xgb_cmx4<-confusionMatrix(xgb_mtab4)
xgb_cmx4$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.648649e-01   7.167519e-01   8.323720e-01   8.931184e-01   6.795367e-01
## AccuracyPValue  McnemarPValue
##   1.626424e-22            NaN
```

```
xgb_cmx4$byClass
```

```
##              Sensitivity Specificity Pos Pred Value Neg Pred Value
## Class: acc     0.8264463   0.8942065      0.7042254      0.9441489
## Class: good    0.4782609   0.9919192      0.7333333      0.9761431
## Class: unacc   0.9289773   0.8734940      0.9396552      0.8529412
## Class: vgood   0.4545455   0.9939516      0.7692308      0.9762376
##              Precision    Recall        F1 Prevalence Detection Rate
## Class: acc   0.7042254 0.8264463 0.7604563 0.23359073     0.19305019
## Class: good  0.7333333 0.4782609 0.5789474 0.04440154     0.02123552
## Class: unacc 0.9396552 0.9289773 0.9342857 0.67953668     0.63127413
## Class: vgood 0.7692308 0.4545455 0.5714286 0.04247104     0.01930502
##              Detection Prevalence Balanced Accuracy
## Class: acc             0.27413127         0.8603264
## Class: good            0.02895753         0.7350900
## Class: unacc           0.67181467         0.9012356
## Class: vgood           0.02509653         0.7242485
```

XGBoost (Extreme Gradient Boost) 5

```
xgb_model5<-xgboost(data=trdatamx,label=trdata$class,max_depth = 5,
eta = 1, nrounds = 4,nthread = 2, objective = "multi:softmax",num_class=5)
```

```
## [1]  train-merror:0.140496
## [2]  train-merror:0.085124
## [3]  train-merror:0.056198
## [4]  train-merror:0.036364
```

```
 xgb_pred5 <- predict(xgb_model5,tstdatamx)
table( xgb_pred5)
```

```
## xgb_pred5
##   1   2   3   4
## 123  22 353  20
```

```
xgb_tab5<-table( xgb_pred5)
temp_xgb_tab5<-xgb_tab5

xgb_pred5[which(xgb_pred5=="1")]<-levels(y)[1]
xgb_pred5[which(xgb_pred5=="2")]<-levels(y)[2]
xgb_pred5[which(xgb_pred5=="3")]<-levels(y)[3]
xgb_pred5[which(xgb_pred5=="4")]<-levels(y)[4]
xgb_mtab5<-table(xgb_pred5,tstdata[[7]])
xgb_cmx5<-confusionMatrix(xgb_mtab5)
xgb_cmx5$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.362934e-01   8.667020e-01   9.116926e-01   9.557452e-01   6.795367e-01
## AccuracyPValue  McnemarPValue
##   3.656179e-46            NaN
```

```
xgb_cmx5$byClass
```

```
##              Sensitivity Specificity Pos Pred Value Neg Pred Value
## Class: acc     0.9008264   0.9647355      0.8861789      0.9696203
## Class: good    0.7391304   0.9898990      0.7727273      0.9879032
## Class: unacc   0.9744318   0.9397590      0.9716714      0.9454545
## Class: vgood   0.7272727   0.9919355      0.8000000      0.9879518
##              Precision    Recall        F1 Prevalence Detection Rate
## Class: acc   0.8861789 0.9008264 0.8934426 0.23359073     0.21042471
## Class: good  0.7727273 0.7391304 0.7555556 0.04440154     0.03281853
## Class: unacc 0.9716714 0.9744318 0.9730496 0.67953668     0.66216216
## Class: vgood 0.8000000 0.7272727 0.7619048 0.04247104     0.03088803
##              Detection Prevalence Balanced Accuracy
## Class: acc              0.23745174         0.9327810
## Class: good             0.04247104         0.8645147
## Class: unacc            0.68146718         0.9570954
## Class: vgood            0.03861004         0.8596041
```

Based on the 3 XGBoost Models Ruffio created, we can suggest him to look at the importance. In the first model, the confusion matrix not populating to display the accuracy and other coeffiecients due to missing test label aurgument. he used objective aurgument ("multi:softmax) to define the objective function for multinomial tree. in his second model he increased the number of iterations to 4 for optimization. To avoid overfitting we can recommend him to add early.stop.round hence maximize parameters.

Conclusion

In Conclusion, we can recommend Ruffio to drop or fix the first Multinomial Model and optimize, GBM, Random Forest and XGBoost Models to further re-evaluate to pick the model to sovle the classification problem (Predicting class evalution of a car).