R Notebook

The following is your first chunk to start with. Remember, you can add chunks using the menu above (Insert -> R) or using the keyboard shortcut Ctrl+Alt+I. A good practice is to use different code chunks to answer different questions. You can delete this comment if you like.

Other useful keyboard shortcuts include Alt- for the assignment operator, and Ctrl+Shift+M for the pipe operator. You can delete these reminders if you don't want them in your report.

```
setwd("C:/Users/munis/Desktop") #Don't forget to set your working directory b
efore you start!

library("tidyverse")

## -- Attaching packages -----------------------------------------------------
--------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts --------------------------------------------------------------
--------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library("tidymodels")

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## -- Attaching packages -----------------------------------------------------
------------------------------------- tidymodels 0.0.3 --

## v broom     0.5.3     v recipes   0.1.9
## v dials     0.0.4     v rsample   0.0.5
## v infer     0.5.1     v yardstick 0.0.4
## v parsnip   0.0.5

## -- Conflicts --------------------------------------------------------------
-------------------------------- tidymodels_conflicts() --
## x scales::discard()   masks purrr::discard()
## x dplyr::filter()     masks stats::filter()
## x recipes::fixed()    masks stringr::fixed()
## x dplyr::lag()        masks stats::lag()
```

```
## x dials::margin()     masks ggplot2::margin()
## x yardstick::spec()   masks readr::spec()
## x recipes::step()     masks stats::step()
## x recipes::yj_trans() masks scales::yj_trans()
```

```r
library("plotly")
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```r
library("skimr")

dfc <-
  read_csv("C:/Users/munis/Desktop/assignment3Carvana.csv")
```

```
## Parsed with column specification:
## cols(
##   Auction = col_character(),
##   Age = col_double(),
##   Make = col_character(),
##   Color = col_character(),
##   WheelType = col_character(),
##   Odo = col_double(),
##   Size = col_character(),
##   MMRAauction = col_double(),
##   MMRAretail = col_double(),
##   BadBuy = col_double()
## )
```

```r
dfc
```

```
## # A tibble: 10,061 x 10
##    Auction    Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <chr>    <dbl> <chr> <chr> <chr>     <dbl> <chr>       <dbl>      <dbl>
<dbl>
##  1 MANHEIM      4 CHRY~ GOLD  Covers    45224 CROS~           0          0
1
##  2 ADESA        6 PONT~ SILV~ Covers    79905 LARGE        3912       4725
1
```

```
## 3 MANHEIM 6 SATU~ WHITE Covers 81116 MEDI~ 2667 3380
0
## 4 ADESA 3 CHEV~ WHITE NULL 82651 COMP~ 5194 8614
1
## 5 MANHEIM 6 CHRY~ BEIGE Alloy 69050 MEDI~ 3265 6441
1
## 6 OTHER 5 CHEV~ RED Alloy 54718 MEDI~ 6921 7975
1
## 7 MANHEIM 5 FORD SILV~ Alloy 75757 COMP~ 3475 6958
1
## 8 ADESA 4 FORD BLUE Covers 65726 VAN 3889 4700
0
## 9 OTHER 5 CHEV~ GOLD Covers 89365 VAN 6131 9793
1
## 10 ADESA 3 CHEV~ WHITE Covers 71794 VAN 6394 7406
0
## # ... with 10,051 more rows
```

```
head(dfc)
```

```
## # A tibble: 6 x 10
## Auction Age Make Color WheelType Odo Size MMRAauction MMRAretail
BadBuy
## <chr> <dbl> <chr> <chr> <chr> <dbl> <chr> <dbl> <dbl>
<dbl>
## 1 MANHEIM 4 CHRYS~ GOLD Covers 45224 CROS~ 0 0
1
## 2 ADESA 6 PONTI~ SILV~ Covers 79905 LARGE 3912 4725
1
## 3 MANHEIM 6 SATURN WHITE Covers 81116 MEDI~ 2667 3380
0
## 4 ADESA 3 CHEVR~ WHITE NULL 82651 COMP~ 5194 8614
1
## 5 MANHEIM 6 CHRYS~ BEIGE Alloy 69050 MEDI~ 3265 6441
1
## 6 OTHER 5 CHEVR~ RED Alloy 54718 MEDI~ 6921 7975
1
```

```
nrow(dfc)
```

```
## [1] 10061
```

```
skim(dfc)
```

Data summary

Name	dfc
Number of rows	10061
Number of columns	10

Column type frequency:

character	5
numeric	5
________________________	
Group variables	None

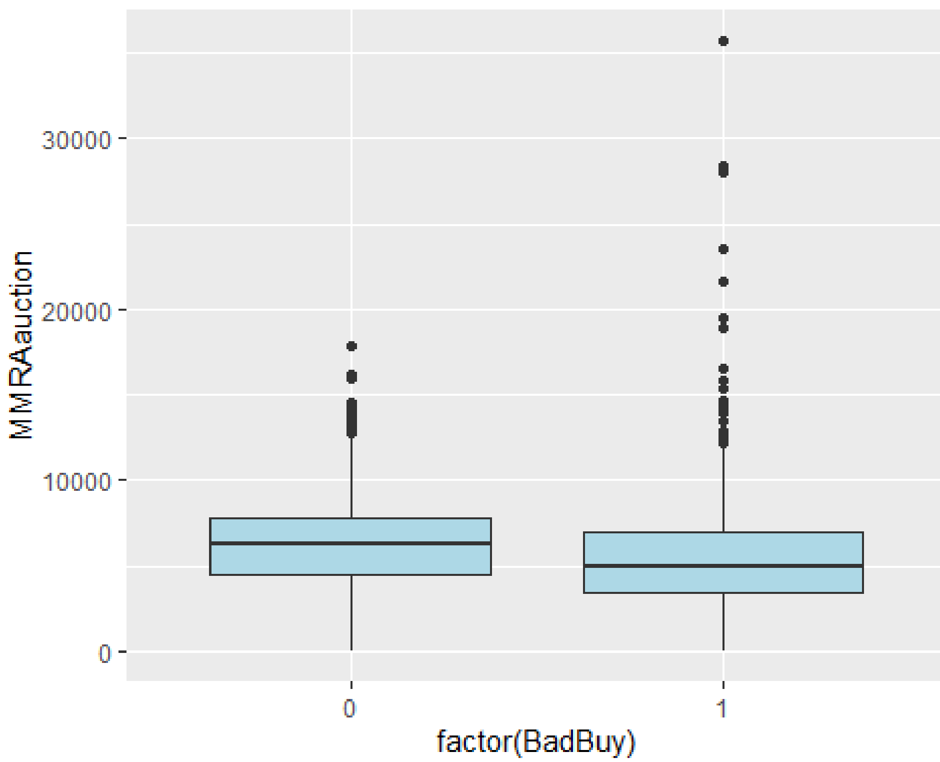
Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Auction	0	1	5	7	0	3	0
Make	0	1	3	10	0	30	0
Color	0	1	3	8	0	17	0
WheelType	0	1	4	7	0	4	0
Size	0	1	3	10	0	12	0

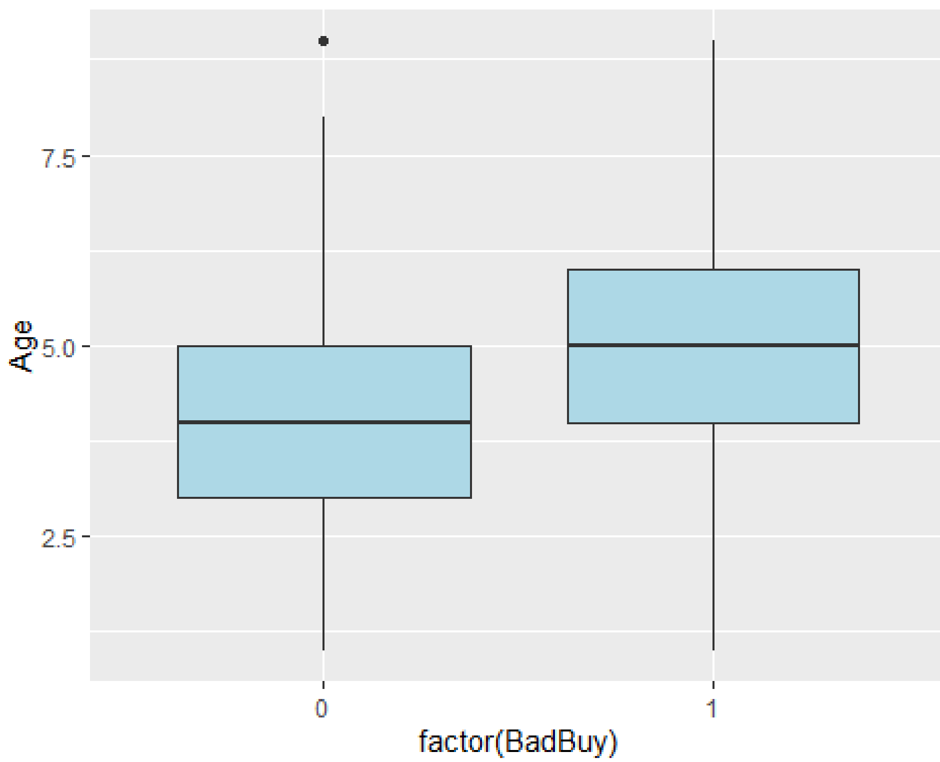
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Age	0	1	4.50	1.77	1	3	4	6	9	
Odo	0	1	72903.87	14498.87	9446	63488	74942	83663	115717	
MMRAauction	0	1	5812.38	2578.85	0	3877	5588	7450	35722	
MMRAretail	0	1	8171.51	3257.19	0	5872	8052	10315	39080	
BadBuy	0	1	0.50	0.50	0	0	0	1	1	

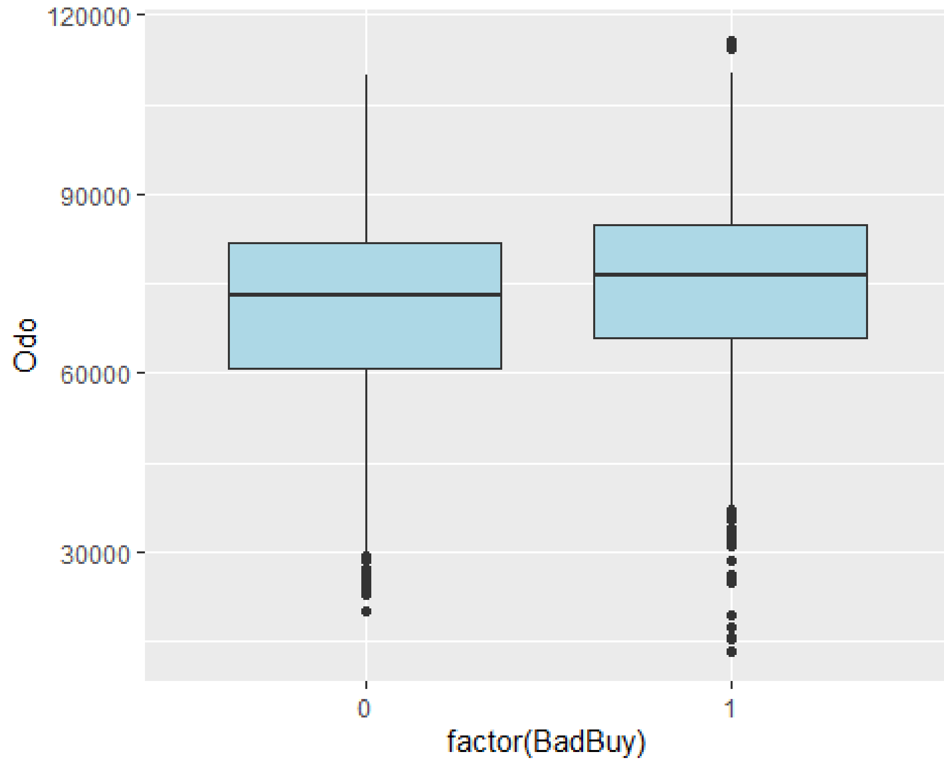
```
set.seed(52156)

#Split the data into two: 80% for the training set, and 20% for the test set
dfcTrain <- dfc %>% sample_frac(0.65)
dfcTest <- dplyr::setdiff(dfc, dfcTrain)

ggplot(data=dfcTrain)+ geom_boxplot(aes(x=factor(BadBuy),y=MMRAauction),fill=
"lightblue")
```




```
ggplotly( ggplot(data=dfcTrain)+ geom_boxplot(aes(x=factor(BadBuy),y=MMRAauct
ion),fill="lightblue"))

 ggplot(data=dfcTrain)+ geom_boxplot(aes(x=factor(BadBuy),y=Age),fill="lightb
lue")
```




```
ggplotly(ggplot(data=dfcTrain)+ geom_boxplot(aes(x=factor(BadBuy),y=Age),fill
="lightblue")
)

 ggplot(data=dfcTrain)+ geom_boxplot(aes(x=factor(BadBuy),y=Odo),fill="lightb
lue")
```




```
ggplotly( ggplot(data=dfcTrain)+ geom_boxplot(aes(x=factor(BadBuy),y=Odo),fil
l="lightblue"))

library(car)

## Loading required package: carData

## Registered S3 methods overwritten by 'car':
##   method                          from
##   influence.merMod                lme4
##   cooks.distance.influence.merMod lme4
##   dfbeta.influence.merMod         lme4
##   dfbetas.influence.merMod        lme4

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some

#library(modelr)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
dfcTrain %>%
  group_by(Size) %>%
  filter(BadBuy==1)%>%
  tally(name='Lemons') %>%
  mutate(pct=100*(Lemons)/sum(Lemons)) %>%
  arrange(desc(Lemons))
```

```
## # A tibble: 12 x 3
##    Size        Lemons   pct
##    <chr>        <int> <dbl>
##  1 MEDIUM        1298 39.8
##  2 COMPACT        448 13.7
##  3 MEDIUMSUV      412 12.6
##  4 LARGE          284  8.70
##  5 VAN            269  8.24
##  6 LARGETRUCK     126  3.86
##  7 SMALLSUV       112  3.43
##  8 LARGESUV        76  2.33
##  9 SPECIALTY       68  2.08
## 10 CROSSOVER       66  2.02
## 11 SPORTS          58  1.78
## 12 SMALLTRUCK      47  1.44
```

```r
dfcTrain %>%
  group_by(Size) %>%
  filter(BadBuy==0)%>%
  tally(name='Good_cars') %>%
  mutate(pct=100*(Good_cars)/sum(Good_cars)) %>%
  arrange(desc(Good_cars))
```

```
## # A tibble: 12 x 3
##    Size        Good_cars   pct
##    <chr>           <int> <dbl>
##  1 MEDIUM           1384 42.2
##  2 LARGE             423 12.9
##  3 MEDIUMSUV         348 10.6
##  4 COMPACT           309  9.43
##  5 VAN               250  7.63
```

```
##   6 LARGETRUCK       156  4.76
##   7 SMALLSUV          97  2.96
##   8 CROSSOVER         88  2.69
##   9 SPECIALTY         79  2.41
##  10 LARGESUV          53  1.62
##  11 SPORTS            46  1.40
##  12 SMALLTRUCK        43  1.31
```

Q3a)b)

```
resultsLPM1<-
  lm(BadBuy~ ., data=dfcTrain)
#detach('package:modelr', unload=TRUE)
resultsLPM1

##
## Call:
## lm(formula = BadBuy ~ ., data = dfcTrain)
##
## Coefficients:
##      (Intercept)     AuctionMANHEIM       AuctionOTHER                Age
##       -1.996e-01          4.065e-02          2.287e-02          5.154e-02
##        MakeBUICK       MakeCADILLAC      MakeCHEVROLET       MakeCHRYSLER
##        2.392e-01          2.664e-01          1.861e-01          2.944e-01
##        MakeDODGE           MakeFORD            MakeGMC          MakeHONDA
##        2.384e-01          2.620e-01          1.398e-01          1.114e-01
##      MakeHYUNDAI       MakeINFINITI          MakeISUZU           MakeJEEP
##        2.099e-01          3.671e-01          1.764e-01          2.537e-01
##          MakeKIA          MakeLEXUS        MakeLINCOLN          MakeMAZDA
##        2.190e-01          8.805e-01          3.712e-01          2.567e-01
##      MakeMERCURY           MakeMINI     MakeMITSUBISHI         MakeNISSAN
##        2.980e-01          3.301e-01          1.179e-01          2.310e-01
##   MakeOLDSMOBILE        MakePONTIAC         MakeSATURN          MakeSCION
##        3.261e-01          2.181e-01          2.800e-01          1.091e-01
##       MakeSUBARU         MakeSUZUKI         MakeTOYOTA     MakeVOLKSWAGEN
##        2.432e-01          3.696e-01          1.638e-01          2.630e-01
##        MakeVOLVO         ColorBLACK          ColorBLUE         ColorBROWN
##       -1.809e-01          2.220e-02          1.890e-02          1.819e-02
##        ColorGOLD         ColorGREEN          ColorGREY        ColorMAROON
##        5.438e-02          2.264e-02          3.804e-02          7.248e-02
##    ColorNOTAVAIL          ColorNULL        ColorORANGE         ColorOTHER
##       -4.753e-02         -1.179e-01          4.598e-02         -1.388e-01
##      ColorPURPLE           ColorRED        ColorSILVER         ColorWHITE
##        1.955e-02          6.169e-02          4.814e-02          6.047e-02
##      ColorYELLOW    WheelTypeCovers      WheelTypeNULL   WheelTypeSpecial
##       -6.072e-02         -3.534e-02          5.096e-01         -8.805e-03
##              Odo      SizeCROSSOVER          SizeLARGE       SizeLARGESUV
##        2.888e-06         -1.783e-01         -1.475e-01         -1.379e-01
##   SizeLARGETRUCK         SizeMEDIUM      SizeMEDIUMSUV       SizeSMALLSUV
##       -1.916e-01         -9.926e-02         -9.874e-02         -1.333e-01
```

```
##   SizeSMALLTRUCK     SizeSPECIALTY        SizeSPORTS           SizeVAN
##       -1.449e-01        -7.220e-02        -1.081e-01        -1.136e-01
##      MMRAauction        MMRAretail
##        1.595e-06        -1.126e-06

#dfcTrain
performance<-metric_set(rmse,mae)
result<-dfcTest%>%
  mutate('predictedVal'= predict(resultsLPM1, dfcTest))
result

## # A tibble: 3,521 x 11
##    Auction    Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <chr>    <dbl> <chr> <chr> <chr>     <dbl> <chr>       <dbl>      <dbl>
<dbl>
##  1 MANHEIM     6 SATU~ WHITE Covers    81116 MEDI~        2667       3380
0
##  2 OTHER       5 CHEV~ RED   Alloy     54718 MEDI~        6921       7975
1
##  3 OTHER       5 CHEV~ GOLD  Covers    89365 VAN          6131       9793
1
##  4 ADESA       3 CHEV~ WHITE Covers    71794 VAN          6394       7406
0
##  5 OTHER       3 CHEV~ WHITE NULL      67229 COMP~        5785       9834
1
##  6 MANHEIM     3 DODGE GOLD  Covers    71079 MEDI~        4297       5141
1
##  7 MANHEIM     6 OLDS~ SILV~ Alloy     71235 MEDI~        3325       4091
1
##  8 MANHEIM     8 PONT~ SILV~ Alloy     90325 MEDI~        2150       4937
1
##  9 MANHEIM     6 PONT~ GREEN Alloy     96893 MEDI~        4059       4884
1
## 10 OTHER       2 DODGE BLUE  Covers    45151 MEDI~        7982       9121
1
## # ... with 3,511 more rows, and 1 more variable: predictedVal <dbl>

Model<-performance(result,truth=BadBuy,estimate=predictedVal)
Model

## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.453
## 2 mae     standard       0.415

result2<-dfcTrain%>%
  mutate('predictedVal'= predict(resultsLPM1, dfcTrain))
Model2<-performance(result2,truth=BadBuy,estimate=predictedVal)
Model2
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.448
## 2 mae     standard       0.410
```

```
colsToFactor<- c('BadBuy')
dfcTest<-dfcTest%>%
  mutate_at(colsToFactor, ~factor(.))


colsToFactor<- c('BadBuy')
dfcTrain<-dfcTrain%>%
  mutate_at(colsToFactor, ~factor(.))

str(dfcTrain)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 6540 obs. of  10
variables:
##  $ Auction    : chr  "MANHEIM" "MANHEIM" "MANHEIM" "ADESA" ...
##  $ Age        : num  4 5 2 4 5 4 2 7 6 8 ...
##  $ Make       : chr  "FORD" "MINI" "CHEVROLET" "NISSAN" ...
##  $ Color      : chr  "SILVER" "BLUE" "SILVER" "BLUE" ...
##  $ WheelType  : chr  "NULL" "Alloy" "Covers" "NULL" ...
##  $ Odo        : num  77591 80013 75493 84827 57388 ...
##  $ Size       : chr  "LARGETRUCK" "COMPACT" "LARGE" "MEDIUM" ...
##  $ MMRAauction: num  9774 11040 9707 6073 5574 ...
##  $ MMRAretail : num  14506 12423 13975 9791 8984 ...
##  $ BadBuy     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 2 1 1 ...
```

Q3)c)

```
#colsToFactor<- c('BadBuy')
#resultsLPM<-resultsLPM%>%
#  mutate_at(colsToFactor, ~factor(.))

#resultsLPM1<-
 # lm(BadBuy~ ., data=dfcTrain)
#result<-
resultsLPM<-resultsLPM1%>%

predict(dfcTest, type='response') %>%     #=> Use the option type='response'
for probabilities
      bind_cols(dfcTest, predictedProb=.)%>%
  mutate(predictedClass = as.factor(ifelse(predictedProb>0.5, 1, 0)))

resultsLPM%>%
  group_by(predictedClass) %>%
  tally  %>%
```

```
  #group_by(school_number) %>%
  mutate(pct=(100*n)/sum(n))#%>%

## # A tibble: 2 x 3
##   predictedClass     n   pct
##   <fct>          <int> <dbl>
## 1 0               2117  60.1
## 2 1               1404  39.9

#resultsLPM

resultsLPM%>%
    conf_mat(truth=BadBuy, estimate=predictedClass)

##           Truth
## Prediction    0    1
##          0 1374  743
##          1  408  996

newData <- tibble(Auction="ADESA",      Age=1,      Make="HONDA",   Color="SI
LVER",
WheelType="Covers", Odo=10000,  Size="LARGE",
MMRAauction=8000,   MMRAretail=10000
)
predict(resultsLPM1, newData)

##          1
## -0.1410712

dfc

## # A tibble: 10,061 x 10
##    Auction    Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <chr>    <dbl> <chr> <chr> <chr>     <dbl> <chr>       <dbl>      <dbl>
<dbl>
##  1 MANHEIM     4 CHRY~ GOLD  Covers    45224 CROS~           0          0
1
##  2 ADESA       6 PONT~ SILV~ Covers    79905 LARGE        3912       4725
1
##  3 MANHEIM     6 SATU~ WHITE Covers    81116 MEDI~        2667       3380
0
##  4 ADESA       3 CHEV~ WHITE NULL      82651 COMP~        5194       8614
1
##  5 MANHEIM     6 CHRY~ BEIGE Alloy     69050 MEDI~        3265       6441
1
##  6 OTHER       5 CHEV~ RED   Alloy     54718 MEDI~        6921       7975
1
##  7 MANHEIM     5 FORD  SILV~ Alloy     75757 COMP~        3475       6958
1
##  8 ADESA       4 FORD  BLUE  Covers    65726 VAN          3889       4700
```

```
0
##  9 OTHER       5 CHEV~ GOLD  Covers    89365 VAN          6131       9793
1
## 10 ADESA       3 CHEV~ WHITE Covers    71794 VAN          6394       7406
0
## # ... with 10,051 more rows
```

```
dfc %>%
  group_by((Color)) %>%
  tally  %>%
  #group_by(school_number) %>%
  mutate(pct=(100*n)/sum(n))
```

```
## # A tibble: 17 x 3
##    `(Color)`     n     pct
##    <chr>     <int>   <dbl>
##  1 BEIGE       237  2.36
##  2 BLACK      1013 10.1
##  3 BLUE       1386 13.8
##  4 BROWN        65  0.646
##  5 GOLD        766  7.61
##  6 GREEN       442  4.39
##  7 GREY       1054 10.5
##  8 MAROON      281  2.79
##  9 NOTAVAIL     26  0.258
## 10 NULL          2  0.0199
## 11 ORANGE       43  0.427
## 12 OTHER        37  0.368
## 13 PURPLE       57  0.567
## 14 RED         881  8.76
## 15 SILVER     2081 20.7
## 16 WHITE      1653 16.4
## 17 YELLOW       37  0.368
```

```
dfc%>%
  filter(Color=="NULL"|Color=="NOTAVAIL")
```

```
## # A tibble: 28 x 10
##    Auction   Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <chr>   <dbl> <chr> <chr> <chr>     <dbl> <chr>       <dbl>      <dbl>
<dbl>
##  1 OTHER       2 DODGE NOTA~ NULL      50104 MEDI~        7533      10574
1
##  2 MANHEIM     6 FORD  NOTA~ Alloy     86745 MEDI~        3838       4645
1
##  3 OTHER       9 FORD  NOTA~ NULL      93287 SPOR~        2827       5871
1
##  4 OTHER       1 CHRY~ NOTA~ Covers    50633 MEDI~        8252       9412
0
##  5 OTHER       5 JEEP  NOTA~ NULL      77394 SMAL~        5485       6424
```

```
1
##   6 OTHER        5 FORD  NOTA~ Alloy     71145 MEDI~        6078        7064
1
##   7 ADESA        7 MERC~ NULL  NULL      85546 MEDI~        3779        4581
1
##   8 OTHER        5 JEEP  NOTA~ Alloy     85385 MEDI~        6682        7717
1
##   9 MANHEIM      8 PONT~ NOTA~ Alloy     73430 SPOR~        4650        7935
1
## 10 ADESA        3 PONT~ NOTA~ NULL      50486 MEDI~        9763       11044
1
## # ... with 18 more rows
```

Q4)a)

```
dfc <- dfc %>% mutate(Color = if_else(Color == "NULL", "NOTAVAIL", Color))

dfc %>%
  group_by((Make)) %>%
  tally()

## # A tibble: 30 x 2
##    `(Make)`       n
##    <chr>      <int>
##  1 ACURA          9
##  2 BUICK        103
##  3 CADILLAC       3
##  4 CHEVROLET   2121
##  5 CHRYSLER    1217
##  6 DODGE       1653
##  7 FORD        1764
##  8 GMC           85
##  9 HONDA         77
## 10 HYUNDAI      239
## # ... with 20 more rows

new<-c("ACURA","VOLVO","CADILLAC","MINI","SUBARU","LEXUS")

dfc<-dfc%>%
  mutate(Make=if_else(Make %in% unlist(new),"OTHER",Make ))

dfc<-dfc%>%
  mutate(BadBuy=as.factor(BadBuy))

set.seed(52156)


dfcTrain<-dfc%>%sample_frac((0.65))

dfcTest<-dplyr::setdiff(dfc,dfcTrain)
```

Q4)d)

```
library(e1071)
library(caret)
resultsCaret<-
  train(BadBuy ~ ., family='binomial', data=dfcTrain, method='glm') %>%
  predict(dfcTest, type='raw')%>%
  bind_cols(dfcTest, predictedClass=.)


resultsCaret%>%
  xtabs(~predictedClass+BadBuy, .)%>%
  confusionMatrix(positive='1')
```

```
## Confusion Matrix and Statistics
##
##               BadBuy
## predictedClass    0    1
##              0 1341  721
##              1  441 1018
##
##                Accuracy : 0.67
##                  95% CI : (0.6542, 0.6855)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3386
##
##  Mcnemar's Test P-Value : 2.731e-16
##
##             Sensitivity : 0.5854
##             Specificity : 0.7525
##          Pos Pred Value : 0.6977
##          Neg Pred Value : 0.6503
##              Prevalence : 0.4939
##          Detection Rate : 0.2891
##    Detection Prevalence : 0.4144
##       Balanced Accuracy : 0.6690
##
##        'Positive' Class : 1
##
```

Q4)b)c)

```
summary(dfc)
```

```
##    Auction                Age           Make               Color
##  Length:10061       Min.   :1.000   Length:10061       Length:10061
##  Class :character   1st Qu.:3.000   Class :character   Class :character
##  Mode  :character   Median :4.000   Mode  :character   Mode  :character
##                     Mean   :4.505
```

```
##                     3rd Qu.:6.000
##                     Max.   :9.000
##   WheelType              Odo              Size            MMRAauction
##  Length:10061       Min.   :  9446   Length:10061       Min.   :    0
##  Class :character   1st Qu.: 63488   Class :character   1st Qu.: 3877
##  Mode  :character   Median : 74942   Mode  :character   Median : 5588
##                     Mean   : 72904                      Mean   : 5812
##                     3rd Qu.: 83663                      3rd Qu.: 7450
##                     Max.   :115717                      Max.   :35722
##    MMRAretail    BadBuy
##  Min.   :    0   0:5058
##  1st Qu.: 5872   1:5003
##  Median : 8052
##  Mean   : 8172
##  3rd Qu.:10315
##  Max.   :39080
```

```
model <- glm(BadBuy ~ ., family='binomial', data=dfcTrain)
summary(model)
```

```
##
## Call:
## glm(formula = BadBuy ~ ., family = "binomial", data = dfcTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0725  -0.9782  -0.4717   1.0946   2.1705
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.472e+00  4.513e-01  -5.478 4.30e-08 ***
## AuctionMANHEIM    1.735e-01  7.493e-02   2.316 0.020579 *
## AuctionOTHER      9.519e-02  9.037e-02   1.053 0.292217
## Age               2.785e-01  2.887e-02   9.647  < 2e-16 ***
## MakeCHEVROLET    -2.774e-01  2.895e-01  -0.958 0.337982
## MakeCHRYSLER      2.527e-01  3.011e-01   0.839 0.401419
## MakeDODGE        -2.483e-02  2.966e-01  -0.084 0.933287
## MakeFORD          1.020e-01  2.945e-01   0.346 0.729155
## MakeGMC          -5.054e-01  4.193e-01  -1.205 0.228054
## MakeHONDA        -6.530e-01  4.317e-01  -1.512 0.130433
## MakeHYUNDAI      -1.623e-01  3.381e-01  -0.480 0.631275
## MakeINFINITI      3.727e-01  1.280e+00   0.291 0.771007
## MakeISUZU        -3.227e-01  7.887e-01  -0.409 0.682408
## MakeJEEP          3.121e-02  3.496e-01   0.089 0.928850
## MakeKIA          -9.342e-02  3.281e-01  -0.285 0.775823
## MakeLINCOLN       6.866e-01  7.410e-01   0.927 0.354146
## MakeMAZDA         3.015e-02  3.530e-01   0.085 0.931925
## MakeMERCURY       2.670e-01  3.632e-01   0.735 0.462313
## MakeMITSUBISHI   -6.722e-01  3.692e-01  -1.821 0.068664 .
## MakeNISSAN       -7.824e-02  3.213e-01  -0.243 0.807645
```

```
## MakeOLDSMOBILE    4.725e-01  5.397e-01   0.875 0.381344    
## MakeOTHER         3.109e-01  6.256e-01   0.497 0.619240    
## MakePONTIAC      -1.156e-01  3.039e-01  -0.380 0.703748    
## MakeSATURN        2.040e-01  3.293e-01   0.620 0.535513    
## MakeSCION        -6.429e-01  7.485e-01  -0.859 0.390426    
## MakeSUZUKI        6.756e-01  3.578e-01   1.888 0.058974 .  
## MakeTOYOTA       -4.609e-01  3.718e-01  -1.240 0.215081    
## MakeVOLKSWAGEN    3.278e-02  6.815e-01   0.048 0.961638    
## ColorBLACK        1.502e-01  2.157e-01   0.696 0.486312    
## ColorBLUE         1.197e-01  2.103e-01   0.569 0.569124    
## ColorBROWN        1.348e-01  3.891e-01   0.346 0.729074    
## ColorGOLD         3.066e-01  2.201e-01   1.393 0.163652    
## ColorGREEN        1.723e-01  2.369e-01   0.727 0.466976    
## ColorGREY         2.307e-01  2.139e-01   1.078 0.280903    
## ColorMAROON       4.114e-01  2.596e-01   1.585 0.112963    
## ColorNOTAVAIL    -2.898e-01  7.521e-01  -0.385 0.700011    
## ColorORANGE       2.922e-01  4.655e-01   0.628 0.530251    
## ColorOTHER       -1.168e+00  6.442e-01  -1.812 0.069933 .  
## ColorPURPLE       1.899e-01  4.250e-01   0.447 0.655029    
## ColorRED          3.374e-01  2.177e-01   1.550 0.121257    
## ColorSILVER       2.850e-01  2.057e-01   1.386 0.165860    
## ColorWHITE        3.409e-01  2.083e-01   1.636 0.101745    
## ColorYELLOW      -2.904e-01  4.947e-01  -0.587 0.557141    
## WheelTypeCovers  -1.082e-01  6.698e-02  -1.615 0.106304    
## WheelTypeNULL     3.489e+00  1.727e-01  20.202  < 2e-16 ***
## WheelTypeSpecial -5.363e-02  2.663e-01  -0.201 0.840390    
## Odo               1.484e-05  2.184e-06   6.796 1.08e-11 ***
## SizeCROSSOVER    -9.331e-01  2.220e-01  -4.203 2.63e-05 ***
## SizeLARGE        -7.613e-01  1.319e-01  -5.770 7.91e-09 ***
## SizeLARGESUV     -7.972e-01  2.454e-01  -3.249 0.001157 ** 
## SizeLARGETRUCK   -1.013e+00  1.827e-01  -5.547 2.90e-08 ***
## SizeMEDIUM       -5.260e-01  1.015e-01  -5.181 2.21e-07 ***
## SizeMEDIUMSUV    -5.453e-01  1.425e-01  -3.826 0.000130 ***
## SizeSMALLSUV     -6.989e-01  2.079e-01  -3.361 0.000776 ***
## SizeSMALLTRUCK   -7.329e-01  2.520e-01  -2.908 0.003632 ** 
## SizeSPECIALTY    -4.271e-01  2.274e-01  -1.878 0.060352 .  
## SizeSPORTS       -5.701e-01  2.545e-01  -2.240 0.025066 *  
## SizeVAN          -5.982e-01  1.362e-01  -4.394 1.11e-05 ***
## MMRAauction       2.895e-05  3.634e-05   0.797 0.425670    
## MMRAretail       -8.784e-06  2.241e-05  -0.392 0.695044    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 9066.3  on 6539  degrees of freedom
## Residual deviance: 7528.1  on 6480  degrees of freedom
## AIC: 7648.1
## 
## Number of Fisher Scoring iterations: 5
```

```r
exp(coef(model))
```

```
##       (Intercept)    AuctionMANHEIM      AuctionOTHER               Age
##        0.08437858        1.18946691        1.09986387        1.32116202
##     MakeCHEVROLET      MakeCHRYSLER         MakeDODGE          MakeFORD
##        0.75778178        1.28744383        0.97547613        1.10735004
##           MakeGMC         MakeHONDA       MakeHYUNDAI      MakeINFINITI
##        0.60323457        0.52049737        0.85021505        1.45161848
##         MakeISUZU          MakeJEEP           MakeKIA       MakeLINCOLN
##        0.72416820        1.03170472        0.91080706        1.98696169
##         MakeMAZDA       MakeMERCURY    MakeMITSUBISHI        MakeNISSAN
##        1.03061116        1.30598879        0.51060714        0.92474703
##    MakeOLDSMOBILE         MakeOTHER       MakePONTIAC        MakeSATURN
##        1.60397687        1.36460391        0.89085880        1.22630987
##         MakeSCION        MakeSUZUKI        MakeTOYOTA    MakeVOLKSWAGEN
##        0.52579005        1.96521483        0.63068658        1.03332045
##        ColorBLACK         ColorBLUE        ColorBROWN         ColorGOLD
##        1.16204597        1.12720518        1.14428128        1.35881972
##        ColorGREEN         ColorGREY       ColorMAROON     ColorNOTAVAIL
##        1.18807424        1.25947392        1.50895187        0.74842483
##       ColorORANGE        ColorOTHER       ColorPURPLE          ColorRED
##        1.33931201        0.31112534        1.20909575        1.40123691
##       ColorSILVER        ColorWHITE       ColorYELLOW  WheelTypeCovers
##        1.32980630        1.40616624        0.74793801        0.89747194
##     WheelTypeNULL WheelTypeSpecial               Odo     SizeCROSSOVER
##       32.73825555        0.94777937        1.00001484        0.39333606
##         SizeLARGE      SizeLARGESUV    SizeLARGETRUCK        SizeMEDIUM
##        0.46705813        0.45057313        0.36300261        0.59094577
##     SizeMEDIUMSUV      SizeSMALLSUV    SizeSMALLTRUCK     SizeSPECIALTY
##        0.57963900        0.49710890        0.48052501        0.65237868
##        SizeSPORTS           SizeVAN        MMRAauction        MMRAretail
##        0.56544253        0.54978912        1.00002895        0.99999122
```

Q4)d)

```r
result <-
  glm(BadBuy ~ ., family='binomial', data=dfcTrain) %>%
  predict(dfcTest, type='response')%>%
  bind_cols(dfcTest, predictedProb=.)%>%
  mutate(predictedClass = as.factor(ifelse(predictedProb>0.5, 1, 0)))

result %>%
  group_by(predictedClass) %>%
  tally  %>%
  #group_by(school_number) %>%
  mutate(pct=(100*n)/sum(n))
```

```
## # A tibble: 2 x 3
##   predictedClass     n   pct
##   <fct>          <int> <dbl>
```

```
## 1 0               2062  58.6
## 2 1               1459  41.4
```

```
result%>%
  conf_mat(truth=BadBuy, estimate=predictedClass)
```

```
##           Truth
## Prediction    0    1
##          0 1341  721
##          1  441 1018
```

```
result
```

```
## # A tibble: 3,521 x 12
##    Auction    Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <chr>    <dbl> <chr> <chr> <chr>     <dbl> <chr>       <dbl>      <dbl>
<fct>
##  1 MANHEIM      6 SATU~ WHITE Covers    81116 MEDI~        2667       3380
0
##  2 OTHER        5 CHEV~ RED   Alloy     54718 MEDI~        6921       7975
1
##  3 OTHER        5 CHEV~ GOLD  Covers    89365 VAN          6131       9793
1
##  4 ADESA        3 CHEV~ WHITE Covers    71794 VAN          6394       7406
0
##  5 OTHER        3 CHEV~ WHITE NULL      67229 COMP~        5785       9834
1
##  6 MANHEIM      3 DODGE GOLD  Covers    71079 MEDI~        4297       5141
1
##  7 MANHEIM      6 OLDS~ SILV~ Alloy     71235 MEDI~        3325       4091
1
##  8 MANHEIM      8 PONT~ SILV~ Alloy     90325 MEDI~        2150       4937
1
##  9 MANHEIM      6 PONT~ GREEN Alloy     96893 MEDI~        4059       4884
1
## 10 OTHER        2 DODGE BLUE  Covers    45151 MEDI~        7982       9121
1
## # ... with 3,511 more rows, and 2 more variables: predictedProb <dbl>,
## #   predictedClass <fct>
```

Q4)e)

```
newData2 <- tibble(Auction="ADESA",     Age=1,      Make="HONDA",   Color="SI
LVER",
WheelType="Covers", Odo=10000,  Size="LARGE",
MMRAauction=8000,   MMRAretail=10000
)
predict(model, newData2)
```

```
##         1
## -3.139145
```

Q5)a)

```r
set.seed(123)

resultsCaret<-
  train(BadBuy ~ ., family='binomial', data=dfcTrain, method='lda',
              trControl = trainControl(method = "cv")) %>%
  predict(dfcTest, type='raw')%>%
  bind_cols(dfcTest, predictedClass=.)


resultsCaret%>%
  xtabs(~predictedClass+BadBuy, .)%>%
  confusionMatrix(positive='1')
```

```
## Confusion Matrix and Statistics
##
##                BadBuy
## predictedClass    0    1
##               0 1377  749
##               1  405  990
##
##                Accuracy : 0.6723
##                  95% CI : (0.6565, 0.6878)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3428
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5693
##             Specificity : 0.7727
##          Pos Pred Value : 0.7097
##          Neg Pred Value : 0.6477
##              Prevalence : 0.4939
##          Detection Rate : 0.2812
##    Detection Prevalence : 0.3962
##       Balanced Accuracy : 0.6710
##
##        'Positive' Class : 1
##
```

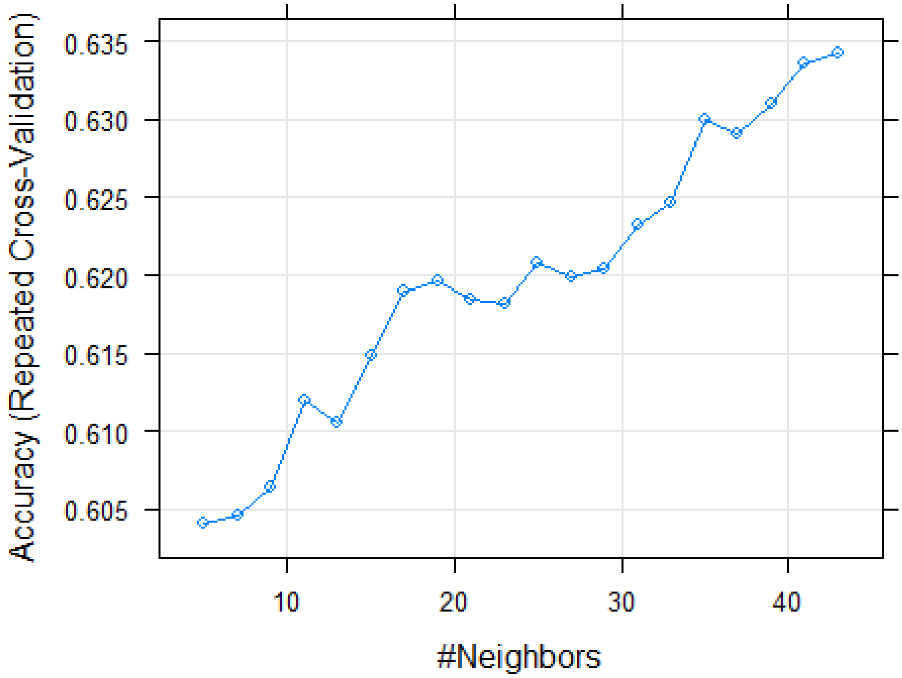
Q5)b)

```r
set.seed(123)

ctrl <- trainControl(method="repeatedcv", number = 10,repeats = 3)
```

```r
knnFit <- train(BadBuy ~ ., data = dfcTrain, method = "knn", trControl = ctrl
, preProcess = c("center","scale"), tuneLength = 20)

knnFit1<-knnFit%>%predict(dfcTest, type='raw')%>%
  bind_cols(dfcTest, predictedClass=.)


knnFit1%>%
  xtabs(~predictedClass+BadBuy, .)%>%
  confusionMatrix(positive='1')

## Confusion Matrix and Statistics
##
##               BadBuy
## predictedClass    0    1
##              0 1322  815
##              1  460  924
##
##                Accuracy : 0.6379
##                  95% CI : (0.6218, 0.6538)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2739
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5313
##             Specificity : 0.7419
##          Pos Pred Value : 0.6676
##          Neg Pred Value : 0.6186
##              Prevalence : 0.4939
##          Detection Rate : 0.2624
##    Detection Prevalence : 0.3931
##       Balanced Accuracy : 0.6366
##
##        'Positive' Class : 1
##

#print(metrics.accuracy_score(y_test, y_pred))

knnFit

## k-Nearest Neighbors
##
## 6540 samples
##    9 predictor
##    2 classes: '0', '1'
##
```

```
## Pre-processing: centered (59), scaled (59)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 5885, 5886, 5887, 5887, 5885, 5886, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.6040325  0.2079463
##    7  0.6045911  0.2090288
##    9  0.6063767  0.2125878
##   11  0.6119275  0.2236915
##   13  0.6106052  0.2210440
##   15  0.6148821  0.2295868
##   17  0.6189115  0.2376305
##   19  0.6196234  0.2390484
##   21  0.6184012  0.2365925
##   23  0.6181482  0.2360795
##   25  0.6207481  0.2412715
##   27  0.6199291  0.2396292
##   29  0.6204356  0.2406329
##   31  0.6232918  0.2463382
##   33  0.6246674  0.2490864
##   35  0.6300197  0.2597791
##   37  0.6291018  0.2579391
##   39  0.6310379  0.2618037
##   41  0.6335360  0.2668000
##   43  0.6343013  0.2683294
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 43.
```

```r
plot(knnFit)
```

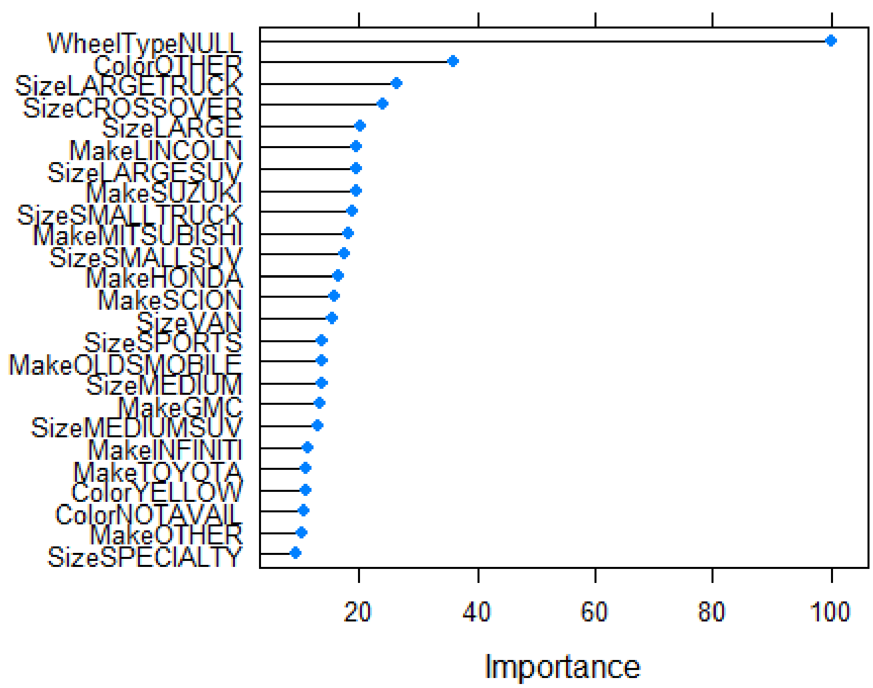


Q5)c)

```
set.seed(123)

#Set the grid for the lambda values
lambdaValues <- 10^seq(-5, 2, length = 100)



fitLasso <- train(BadBuy ~ ., family='binomial', data=dfcTrain, method='glmne
t', trControl=trainControl(method='cv', number=10), tuneGrid = expand.grid(al
pha=1, lambda=lambdaValues))

#Variable importance complete table
varImp(fitLasso)$importance %>%    # Add scale=FALSE inside VarImp if you don
't want to scale
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()

## # A tibble: 59 x 3
##    Variable       Overall Importance
##    <chr>            <dbl> <chr>
##  1 WheelTypeNULL    100   100%
##  2 ColorOTHER        36.2 36%
```

```
## 3 SizeLARGETRUCK 26.5 26%
## 4 SizeCROSSOVER 24.2 24%
## 5 SizeLARGE 20.1 20%
## 6 MakeLINCOLN 19.7 20%
## 7 SizeLARGESUV 19.6 20%
## 8 MakeSUZUKI 19.4 19%
## 9 SizeSMALLTRUCK 18.8 19%
## 10 MakeMITSUBISHI 18.1 18%
## # ... with 49 more rows
```

```
#Variable importance plot with the most important variables
plot(varImp(fitLasso),top=25)  # Add top = XX to change the number of visible
variables
```




```
#Optimum lambda selected by the algorithm
fitLasso$bestTune$lambda   # You can also run fitLasso$finalModel$lambdaOpt

## [1] 0.0003053856

#Not so useful but helps with understanding -See how variables are dropped as
lambda increases
#plot(fitLasso$finalModel, xvar="lambda", label = TRUE)

#Not so useful but helps with understanding -See the coefficients from the fi
nal lasso model
#coef(fitLasso$finalModel, fitLasso$bestTune$lambda)   # You can also use fit
Lasso$finalModel$lambdaOpt for optimum lambda
```

```r
resultsLasso <-
  fitLasso %>%
  predict(dfcTest, type='raw') %>%
  bind_cols(dfcTest, predictedClass=.)

resultsLasso %>%
  xtabs(~predictedClass+BadBuy, .) %>%
  confusionMatrix(positive = '1')

## Confusion Matrix and Statistics
##
##               BadBuy
## predictedClass    0    1
##              0 1339  721
##              1  443 1018
##
##                Accuracy : 0.6694
##                  95% CI : (0.6536, 0.6849)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.3374
##
##  Mcnemar's Test P-Value : 4.7e-16
##
##             Sensitivity : 0.5854
##             Specificity : 0.7514
##          Pos Pred Value : 0.6968
##          Neg Pred Value : 0.6500
##              Prevalence : 0.4939
##          Detection Rate : 0.2891
##    Detection Prevalence : 0.4149
##       Balanced Accuracy : 0.6684
##
##        'Positive' Class : 1
##

fitLasso

## glmnet
##
## 6540 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5885, 5886, 5887, 5887, 5885, 5886, ...
## Resampling results across tuning parameters:
```

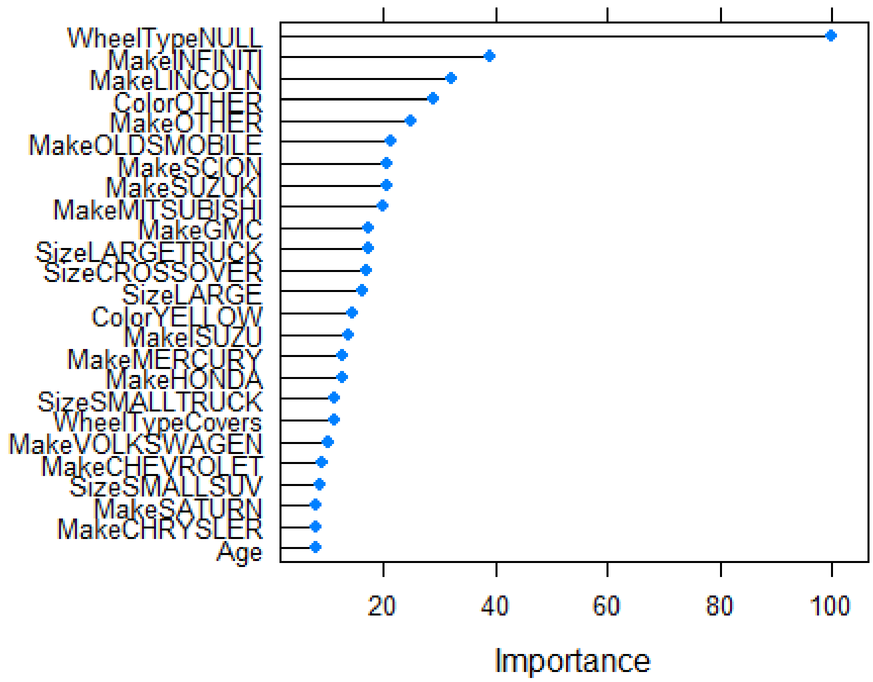
```
##
##   lambda        Accuracy   Kappa
##   1.000000e-05  0.6746304  0.3490374
##   1.176812e-05  0.6746304  0.3490374
##   1.384886e-05  0.6746304  0.3490374
##   1.629751e-05  0.6746304  0.3490374
##   1.917910e-05  0.6746304  0.3490374
##   2.257020e-05  0.6746304  0.3490374
##   2.656088e-05  0.6746304  0.3490374
##   3.125716e-05  0.6746304  0.3490374
##   3.678380e-05  0.6746304  0.3490374
##   4.328761e-05  0.6746304  0.3490374
##   5.094138e-05  0.6746304  0.3490374
##   5.994843e-05  0.6746304  0.3490374
##   7.054802e-05  0.6746304  0.3490374
##   8.302176e-05  0.6746304  0.3490374
##   9.770100e-05  0.6746304  0.3490374
##   1.149757e-04  0.6746304  0.3490374
##   1.353048e-04  0.6746304  0.3490374
##   1.592283e-04  0.6746304  0.3490374
##   1.873817e-04  0.6746304  0.3490374
##   2.205131e-04  0.6746304  0.3490369
##   2.595024e-04  0.6743246  0.3484258
##   3.053856e-04  0.6747826  0.3493417
##   3.593814e-04  0.6744763  0.3487289
##   4.229243e-04  0.6741705  0.3481170
##   4.977024e-04  0.6738659  0.3475095
##   5.857021e-04  0.6721834  0.3441435
##   6.892612e-04  0.6724876  0.3447535
##   8.111308e-04  0.6718764  0.3435332
##   9.545485e-04  0.6717231  0.3432275
##   1.123324e-03  0.6724869  0.3447525
##   1.321941e-03  0.6721823  0.3441394
##   1.555676e-03  0.6723356  0.3444435
##   1.830738e-03  0.6717240  0.3432168
##   2.154435e-03  0.6721811  0.3441317
##   2.535364e-03  0.6717235  0.3432128
##   2.983647e-03  0.6712632  0.3422928
##   3.511192e-03  0.6705008  0.3407626
##   4.132012e-03  0.6692787  0.3383076
##   4.862602e-03  0.6688195  0.3373833
##   5.722368e-03  0.6666784  0.3330927
##   6.734151e-03  0.6669846  0.3336966
##   7.924829e-03  0.6683620  0.3364449
##   9.326033e-03  0.6691241  0.3379647
##   1.097499e-02  0.6680503  0.3358073
##   1.291550e-02  0.6689680  0.3376311
##   1.519911e-02  0.6688155  0.3373188
##   1.788650e-02  0.6675909  0.3348565
##   2.104904e-02  0.6682027  0.3360617
```

```
##   2.477076e-02  0.6668271  0.3333036
##   2.915053e-02  0.6665212  0.3326958
##   3.430469e-02  0.6660642  0.3317883
##   4.037017e-02  0.6666763  0.3330213
##   4.750810e-02  0.6666760  0.3330266
##   5.590810e-02  0.6666760  0.3330266
##   6.579332e-02  0.6562754  0.3119155
##   7.742637e-02  0.6559696  0.3113039
##   9.111628e-02  0.6406690  0.2804980
##   1.072267e-01  0.6209484  0.2408648
##   1.261857e-01  0.6212542  0.2414764
##   1.484968e-01  0.6212542  0.2414764
##   1.747528e-01  0.6212542  0.2414764
##   2.056512e-01  0.5009174  0.0000000
##   2.420128e-01  0.5009174  0.0000000
##   2.848036e-01  0.5009174  0.0000000
##   3.351603e-01  0.5009174  0.0000000
##   3.944206e-01  0.5009174  0.0000000
##   4.641589e-01  0.5009174  0.0000000
##   5.462277e-01  0.5009174  0.0000000
##   6.428073e-01  0.5009174  0.0000000
##   7.564633e-01  0.5009174  0.0000000
##   8.902151e-01  0.5009174  0.0000000
##   1.047616e+00  0.5009174  0.0000000
##   1.232847e+00  0.5009174  0.0000000
##   1.450829e+00  0.5009174  0.0000000
##   1.707353e+00  0.5009174  0.0000000
##   2.009233e+00  0.5009174  0.0000000
##   2.364489e+00  0.5009174  0.0000000
##   2.782559e+00  0.5009174  0.0000000
##   3.274549e+00  0.5009174  0.0000000
##   3.853529e+00  0.5009174  0.0000000
##   4.534879e+00  0.5009174  0.0000000
##   5.336699e+00  0.5009174  0.0000000
##   6.280291e+00  0.5009174  0.0000000
##   7.390722e+00  0.5009174  0.0000000
##   8.697490e+00  0.5009174  0.0000000
##   1.023531e+01  0.5009174  0.0000000
##   1.204504e+01  0.5009174  0.0000000
##   1.417474e+01  0.5009174  0.0000000
##   1.668101e+01  0.5009174  0.0000000
##   1.963041e+01  0.5009174  0.0000000
##   2.310130e+01  0.5009174  0.0000000
##   2.718588e+01  0.5009174  0.0000000
##   3.199267e+01  0.5009174  0.0000000
##   3.764936e+01  0.5009174  0.0000000
##   4.430621e+01  0.5009174  0.0000000
##   5.214008e+01  0.5009174  0.0000000
##   6.135907e+01  0.5009174  0.0000000
##   7.220809e+01  0.5009174  0.0000000
```

```
##   8.497534e+01  0.5009174  0.0000000
##   1.000000e+02  0.5009174  0.0000000
##
## Tuning parameter 'alpha' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 1 and lambda = 0.00030538
56.
```

Q5)d)

```
set.seed(123)

#Set the grid for the lambda values
lambdaValues <- 10^seq(-5, 2, length = 100)



fitRidge <- train(BadBuy ~ ., family='binomial', data=dfcTrain, method='glmne
t', trControl=trainControl(method='cv', number=10), tuneGrid = expand.grid(al
pha=0, lambda=lambdaValues))

#Variable importance complete table
varImp(fitRidge)$importance %>%    # Add scale=FALSE inside VarImp if you don
't want to scale
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()

## # A tibble: 59 x 3
##    Variable       Overall Importance
##    <chr>            <dbl> <chr>
##  1 WheelTypeNULL    100   100.00000%
##  2 MakeINFINITI      38.9 38.93548%
##  3 MakeLINCOLN       32.2 32.15789%
##  4 ColorOTHER        28.8 28.84262%
##  5 MakeOTHER         24.8 24.79872%
##  6 MakeOLDSMOBILE    21.5 21.50571%
##  7 MakeSCION         20.8 20.77209%
##  8 MakeSUZUKI        20.7 20.74687%
##  9 MakeMITSUBISHI    19.9 19.91029%
## 10 MakeGMC           17.5 17.50910%
## # ... with 49 more rows

#Variable importance plot with the most important variables
plot(varImp(fitRidge),top=25)  # Add top = XX to change the number of visible
variables
```




```
#Optimum lambda selected by the algorithm
fitRidge$bestTune$lambda   # You can also run fitLasso$finalModel$lambdaOpt

## [1] 0.0559081

#Not so useful but helps with understanding -See how variables are dropped as
lambda increases
#plot(fitLasso$finalModel, xvar="lambda", label = TRUE)

#Not so useful but helps with understanding -See the coefficients from the fi
nal lasso model
#coef(fitLasso$finalModel, fitLasso$bestTune$lambda)   # You can also use fit
Lasso$finalModel$lambdaOpt for optimum lambda

resultsRidge <-
  fitRidge %>%
  predict(dfcTest, type='raw') %>%
  bind_cols(dfcTest, predictedClass=.)

resultsRidge %>%
  xtabs(~predictedClass+BadBuy, .) %>%
  confusionMatrix(positive = '1')

## Confusion Matrix and Statistics
##
##               BadBuy
## predictedClass    0    1
```

```
##              0 1323  699
##              1  459 1040
##
##                Accuracy : 0.6711
##                  95% CI : (0.6553, 0.6866)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.341
##
##  Mcnemar's Test P-Value : 2.166e-12
##
##             Sensitivity : 0.5980
##             Specificity : 0.7424
##          Pos Pred Value : 0.6938
##          Neg Pred Value : 0.6543
##              Prevalence : 0.4939
##          Detection Rate : 0.2954
##    Detection Prevalence : 0.4257
##       Balanced Accuracy : 0.6702
##
##        'Positive' Class : 1
##
```

```
fitRidge
```

```
## glmnet
##
## 6540 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5885, 5886, 5887, 5887, 5885, 5886, ...
## Resampling results across tuning parameters:
##
##   lambda        Accuracy   Kappa
##   1.000000e-05  0.6720303  0.3438493
##   1.176812e-05  0.6720303  0.3438493
##   1.384886e-05  0.6720303  0.3438493
##   1.629751e-05  0.6720303  0.3438493
##   1.917910e-05  0.6720303  0.3438493
##   2.257020e-05  0.6720303  0.3438493
##   2.656088e-05  0.6720303  0.3438493
##   3.125716e-05  0.6720303  0.3438493
##   3.678380e-05  0.6720303  0.3438493
##   4.328761e-05  0.6720303  0.3438493
##   5.094138e-05  0.6720303  0.3438493
##   5.994843e-05  0.6720303  0.3438493
```

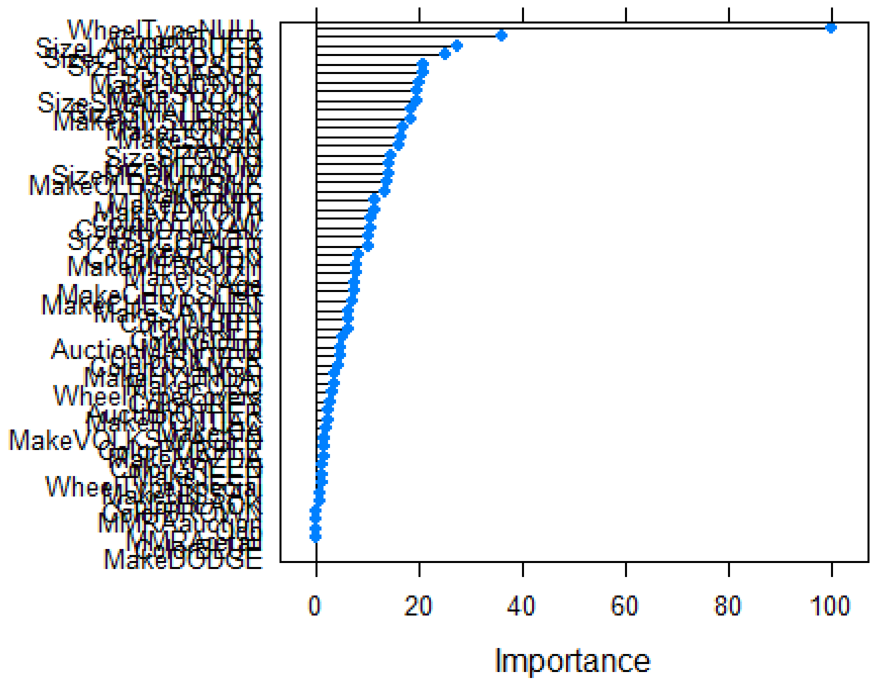
```
##   7.054802e-05  0.6720303  0.3438493
##   8.302176e-05  0.6720303  0.3438493
##   9.770100e-05  0.6720303  0.3438493
##   1.149757e-04  0.6720303  0.3438493
##   1.353048e-04  0.6720303  0.3438493
##   1.592283e-04  0.6720303  0.3438493
##   1.873817e-04  0.6720303  0.3438493
##   2.205131e-04  0.6720303  0.3438493
##   2.595024e-04  0.6720303  0.3438493
##   3.053856e-04  0.6720303  0.3438493
##   3.593814e-04  0.6720303  0.3438493
##   4.229243e-04  0.6720303  0.3438493
##   4.977024e-04  0.6720303  0.3438493
##   5.857021e-04  0.6720303  0.3438493
##   6.892612e-04  0.6720303  0.3438493
##   8.111308e-04  0.6720303  0.3438493
##   9.545485e-04  0.6720303  0.3438493
##   1.123324e-03  0.6720303  0.3438493
##   1.321941e-03  0.6720303  0.3438493
##   1.555676e-03  0.6720303  0.3438493
##   1.830738e-03  0.6720303  0.3438493
##   2.154435e-03  0.6720303  0.3438493
##   2.535364e-03  0.6720303  0.3438493
##   2.983647e-03  0.6720303  0.3438493
##   3.511192e-03  0.6720303  0.3438493
##   4.132012e-03  0.6720303  0.3438493
##   4.862602e-03  0.6720303  0.3438493
##   5.722368e-03  0.6720303  0.3438493
##   6.734151e-03  0.6720303  0.3438493
##   7.924829e-03  0.6720303  0.3438493
##   9.326033e-03  0.6720303  0.3438493
##   1.097499e-02  0.6720303  0.3438493
##   1.291550e-02  0.6720303  0.3438493
##   1.519911e-02  0.6720303  0.3438493
##   1.788650e-02  0.6718771  0.3435427
##   2.104904e-02  0.6714182  0.3426252
##   2.477076e-02  0.6724883  0.3447662
##   2.915053e-02  0.6726405  0.3450732
##   3.430469e-02  0.6724885  0.3447713
##   4.037017e-02  0.6724909  0.3447774
##   4.750810e-02  0.6721848  0.3441688
##   5.590810e-02  0.6734076  0.3466187
##   6.579332e-02  0.6727965  0.3454017
##   7.742637e-02  0.6711157  0.3420419
##   9.111628e-02  0.6692813  0.3383774
##   1.072267e-01  0.6688237  0.3374670
##   1.261857e-01  0.6705040  0.3408389
##   1.484968e-01  0.6703523  0.3405422
##   1.747528e-01  0.6689771  0.3377974
##   2.056512e-01  0.6686722  0.3371928
```

```
##   2.420128e-01  0.6685191  0.3368930
##   2.848036e-01  0.6674487  0.3347573
##   3.351603e-01  0.6669910  0.3338472
##   3.944206e-01  0.6674492  0.3347690
##   4.641589e-01  0.6674473  0.3347721
##   5.462277e-01  0.6671418  0.3341655
##   6.428073e-01  0.6691281  0.3381464
##   7.564633e-01  0.6686701  0.3372317
##   8.902151e-01  0.6677527  0.3354014
##   1.047616e+00  0.6674464  0.3347907
##   1.232847e+00  0.6668345  0.3335684
##   1.450829e+00  0.6669870  0.3338752
##   1.707353e+00  0.6668348  0.3335704
##   2.009233e+00  0.6662232  0.3323469
##   2.364489e+00  0.6659176  0.3317352
##   2.782559e+00  0.6662241  0.3323476
##   3.274549e+00  0.6665294  0.3329564
##   3.853529e+00  0.6663761  0.3326458
##   4.534879e+00  0.6659183  0.3317263
##   5.336699e+00  0.6665299  0.3329428
##   6.280291e+00  0.6666828  0.3332437
##   7.390722e+00  0.6672944  0.3344623
##   8.697490e+00  0.6679049  0.3356706
##   1.023531e+01  0.6663747  0.3325993
##   1.204504e+01  0.6659169  0.3316659
##   1.417474e+01  0.6686710  0.3371537
##   1.668101e+01  0.6683655  0.3365257
##   1.963041e+01  0.6680580  0.3358789
##   2.310130e+01  0.6668345  0.3334016
##   2.718588e+01  0.6639277  0.3275498
##   3.199267e+01  0.6610202  0.3216894
##   3.764936e+01  0.6596445  0.3188901
##   4.430621e+01  0.6542937  0.3081220
##   5.214008e+01  0.6504699  0.3004086
##   6.135907e+01  0.6415970  0.2825817
##   7.220809e+01  0.6307397  0.2607601
##   8.497534e+01  0.6272166  0.2536259
##   1.000000e+02  0.6188082  0.2367086
##
## Tuning parameter 'alpha' was held constant at a value of 0
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0 and lambda = 0.0559081.
```

Q5d)II)

```
# Elastic net with alpha = 0.5
# [*Naive elastic net - Read "Zou & Hastie (2005) Regularization and variable
selection via the elastic net" for details]

#Set the grid for the lambda values
```

```
lambdaValues <- 10^seq(-5, 2, length = 100)

set.seed(2020)

fitElastic <- train(BadBuy ~ ., family='binomial', data=dfcTrain, method='glm
net', trControl=trainControl(method='cv', number=10), tuneGrid=expand.grid(al
pha=0.5, lambda=lambdaValues))

#Variable importance complete table
varImp(fitElastic)$importance %>%    # Add scale=FALSE inside VarImp if you d
on't want to scale
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()

## # A tibble: 59 x 3
##    Variable       Overall Importance
##    <chr>            <dbl> <chr>
##  1 WheelTypeNULL    100   100%
##  2 ColorOTHER        35.9 36%
##  3 SizeLARGETRUCK    27.5 27%
##  4 SizeCROSSOVER     25.2 25%
##  5 SizeLARGESUV      20.8 21%
##  6 SizeLARGE         20.8 21%
##  7 MakeLINCOLN       20.1 20%
##  8 MakeSUZUKI        19.7 20%
##  9 SizeSMALLTRUCK    19.6 20%
## 10 SizeSMALLSUV      18.4 18%
## # ... with 49 more rows

#Variable importance plot with the most important variables
plot(varImp(fitElastic))    # Add top = XX to change the number of visible va
riables
```




```
#Optimum lambda selected by the algorithm
fitElastic$bestTune$lambda   # You can also run fitElastic$finalModel$lambdaO
pt

## [1] 0.0003053856

#Not so useful but helps with understanding -See how variables are dropped as
lambda increases
#plot(fitElastic$finalModel, xvar="lambda", label = TRUE)

#Not so useful but helps with understanding -See the coefficients from the fi
nal Elastic model
#coef(fitElastic$finalModel, fitElastic$bestTune$lambda)   # You can also use
fitElastic$finalModel$lambdaOpt for optimum lambda

resultsElastic <-
  fitElastic %>%
  predict(dfcTest, type='raw') %>%
  bind_cols(dfcTest, predictedClass=.)

resultsElastic %>%
  xtabs(~predictedClass+BadBuy, .) %>%
  confusionMatrix(positive = '1')

## Confusion Matrix and Statistics
##
##                BadBuy
```

```
## predictedClass    0    1
##              0 1342  720
##              1  440 1019
##
##                Accuracy : 0.6705
##                  95% CI : (0.6547, 0.6861)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3397
##
##  Mcnemar's Test P-Value : 2.575e-16
##
##             Sensitivity : 0.5860
##             Specificity : 0.7531
##          Pos Pred Value : 0.6984
##          Neg Pred Value : 0.6508
##              Prevalence : 0.4939
##          Detection Rate : 0.2894
##    Detection Prevalence : 0.4144
##       Balanced Accuracy : 0.6695
##
##        'Positive' Class : 1
##
```

Q5)e)

```
set.seed(123)

resultsCaret<-
  train(BadBuy ~ ., family='binomial', data=dfcTrain, method='qda',
              trControl = trainControl(method = "cv")) %>%
  predict(dfcTest, type='raw')%>%
  bind_cols(dfcTest, predictedClass=.)

## Warning: model fit failed for Fold03: parameter=none Error in qda.default(
x, grouping, ...) : rank deficiency in group 0

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainI
nfo, :
## There were missing values in resampled performance measures.

resultsCaret%>%
  xtabs(~predictedClass+BadBuy, .)%>%
  confusionMatrix(positive='1')

## Confusion Matrix and Statistics
##
##                BadBuy
## predictedClass    0    1
##              0 1483  973
```

```
##             1  299  766
##
##                Accuracy : 0.6387
##                  95% CI : (0.6226, 0.6546)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.274
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4405
##             Specificity : 0.8322
##          Pos Pred Value : 0.7192
##          Neg Pred Value : 0.6038
##              Prevalence : 0.4939
##          Detection Rate : 0.2176
##    Detection Prevalence : 0.3025
##       Balanced Accuracy : 0.6363
##
##        'Positive' Class : 1
##
```