Assignment 3

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:\Program Files\R\R-3.6.2")

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()
```

```
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
```

```
library("skimr")
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
```

Question 1

```
dfc <- read_csv("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()
## )

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

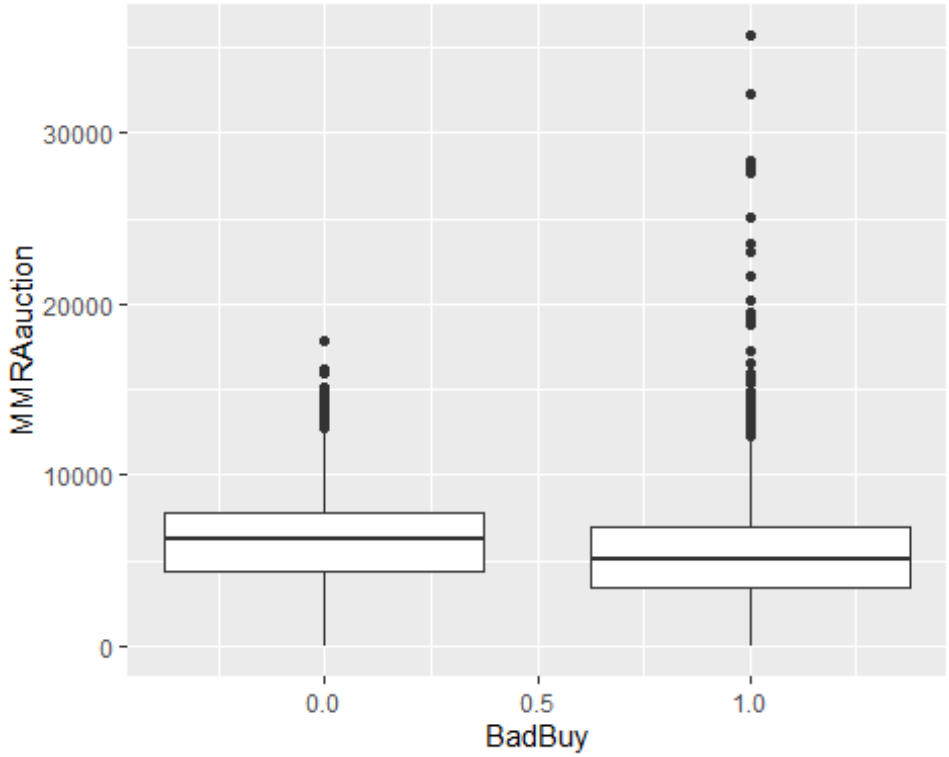
Variable type: numeric

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	▇▁▁▁▇

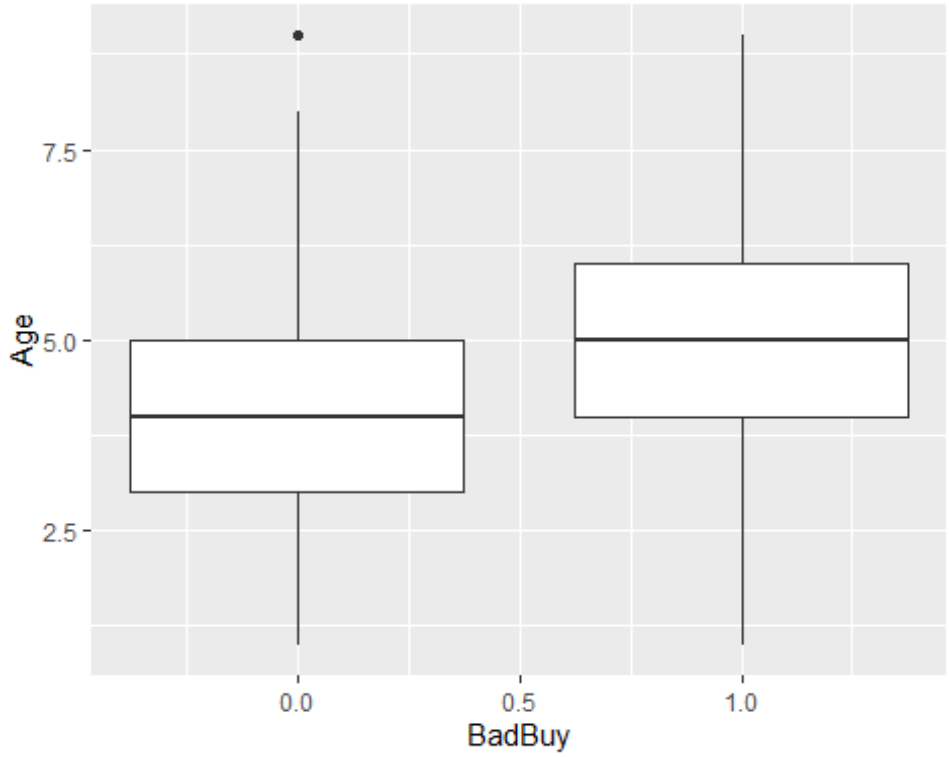
```r
set.seed(52156)

dfcTrain <- dfc %>% sample_frac(0.65)
dfcTest <- dplyr::setdiff(dfc, dfcTrain)
```

Question 2

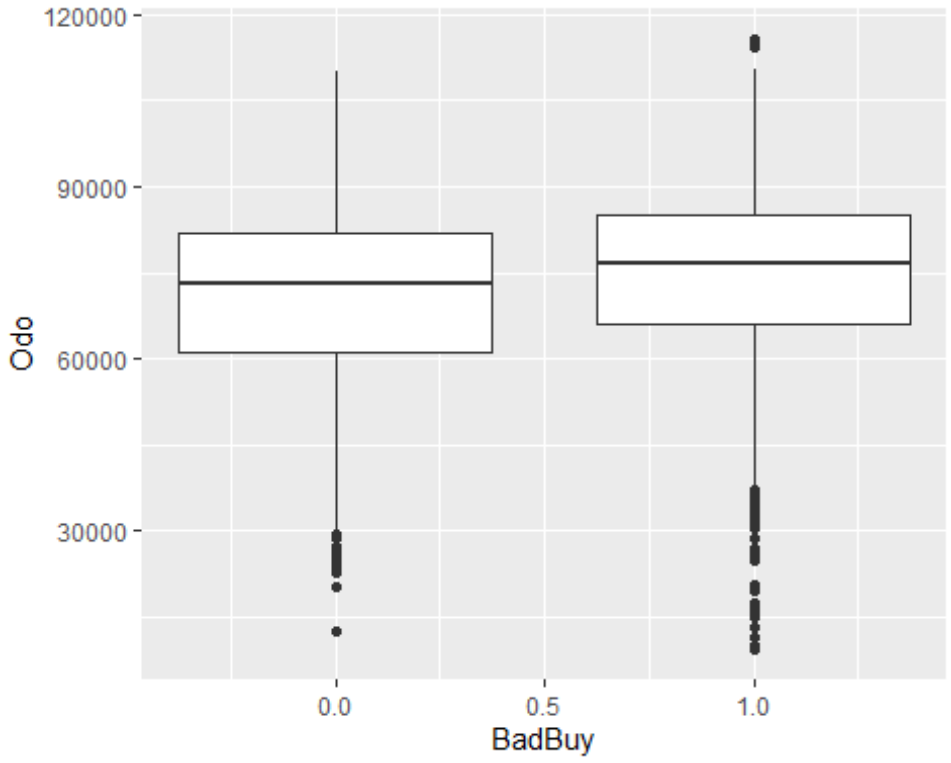
```r
# 2.a

boxPlotsForAuction <- dfc %>%
  ggplot(aes(x = BadBuy, y = MMRAauction, group = BadBuy)) +
  geom_boxplot()

boxPlotsForAuction
```




```r
boxPlotsForAge <- dfc %>%
  ggplot(aes(x = BadBuy, y = Age, group = BadBuy)) +
  geom_boxplot()

boxPlotsForAge
```




```
boxPlotsForOdometer <- dfc %>%
  ggplot(aes(x = BadBuy, y = Odo, group = BadBuy)) +
  geom_boxplot()

boxPlotsForOdometer
```




```
# 2.b

a <- table(dfc$Size, dfc$BadBuy)
report <- as.data.frame(a)
report <- spread(report, key = Var2, value = Freq)
report$Total = report$`0` + report$`1`
report <- setNames(report, c("Size", "Good_investment", "Lemons",
"Total_cars"))
arrange(report, desc(report$Total_cars))
```

```
##          Size Good_investment Lemons Total_cars
## 1      MEDIUM            2122   1986       4108
## 2   MEDIUMSUV             534    647       1181
## 3     COMPACT             475    647       1122
## 4       LARGE             654    460       1114
## 5         VAN             407    412        819
## 6  LARGETRUCK             234    207        441
## 7    SMALLSUV             144    163        307
## 8   CROSSOVER             146     96        242
## 9   SPECIALTY             131    105        236
## 10   LARGESUV              82    117        199
## 11     SPORTS              67     82        149
## 12 SMALLTRUCK              62     81        143
```

```
dfc %>%
  group_by(BadBuy) %>%
  tally()
```

```
## # A tibble: 2 x 2
##   BadBuy     n
##    <dbl> <int>
## 1      0  5058
## 2      1  5003
```

```
report$Perc_of_lemons = report$Lemons/5003 * 100
arrange(report, desc(report$Perc_of_lemons))
```

```
##          Size Good_investment Lemons Total_cars Perc_of_lemons
## 1      MEDIUM            2122   1986       4108      39.696182
## 2     COMPACT             475    647       1122      12.932241
## 3   MEDIUMSUV             534    647       1181      12.932241
## 4       LARGE             654    460       1114       9.194483
## 5         VAN             407    412        819       8.235059
## 6  LARGETRUCK             234    207        441       4.137517
## 7    SMALLSUV             144    163        307       3.258045
## 8    LARGESUV              82    117        199       2.338597
## 9   SPECIALTY             131    105        236       2.098741
## 10  CROSSOVER             146     96        242       1.918849
## 11     SPORTS              67     82        149       1.639017
## 12 SMALLTRUCK              62     81        143       1.619029
```

Question 3

```
fitLPM  <- lm(BadBuy ~ ., data = dfcTrain)

summary(fitLPM)
```

```
##
## Call:
## lm(formula = BadBuy ~ ., data = dfcTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2353 -0.3934 -0.1635  0.4658  0.9587
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.996e-01  2.394e-01  -0.834  0.40434
## AuctionMANHEIM    4.065e-02  1.490e-02   2.728  0.00638 **
## AuctionOTHER      2.287e-02  1.706e-02   1.341  0.18008
## Age               5.154e-02  5.619e-03   9.172  < 2e-16 ***
## MakeBUICK         2.392e-01  2.360e-01   1.013  0.31089
## MakeCADILLAC      2.664e-01  5.045e-01   0.528  0.59756
## MakeCHEVROLET     1.861e-01  2.299e-01   0.810  0.41820
## MakeCHRYSLER      2.944e-01  2.297e-01   1.282  0.19993
## MakeDODGE         2.384e-01  2.293e-01   1.040  0.29853
## MakeFORD          2.620e-01  2.298e-01   1.140  0.25427
## MakeGMC           1.398e-01  2.379e-01   0.588  0.55685
```

```
## MakeHONDA          1.114e-01  2.374e-01   0.469  0.63904
## MakeHYUNDAI        2.099e-01  2.321e-01   0.904  0.36578
## MakeINFINITI       3.671e-01  3.201e-01   1.147  0.25141
## MakeISUZU          1.764e-01  2.747e-01   0.642  0.52082
## MakeJEEP           2.537e-01  2.331e-01   1.089  0.27638
## MakeKIA            2.190e-01  2.316e-01   0.946  0.34440
## MakeLEXUS          8.805e-01  3.221e-01   2.733  0.00629 **
## MakeLINCOLN        3.712e-01  2.577e-01   1.440  0.14980
## MakeMAZDA          2.567e-01  2.329e-01   1.102  0.27036
## MakeMERCURY        2.980e-01  2.337e-01   1.275  0.20229
## MakeMINI           3.301e-01  3.082e-01   1.071  0.28422
## MakeMITSUBISHI     1.179e-01  2.338e-01   0.504  0.61396
## MakeNISSAN         2.310e-01  2.313e-01   0.999  0.31801
## MakeOLDSMOBILE     3.261e-01  2.441e-01   1.336  0.18156
## MakePONTIAC        2.181e-01  2.306e-01   0.946  0.34427
## MakeSATURN         2.800e-01  2.316e-01   1.209  0.22684
## MakeSCION          1.091e-01  2.669e-01   0.409  0.68272
## MakeSUBARU         2.432e-01  3.922e-01   0.620  0.53520
## MakeSUZUKI         3.696e-01  2.335e-01   1.583  0.11354
## MakeTOYOTA         1.638e-01  2.341e-01   0.700  0.48414
## MakeVOLKSWAGEN     2.630e-01  2.613e-01   1.007  0.31409
## MakeVOLVO         -1.809e-01  3.906e-01  -0.463  0.64322
## ColorBLACK         2.220e-02  4.160e-02   0.534  0.59365
## ColorBLUE          1.890e-02  4.055e-02   0.466  0.64111
## ColorBROWN         1.819e-02  7.917e-02   0.230  0.81826
## ColorGOLD          5.438e-02  4.271e-02   1.273  0.20298
## ColorGREEN         2.264e-02  4.620e-02   0.490  0.62408
## ColorGREY          3.804e-02  4.137e-02   0.919  0.35793
## ColorMAROON        7.248e-02  5.097e-02   1.422  0.15503
## ColorNOTAVAIL     -4.753e-02  1.265e-01  -0.376  0.70717
## ColorNULL         -1.179e-01  4.546e-01  -0.259  0.79543
## ColorORANGE        4.598e-02  8.977e-02   0.512  0.60852
## ColorOTHER        -1.388e-01  9.958e-02  -1.394  0.16327
## ColorPURPLE        1.955e-02  8.259e-02   0.237  0.81289
## ColorRED           6.169e-02  4.214e-02   1.464  0.14326
## ColorSILVER        4.814e-02  3.960e-02   1.216  0.22418
## ColorWHITE         6.047e-02  4.013e-02   1.507  0.13186
## ColorYELLOW       -6.072e-02  1.016e-01  -0.597  0.55031
## WheelTypeCovers   -3.534e-02  1.395e-02  -2.533  0.01134 *
## WheelTypeNULL      5.096e-01  1.861e-02  27.379  < 2e-16 ***
## WheelTypeSpecial  -8.805e-03  5.743e-02  -0.153  0.87815
## Odo                2.888e-06  4.327e-07   6.675 2.69e-11 ***
## SizeCROSSOVER     -1.783e-01  4.404e-02  -4.048 5.23e-05 ***
## SizeLARGE         -1.475e-01  2.616e-02  -5.640 1.77e-08 ***
## SizeLARGESUV      -1.379e-01  4.893e-02  -2.817  0.00486 **
## SizeLARGETRUCK    -1.916e-01  3.669e-02  -5.224 1.81e-07 ***
## SizeMEDIUM        -9.926e-02  2.020e-02  -4.913 9.18e-07 ***
## SizeMEDIUMSUV     -9.874e-02  2.840e-02  -3.477  0.00051 ***
## SizeSMALLSUV      -1.333e-01  4.231e-02  -3.149  0.00164 **
## SizeSMALLTRUCK    -1.449e-01  5.170e-02  -2.803  0.00508 **
```

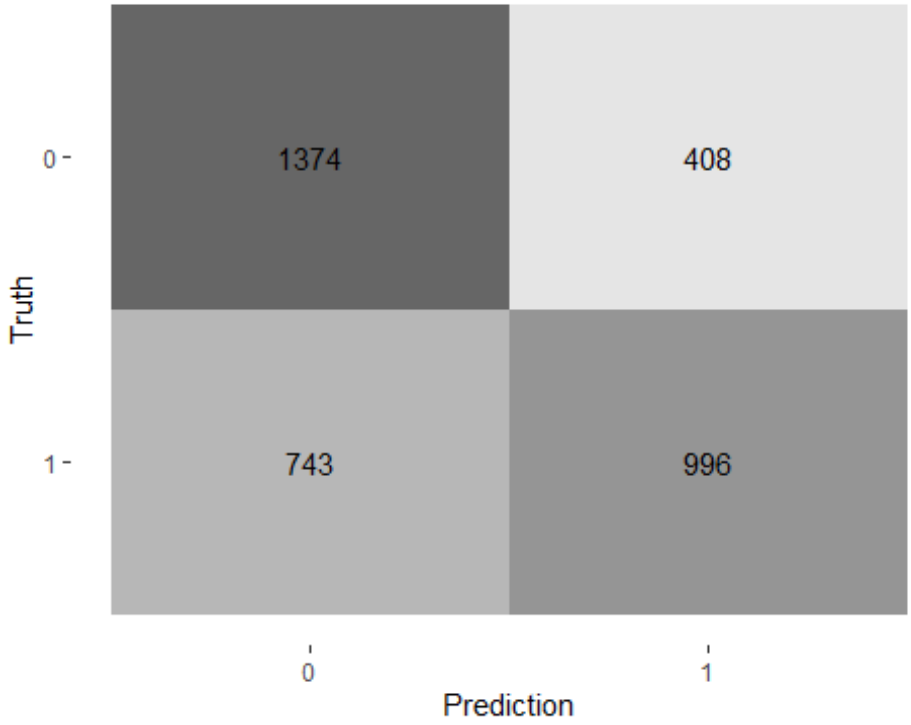
```
## SizeSPECIALTY    -7.220e-02  4.718e-02  -1.530  0.12599
## SizeSPORTS       -1.081e-01  5.064e-02  -2.135  0.03277 *
## SizeVAN          -1.136e-01  2.727e-02  -4.164 3.16e-05 ***
## MMRAauction       1.595e-06  7.264e-06   0.220  0.82626
## MMRAretail       -1.126e-06  4.514e-06  -0.249  0.80302
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4502 on 6474 degrees of freedom
## Multiple R-squared:  0.1975, Adjusted R-squared:  0.1894
## F-statistic: 24.51 on 65 and 6474 DF,  p-value: < 2.2e-16

# 3.a

resultsTrain <- dfcTrain %>%
            mutate(predictedBadBuy = predict(fitLPM, dfcTrain))

resultsTrain

## # A tibble: 6,540 x 11
##    Auction   Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <chr>   <dbl> <chr> <chr> <chr>     <dbl> <chr>       <dbl>      <dbl>
<dbl>
##  1 MANHEIM     4 FORD  SILV~ NULL      77591 LARG~        9774      14506
1
##  2 MANHEIM     5 MINI  BLUE  Alloy     80013 COMP~       11040      12423
1
##  3 MANHEIM     2 CHEV~ SILV~ Covers    75493 LARGE        9707      13975
1
##  4 ADESA       4 NISS~ BLUE  NULL      84827 MEDI~        6073       9791
1
##  5 MANHEIM     5 FORD  GREY  Alloy     57388 SPOR~        5574       8984
1
##  6 ADESA       4 SUZU~ BLACK NULL      75822 MEDI~        4033       6979
1
##  7 MANHEIM     2 KIA   BLACK Covers    51059 MEDI~        4839       5726
0
##  8 OTHER       7 FORD  GREY  NULL      74595 LARG~        7649      11059
1
##  9 MANHEIM     6 FORD  BLUE  Alloy     80328 LARG~        6172       7166
0
## 10 MANHEIM     8 PONT~ WHITE Alloy     97173 LARGE        3242       6225
0
## # ... with 6,530 more rows, and 1 more variable: predictedBadBuy <dbl>

performanceTrain <- metric_set(rmse, mae)
performanceTrain(resultsTrain, truth = BadBuy, estimate = predictedBadBuy)

## # A tibble: 2 x 3
##   .metric .estimator .estimate
```

```
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.448
## 2 mae     standard       0.410

resultsTest <- dfcTest %>%
            mutate(predictedBadBuy = predict(fitLPM, dfcTest))

resultsTest

## # 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: predictedBadBuy <dbl>

performanceTest <- metric_set(rmse, mae)
performanceTest(resultsTest, truth = BadBuy, estimate = predictedBadBuy)

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

# 3.c

colsToFactor <- c("BadBuy")
dfc <- dfc %>%
  mutate_at(colsToFactor, ~factor(.))
```

```r
colsToFactor1 <- c("BadBuy")
dfcTrain <- dfcTrain %>%
  mutate_at(colsToFactor1, ~factor(.))

colsToFactor2 <- c("BadBuy")
dfcTest <- dfcTest %>%
  mutate_at(colsToFactor2, ~factor(.))

resultsTest1 <-
    fitLPM %>%
    predict(dfcTest, type = "response") %>%
    bind_cols(dfcTest, predictedProb = .) %>%
    mutate(predictedClass = as.factor(ifelse(predictedProb > 0.5, 1, 0)))

resultsTest1
```

```
## # 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>
```

```r
resultsTest1 %>%
  conf_mat(truth = BadBuy, estimate = predictedClass) %>%
  autoplot(type = "heatmap")
```




```
# 3.d

resultsTest1 %>%
  xtabs(~predictedClass + BadBuy, .) %>%
  confusionMatrix(positive = "1")

## Confusion Matrix and Statistics
##
##               BadBuy
## predictedClass    0    1
##              0 1374  743
##              1  408  996
##
##                Accuracy : 0.6731
##                  95% CI : (0.6573, 0.6886)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3446
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5727
##             Specificity : 0.7710
##          Pos Pred Value : 0.7094
##          Neg Pred Value : 0.6490
##              Prevalence : 0.4939
```

```
##          Detection Rate : 0.2829
##    Detection Prevalence : 0.3988
##       Balanced Accuracy : 0.6719
##
##        'Positive' Class : 1
##
```

Question 4

```
colsToFactor <- c("Auction", "Make", "Color", "WheelType", "Size")
dfc <- dfc %>%
  mutate_at(colsToFactor, ~factor(.))

colsToFactor1 <- c("Auction", "Make", "Color", "WheelType", "Size")
dfcTrain <- dfcTrain %>%
  mutate_at(colsToFactor1, ~factor(.))

colsToFactor2 <- c("Auction", "Make", "Color", "WheelType", "Size")
dfcTest <- dfcTest %>%
  mutate_at(colsToFactor2, ~factor(.))

fitLGM1 <- train(BadBuy ~ ., family = "binomial", data = dfcTrain, method =
"glm")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
```

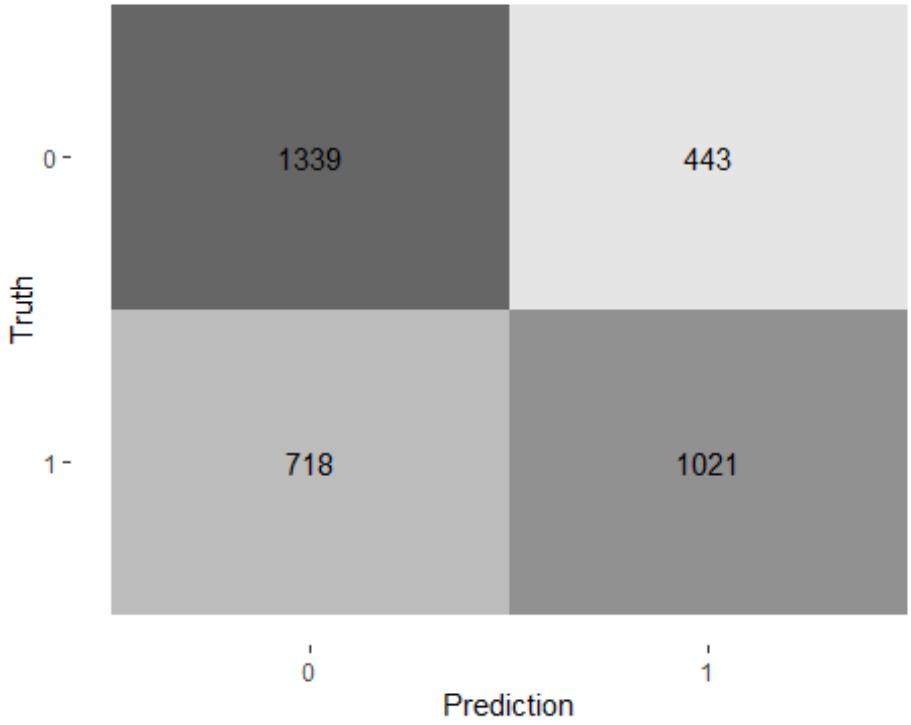
```
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading
```

```r
resultsCaret1 <-    fitLGM1 %>%
    predict(dfcTest, type = "raw") %>%
```

```
    bind_cols(dfcTest, predictedClass = .)

resultsCaret1
```

```
## # A tibble: 3,521 x 11
##    Auction   Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <fct>   <dbl> <fct> <fct> <fct>     <dbl> <fct>       <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 1 more variable: predictedClass <fct>
```

```
resultsCaret1 %>%
  conf_mat(truth = BadBuy, estimate = predictedClass) %>%
  autoplot(type = "heatmap")
```




```
resultsCaret1 %>%
  xtabs(~predictedClass + BadBuy, .) %>%
  confusionMatrix(positive = "1")
```

```
## Confusion Matrix and Statistics
##
##                BadBuy
## predictedClass    0    1
##              0 1339  718
##              1  443 1021
##
##                Accuracy : 0.6703
##                  95% CI : (0.6545, 0.6858)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3392
##
##  Mcnemar's Test P-Value : 8.878e-16
##
##             Sensitivity : 0.5871
##             Specificity : 0.7514
##          Pos Pred Value : 0.6974
##          Neg Pred Value : 0.6509
##              Prevalence : 0.4939
##          Detection Rate : 0.2900
##    Detection Prevalence : 0.4158
```

```
##       Balanced Accuracy : 0.6693
##
##        'Positive' Class : 1
##
```

```
# install.packages("rockchalk")
library(rockchalk)
```

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

```
##
## Attaching package: 'rockchalk'
```

```
## The following object is masked from 'package:dplyr':
##
##     summarize
```

```
# 4.a
```

```
dfc$Color <- combineLevels(dfc$Color, levs = c("NULL", "NOTAVAIL"), newLabel
= "NULL")
```

```
## The original levels BEIGE BLACK BLUE BROWN GOLD GREEN GREY MAROON NOTAVAIL
NULL ORANGE OTHER PURPLE RED SILVER WHITE YELLOW
## have been replaced by BEIGE BLACK BLUE BROWN GOLD GREEN GREY MAROON ORANGE
OTHER PURPLE RED SILVER WHITE YELLOW NULL
```

```
dfc$Make <- combineLevels(dfc$Make, levs = c("ACURA", "CADILLAC", "LEXUS",
"MINI", "SUBARU", "VOLVO"), newLabel = "OTHER")
```

```
## The original levels ACURA BUICK CADILLAC CHEVROLET CHRYSLER DODGE FORD GMC
HONDA HYUNDAI INFINITI ISUZU JEEP KIA LEXUS LINCOLN MAZDA MERCURY MINI
MITSUBISHI NISSAN OLDSMOBILE PONTIAC SATURN SCION SUBARU SUZUKI TOYOTA
VOLKSWAGEN VOLVO
## have been replaced by BUICK CHEVROLET CHRYSLER DODGE FORD GMC HONDA
HYUNDAI INFINITI ISUZU JEEP KIA LINCOLN MAZDA MERCURY MITSUBISHI NISSAN
OLDSMOBILE PONTIAC SATURN SCION SUZUKI TOYOTA VOLKSWAGEN OTHER
```

```
set.seed(52156)
```

```
dfcTrain1 <- dfc %>% sample_frac(0.65)
dfcTest1 <- dplyr::setdiff(dfc, dfcTrain1)
```

```
fitLGM2 <- train(BadBuy ~ ., family = "binomial", data = dfcTrain1, method =
"glm")
```

```
resultsCaret2 <-    fitLGM2 %>%
    predict(dfcTest1, type = "raw") %>%
    bind_cols(dfcTest1, predictedClass = .)
```

```
resultsCaret2
```

```
## # A tibble: 3,521 x 11
##    Auction   Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <fct>   <dbl> <fct> <fct> <fct>     <dbl> <fct>       <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 1 more variable: predictedClass <fct>
```

```
# 4.b & 4.c

summary(fitLGM2)
```

```
##
## Call:
## NULL
##
## 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
## 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
## MakeOTHER          3.109e-01  6.256e-01   0.497 0.619240
## 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
## 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
## ColorNULL         -2.898e-01  7.521e-01  -0.385 0.700011
## 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
```

```
# 4.d
```

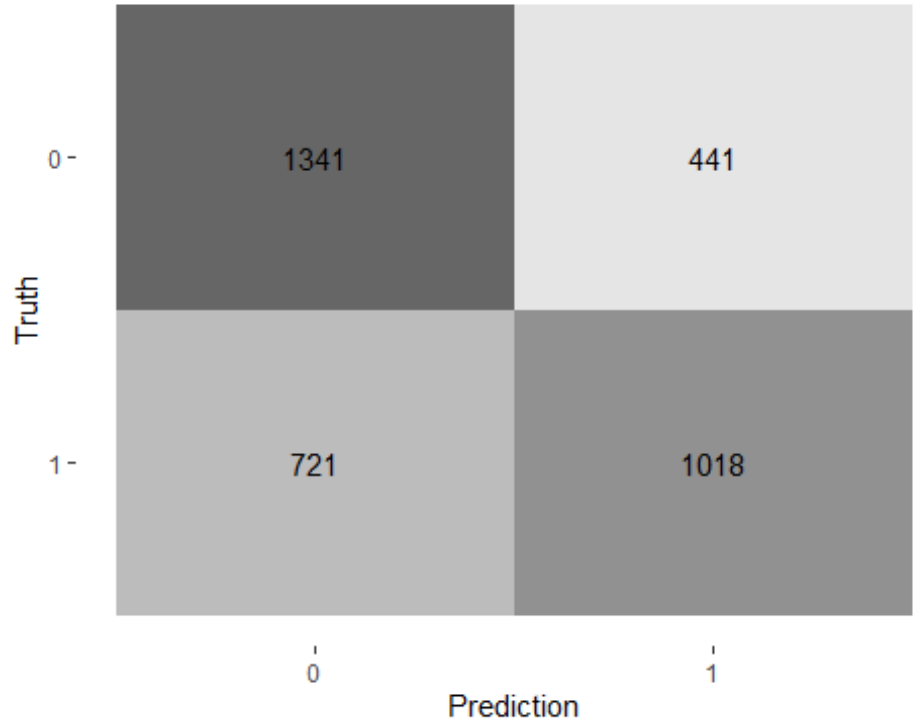
```
fitLGMresults2 <-
    fitLGM2 %>%
    predict(dfcTest1, type = "raw") %>%
    bind_cols(dfcTest1, predictedProb = .) %>%
    mutate(predictedClass = as.factor(ifelse(predictedProb > 0.5, 1, 0)))
```

```
## Warning in Ops.factor(predictedProb, 0.5): '>' not meaningful for factors
```

```
fitLGMresults2
```

```
## # A tibble: 3,521 x 12
##    Auction   Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <fct>   <dbl> <fct> <fct> <fct>     <dbl> <fct>       <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 <fct>,
## #   predictedClass <fct>
```

```
resultsCaret2 %>%
  conf_mat(truth = BadBuy, estimate = predictedClass) %>%
  autoplot(type = "heatmap")
```




```
resultsCaret2 %>%
  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
##
```

```
# 4.e

resultsCaret2 %>%
  xtabs(~predictedClass + BadBuy, .) %>%
  confusionMatrix(positive = "1") %>%
  tidy()
```

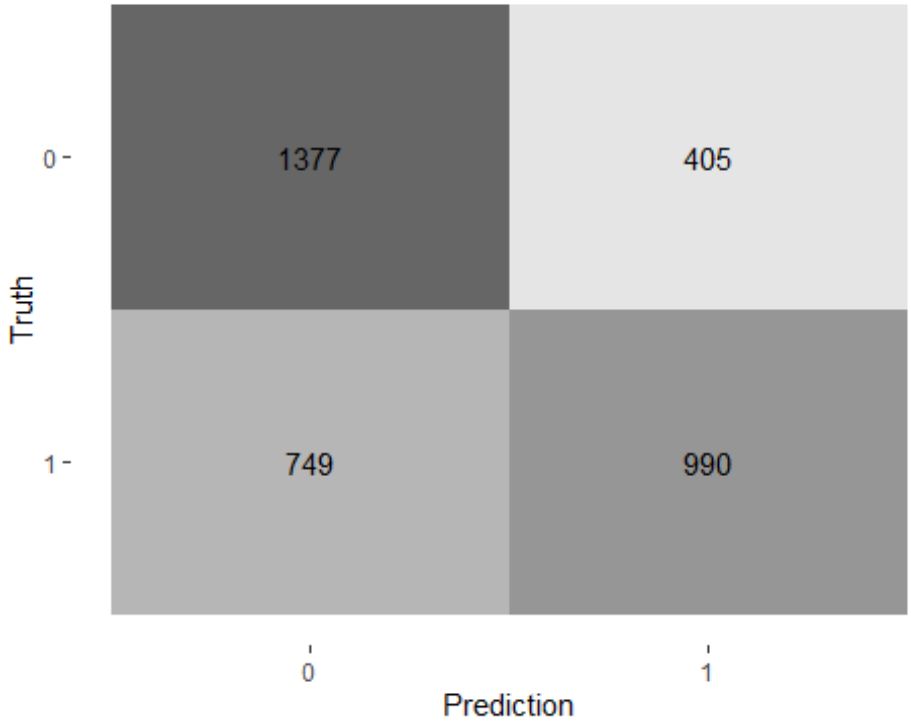
```
## # A tibble: 13 x 6
##    term                class estimate conf.low conf.high   p.value
##    <chr>               <chr>    <dbl>    <dbl>     <dbl>     <dbl>
##  1 accuracy            <NA>     0.670    0.654     0.686  3.50e-86
##  2 kappa               <NA>     0.339   NA        NA      2.73e-16
##  3 sensitivity         1        0.585   NA        NA     NA
##  4 specificity         1        0.753   NA        NA     NA
##  5 pos_pred_value      1        0.698   NA        NA     NA
##  6 neg_pred_value      1        0.650   NA        NA     NA
##  7 precision           1        0.698   NA        NA     NA
##  8 recall              1        0.585   NA        NA     NA
##  9 f1                  1        0.637   NA        NA     NA
## 10 prevalence          1        0.494   NA        NA     NA
## 11 detection_rate      1        0.289   NA        NA     NA
## 12 detection_prevalence 1       0.414   NA        NA     NA
## 13 balanced_accuracy   1        0.669   NA        NA     NA
```

Question 5

```
# 5.a

set.seed(123)

fitLDA <- train(BadBuy ~ ., family = 'binomial', data = dfcTrain1, method =
'lda', trControl = trainControl(method = 'cv', number = 10))

resultsLDA <-   fitLDA %>%
    predict(dfcTest1, type = "raw") %>%
    bind_cols(dfcTest1, predictedClass = .)

resultsLDA
```

```
## # A tibble: 3,521 x 11
##    Auction   Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
```

```
BadBuy
##    <fct>   <dbl> <fct> <fct> <fct>     <dbl> <fct>       <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 1 more variable: predictedClass <fct>

resultsLDA %>%
  conf_mat(truth = BadBuy, estimate = predictedClass) %>%
  autoplot(type = "heatmap")
```

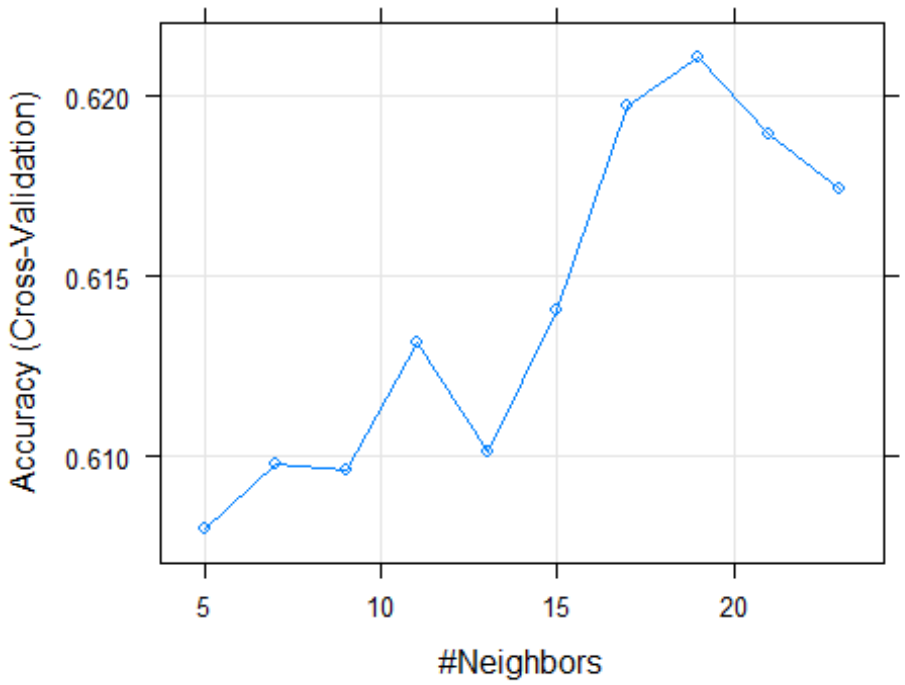



```
resultsLDA %>%
  xtabs(~predictedClass + BadBuy, .) %>%
  confusionMatrix(positive = "1")
```

```
## 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
##
```

```
# 5.b.i

library(class)

set.seed(123)

fitKNN <- train(BadBuy ~ ., data=dfcTrain1, method = 'knn', preProcess =
c("center","scale"), trControl = trainControl(method = 'cv', number = 10),
tuneLength = 10)

plot(fitKNN)
```




```
#5.b.iii

resultsKNN <-   fitKNN %>%
    predict(dfcTest1, type = "raw") %>%
    bind_cols(dfcTest1, predictedClass = .)

resultsKNN
```

```
## # A tibble: 3,521 x 11
##    Auction   Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
```

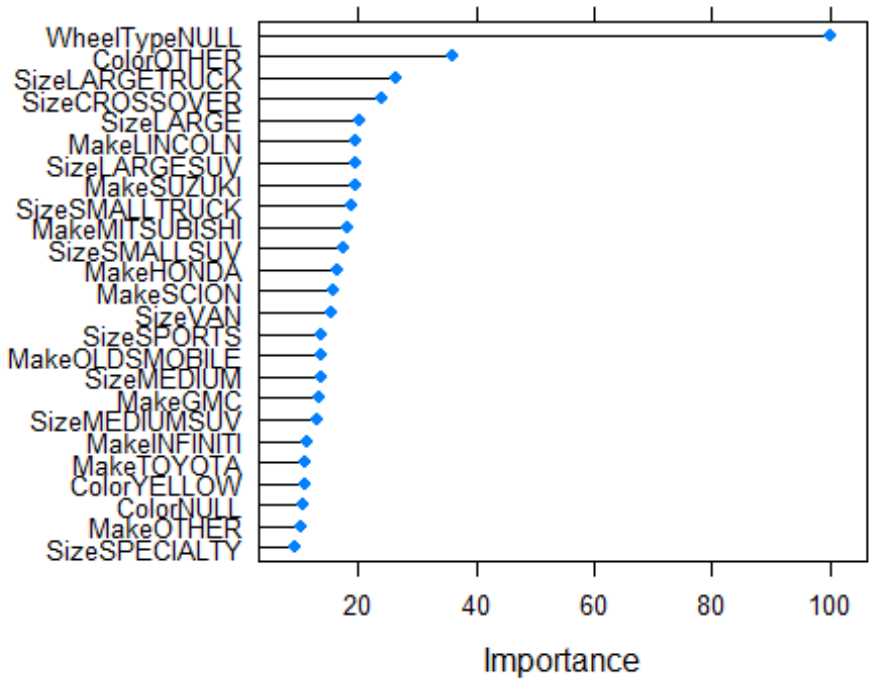
```
BadBuy
##    <fct>   <dbl> <fct> <fct> <fct>     <dbl> <fct>       <dbl>      <dbl>
<fct>
##  1 MANHEIM     6 SATU~ WHITE Covers    81116 MEDI~        2667       3380
0
##  2 OTHER       5 CHEV~ RED   Alloy     54718 MEDI~        6921       7975
1
##  3 OTHER       5 CHEV~ GOLD  Covers    89365 VAN          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 1 more variable: predictedClass <fct>

resultsKNN %>%
  xtabs(~predictedClass + BadBuy, .) %>%
  confusionMatrix(positive = "1")

## Confusion Matrix and Statistics
##
##               BadBuy
## predictedClass    0    1
##              0 1249  774
##              1  533  965
##
##                Accuracy : 0.6288
##                  95% CI : (0.6126, 0.6448)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2562
##
##  Mcnemar's Test P-Value : 3.168e-11
##
##             Sensitivity : 0.5549
##             Specificity : 0.7009
##          Pos Pred Value : 0.6442
##          Neg Pred Value : 0.6174
##              Prevalence : 0.4939
```

```
##          Detection Rate : 0.2741
##    Detection Prevalence : 0.4254
##       Balanced Accuracy : 0.6279
##
##        'Positive' Class : 1
##
```

```
# 5.c.i

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

set.seed(123)

fitLasso <- train(BadBuy ~ ., family = 'binomial', data = dfcTrain1, method =
'glmnet', trControl = trainControl(method = 'cv', number = 10), tuneGrid =
expand.grid(alpha = 1, lambda = lambdaValues))

varImp(fitLasso)$importance %>%
  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
```

```
# 5.c.ii

plot(varImp(fitLasso), top = 25)
```



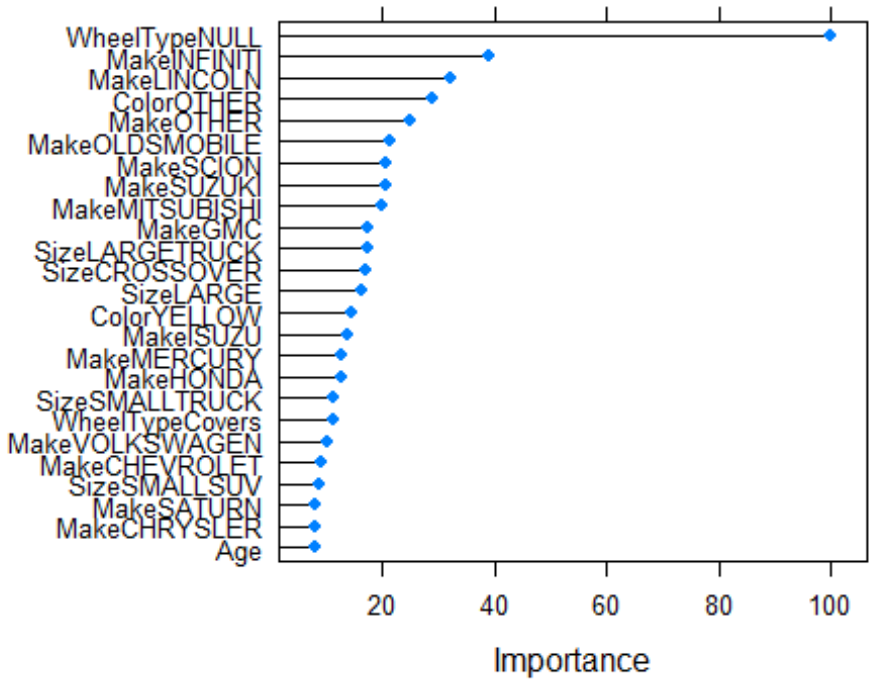

```
# 5.c.iii

fitLasso$bestTune$lambda

## [1] 0.0003053856

# 5.c.iv

resultsLasso <-
  fitLasso %>%
  predict(dfcTest1, type = 'raw') %>%
  bind_cols(dfcTest1, 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
##
```

```
# 5.d

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

set.seed(123)

fitRidge <- train(BadBuy ~ ., family = 'binomial', data = dfcTrain1, method =
'glmnet', trControl = trainControl(method = 'cv', number = 10), tuneGrid =
expand.grid(alpha = 0, lambda = lambdaValues))

varImp(fitRidge)$importance %>%
  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 38.93549%
##  3 MakeLINCOLN       32.2 32.15791%
##  4 ColorOTHER        28.8 28.84244%
##  5 MakeOTHER         24.8 24.79870%
##  6 MakeOLDSMOBILE    21.5 21.50572%
##  7 MakeSCION         20.8 20.77209%
##  8 MakeSUZUKI        20.7 20.74687%
##  9 MakeMITSUBISHI    19.9 19.91028%
## 10 MakeGMC           17.5 17.50911%
## # ... with 49 more rows
```

```
plot(varImp(fitRidge), top = 25)
```




```
fitRidge$bestTune$lambda
```

```
## [1] 0.0559081
```

```
resultsRidge <-
  fitRidge %>%
  predict(dfcTest1, type = 'raw') %>%
  bind_cols(dfcTest1, predictedClass = .)

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

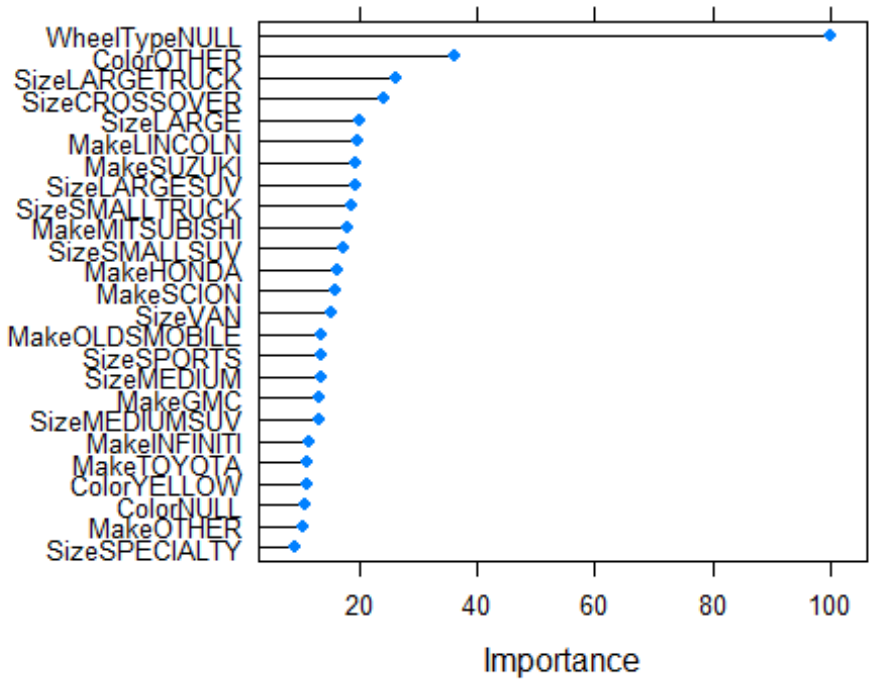
```
## Confusion Matrix and Statistics
##
##                BadBuy
## predictedClass    0    1
##              0 1323  699
##              1  459 1040
##
##                Accuracy : 0.6711
##                  95% CI : (0.6553, 0.6866)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 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
##
```

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

set.seed(123)

fitElastic <- train(BadBuy ~ ., family = 'binomial', data = dfcTrain1,
method='glmnet', trControl = trainControl(method = 'cv', number = 10),
tuneGrid = expand.grid(alpha = 0.5, lambda=lambdaValues))

varImp(fitElastic)$importance %>%
  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        36.1 36%
##  3 SizeLARGETRUCK    26.4 26%
##  4 SizeCROSSOVER     24.1 24%
##  5 SizeLARGE         20.1 20%
##  6 MakeLINCOLN       19.9 20%
##  7 MakeSUZUKI        19.5 19%
##  8 SizeLARGESUV      19.4 19%
##  9 SizeSMALLTRUCK    18.7 19%
## 10 MakeMITSUBISHI    18.1 18%
## # ... with 49 more rows
```

```
plot(varImp(fitElastic), top = 25)
```




```
fitElastic$bestTune$lambda
```

```
## [1] 0.0005857021
```

```
resultsElastic <-
  fitElastic %>%
  predict(dfcTest1, type='raw') %>%
  bind_cols(dfcTest1, predictedClass=.)

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

```
## Confusion Matrix and Statistics
##
##                BadBuy
## predictedClass    0    1
##              0 1339  723
##              1  443 1016
##
##                Accuracy : 0.6688
##                  95% CI : (0.653, 0.6844)
##     No Information Rate : 0.5061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3363
##
```

```
##  Mcnemar's Test P-Value : 3.068e-16
##
##             Sensitivity : 0.5842
##             Specificity : 0.7514
##          Pos Pred Value : 0.6964
##          Neg Pred Value : 0.6494
##              Prevalence : 0.4939
##          Detection Rate : 0.2886
##    Detection Prevalence : 0.4144
##       Balanced Accuracy : 0.6678
##
##        'Positive' Class : 1
##
```

```
# 5.e.i
```

```
set.seed(123)
```

```
fitQDA <- train(BadBuy ~ ., family = 'binomial', data = dfcTrain1, method =
'qda', trControl = trainControl(method = 'cv', number = 10))
```

```
## 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 =
trainInfo, :
## There were missing values in resampled performance measures.
```

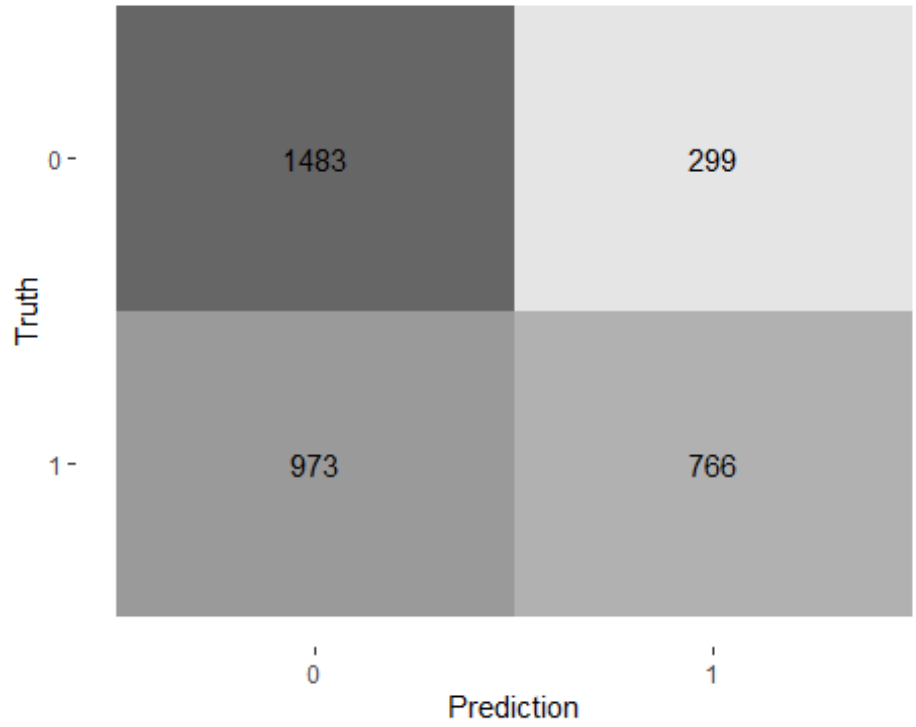
```
resultsQDA <-   fitQDA %>%
    predict(dfcTest1, type = "raw") %>%
    bind_cols(dfcTest1, predictedClass = .)
```

```
resultsQDA
```

```
## # A tibble: 3,521 x 11
##    Auction   Age Make  Color WheelType   Odo Size  MMRAauction MMRAretail
BadBuy
##    <fct>   <dbl> <fct> <fct> <fct>     <dbl> <fct>       <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 1 more variable: predictedClass <fct>
```

```
resultsQDA %>%
  conf_mat(truth = BadBuy, estimate = predictedClass) %>%
  autoplot(type = "heatmap")
```




```
resultsQDA %>%
  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
##
```

```
# 5.f

options(yardstick.event_first = FALSE)

# install.packages("cowplot")
library(cowplot)
```

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

```
##
## ************************************************************
```

```
## Note: As of version 1.0.0, cowplot does not change the
```

```
##   default ggplot2 theme anymore. To recover the previous
```

```
##   behavior, execute:
##   theme_set(theme_cowplot())
```

```
## ************************************************************
```

```
fitLPMCopy <- resultsTest1 %>%
  mutate(model = "m1")

fitLGM1Copy <- resultsCaret1 %>%
  mutate(model = "m2")

fitLGM2Copy <- resultsCaret2 %>%
  mutate(model = "m3")

fitLDACopy <- resultsLDA %>%
  mutate(model = "m4")
```

```r
fitLassoCopy <- resultsLasso %>%
  mutate(model = "m5")

fitRidgeCopy <- resultsRidge %>%
 mutate(model = "m6")

fitElasticCopy <- resultsElastic %>%
  mutate(model = "m7")

fitQDACopy <- resultsQDA %>%
  mutate(model = "m8")

fitKNNCopy <- resultsKNN %>%
  mutate(model = "m9")

glmOutAll <- bind_rows(fitLPMCopy, fitLGM1Copy,fitLGM2Copy, fitLDACopy,
fitLassoCopy, fitRidgeCopy, fitElasticCopy, fitQDACopy, fitKNNCopy)
```

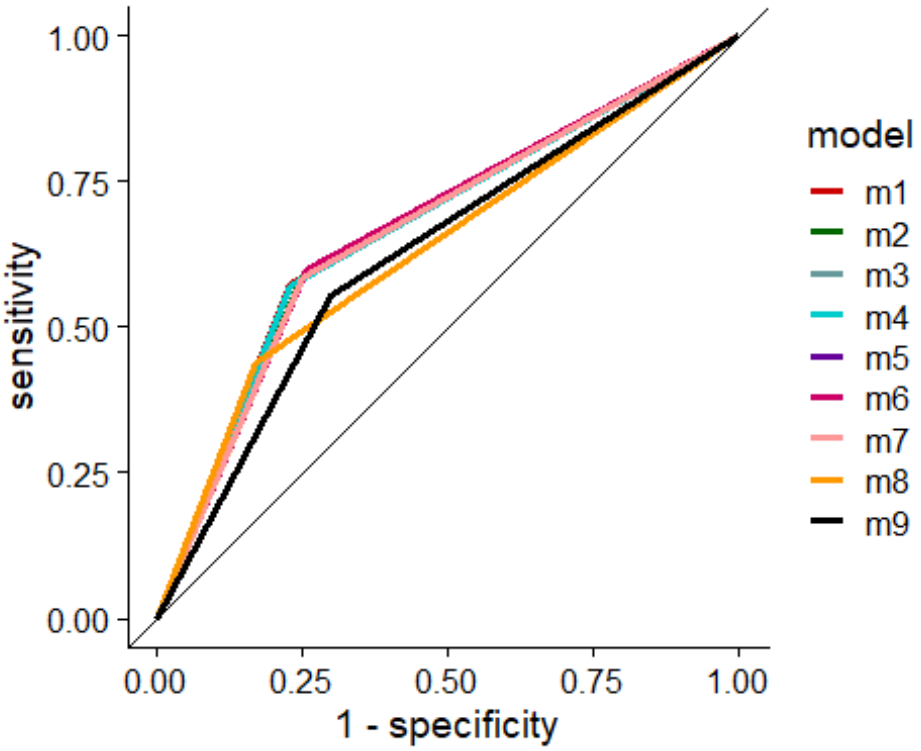
```
## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
```

```
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
coercing
## into character vector
```

```
glmOutAll$predictedClass <- as.numeric(glmOutAll$predictedClass)

glmOutAll %>%
  group_by(model) %>%
  roc_curve(truth = BadBuy, predictedClass) %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity, color = model)) +
  geom_line(size = 1.1) +
  geom_abline(slope = 1, intercept = 0, size = 0.4) +
  scale_color_manual(values = c("#CC0000", "#006600", "#669999", "#00CCCC",
                                "#660099", "#CC0066", "#FF9999", "#FF9900",
                                "black", "black", "black", "black", "black")) +
  coord_fixed() +
  theme_cowplot()
```




```
glmOutAll %>%
  group_by(model) %>%
  roc_auc(truth = BadBuy, predictedClass)
```

```
## # A tibble: 9 x 4
##   model .metric .estimator .estimate
##   <chr> <chr>   <chr>          <dbl>
## 1 m1    roc_auc binary         0.672
## 2 m2    roc_auc binary         0.669
## 3 m3    roc_auc binary         0.669
## 4 m4    roc_auc binary         0.671
## 5 m5    roc_auc binary         0.668
## 6 m6    roc_auc binary         0.670
## 7 m7    roc_auc binary         0.668
## 8 m8    roc_auc binary         0.636
## 9 m9    roc_auc binary         0.628
```

```
# Bonus question

# install.packages("grplasso")
library(grplasso)

set.seed(123)

dfTrainGroup <-
  dfcTrain1 %>%
  mutate(BadBuy = as.numeric(BadBuy)) %>%
```

```r
  mutate(BadBuy = ifelse(BadBuy == 2, 1, 0))

fitGroupedLasso1 <- grplasso(BadBuy ~ ., data = dfTrainGroup, model =
LogReg(), lambda = 50)
```

```
## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
```

```
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
```

```
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Lambda: 50  nr.var: 42

fitGroupedLasso1$coefficients

##                             50
## (Intercept)      -1.732604e+00
## AuctionMANHEIM    0.000000e+00
## AuctionOTHER      0.000000e+00
## Age               2.272898e-01
## MakeCHEVROLET    -2.793690e-02
## MakeCHRYSLER      1.200698e-02
## MakeDODGE        -8.985755e-03
## MakeFORD          6.286615e-03
## MakeGMC          -5.337319e-02
## MakeHONDA        -4.819393e-02
## MakeHYUNDAI      -1.549876e-02
## MakeINFINITI      6.727813e-02
## MakeISUZU        -3.584038e-02
## MakeJEEP         -9.067129e-04
## MakeKIA          -1.930114e-02
## MakeLINCOLN       6.482665e-02
## MakeMAZDA         1.932910e-03
## MakeMERCURY       2.500740e-02
## MakeMITSUBISHI   -5.797448e-02
## MakeNISSAN       -7.640744e-05
## MakeOLDSMOBILE    4.177004e-02
## MakePONTIAC      -1.669154e-02
## MakeSATURN        1.442574e-02
## MakeSCION        -6.077640e-02
## MakeSUZUKI        5.205267e-02
## MakeTOYOTA       -3.277233e-02
## MakeVOLKSWAGEN    2.505957e-02
## MakeOTHER         4.418157e-02
```

```
## ColorBLACK         0.000000e+00
## ColorBLUE          0.000000e+00
## ColorBROWN         0.000000e+00
## ColorGOLD          0.000000e+00
## ColorGREEN         0.000000e+00
## ColorGREY          0.000000e+00
## ColorMAROON        0.000000e+00
## ColorORANGE        0.000000e+00
## ColorOTHER         0.000000e+00
## ColorPURPLE        0.000000e+00
## ColorRED           0.000000e+00
## ColorSILVER        0.000000e+00
## ColorWHITE         0.000000e+00
## ColorYELLOW        0.000000e+00
## ColorNULL          0.000000e+00
## WheelTypeCovers   -1.147940e-01
## WheelTypeNULL      2.715202e+00
## WheelTypeSpecial   1.092006e-02
## Odo                1.012407e-05
## SizeCROSSOVER     -1.973973e-01
## SizeLARGE         -2.388554e-01
## SizeLARGESUV      -1.600920e-01
## SizeLARGETRUCK    -2.665217e-01
## SizeMEDIUM        -1.229139e-01
## SizeMEDIUMSUV     -1.267654e-01
## SizeSMALLSUV      -1.507574e-01
## SizeSMALLTRUCK    -1.831945e-01
## SizeSPECIALTY     -3.367548e-02
## SizeSPORTS        -1.303393e-01
## SizeVAN           -1.484230e-01
## MMRAauction       -1.790527e-05
## MMRAretail         0.000000e+00
```

```
fitGroupedLasso2 <- grplasso(BadBuy ~ ., data = dfTrainGroup, model =
LogReg(), lambda = 100)
```

```
## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
```

```
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in cond/sqrt(cond.norm2): Recycling array of length 1 in vector-
array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Lambda: 100  nr.var: 7
```

```
fitGroupedLasso2$coefficients
```

```
##                              100
## (Intercept)      -1.571244e+00
## AuctionMANHEIM    0.000000e+00
## AuctionOTHER      0.000000e+00
## Age               2.103677e-01
## MakeCHEVROLET     0.000000e+00
## MakeCHRYSLER      0.000000e+00
## MakeDODGE         0.000000e+00
```

```
## MakeFORD          0.000000e+00
## MakeGMC           0.000000e+00
## MakeHONDA         0.000000e+00
## MakeHYUNDAI       0.000000e+00
## MakeINFINITI      0.000000e+00
## MakeISUZU         0.000000e+00
## MakeJEEP          0.000000e+00
## MakeKIA           0.000000e+00
## MakeLINCOLN       0.000000e+00
## MakeMAZDA         0.000000e+00
## MakeMERCURY       0.000000e+00
## MakeMITSUBISHI    0.000000e+00
## MakeNISSAN        0.000000e+00
## MakeOLDSMOBILE    0.000000e+00
## MakePONTIAC       0.000000e+00
## MakeSATURN        0.000000e+00
## MakeSCION         0.000000e+00
## MakeSUZUKI        0.000000e+00
## MakeTOYOTA        0.000000e+00
## MakeVOLKSWAGEN    0.000000e+00
## MakeOTHER         0.000000e+00
## ColorBLACK        0.000000e+00
## ColorBLUE         0.000000e+00
## ColorBROWN        0.000000e+00
## ColorGOLD         0.000000e+00
## ColorGREEN        0.000000e+00
## ColorGREY         0.000000e+00
## ColorMAROON       0.000000e+00
## ColorORANGE       0.000000e+00
## ColorOTHER        0.000000e+00
## ColorPURPLE       0.000000e+00
## ColorRED          0.000000e+00
## ColorSILVER       0.000000e+00
## ColorWHITE        0.000000e+00
## ColorYELLOW       0.000000e+00
## ColorNULL         0.000000e+00
## WheelTypeCovers  -1.096563e-01
## WheelTypeNULL     2.285604e+00
## WheelTypeSpecial  2.736726e-02
## Odo               7.164414e-06
## SizeCROSSOVER     0.000000e+00
## SizeLARGE         0.000000e+00
## SizeLARGESUV      0.000000e+00
## SizeLARGETRUCK    0.000000e+00
## SizeMEDIUM        0.000000e+00
## SizeMEDIUMSUV     0.000000e+00
## SizeSMALLSUV      0.000000e+00
## SizeSMALLTRUCK    0.000000e+00
## SizeSPECIALTY     0.000000e+00
## SizeSPORTS        0.000000e+00
```

```
## SizeVAN           0.000000e+00
## MMRAauction      -1.587228e-05
## MMRAretail        0.000000e+00
```

```
fitLasso1 <- train(BadBuy ~ ., family = 'binomial', data = dfcTrain1, method
= 'glmnet', trControl = trainControl(method = 'cv', number = 10), tuneGrid =
expand.grid(alpha = 1, lambda = 0.01))

coefficients(fitLasso1)
```

```
## NULL
```