Purchase Predction Model Selection

Due February 20, 2022

Problem Overview

The goal of this homework is hands-on practice with linear regression, logistic regression, classification, and model selection. You will:

1. Conduct basic exploratory analysis of a data set
2. Develop linear and logistic regression models
3. Interpret your models
4. Partition your dataset and evaluate your models in terms of classification performance

The Assignment

The data in the accompanying file "car_sales.csv" (posted on Canvas) contains data from 10,062 car auctions. Auto dealers purchase used cars at auctions with the plan to sell them to consumers, but sometimes these auctioned vehicles can have severe issues that prevent them from being resold. The data contains information about each auctioned vehicle (for instance: the make, color, and age, among other variables). A full data dictionary is given in carvana_data_dictionary.txt (we have included only a subset of the variables in their data set). See http://www.kaggle.com/c/DontGetKicked (http://www.kaggle.com/c/DontGetKicked) for documentation on the problem.

Your task is to develop models to predict the target variable "IsBadBuy", which labels whether a car purchased at auction was a "bad buy" or not. The intended use case for this model is to help an auto dealership decide whether or not to purchase an individual vehicle.

```
library(tidyverse)
car = read_csv("car_data.csv")  #read the car_data dataset in R
names(car)                       #variables used in dataset
```

```
##  [1] "Auction"                           "VehicleAge"
##  [3] "Make"                              "Color"
##  [5] "WheelType"                         "VehOdo"
##  [7] "Size"                              "MMRAcquisitionAuctionAveragePrice"
##  [9] "MMRAcquisitionRetailAveragePrice"  "IsBadBuy"
```

0: Example answer

What is the mean of VehicleAge variable?

ANSWER: The mean age of a vehicle in this dataset is 4.504969.

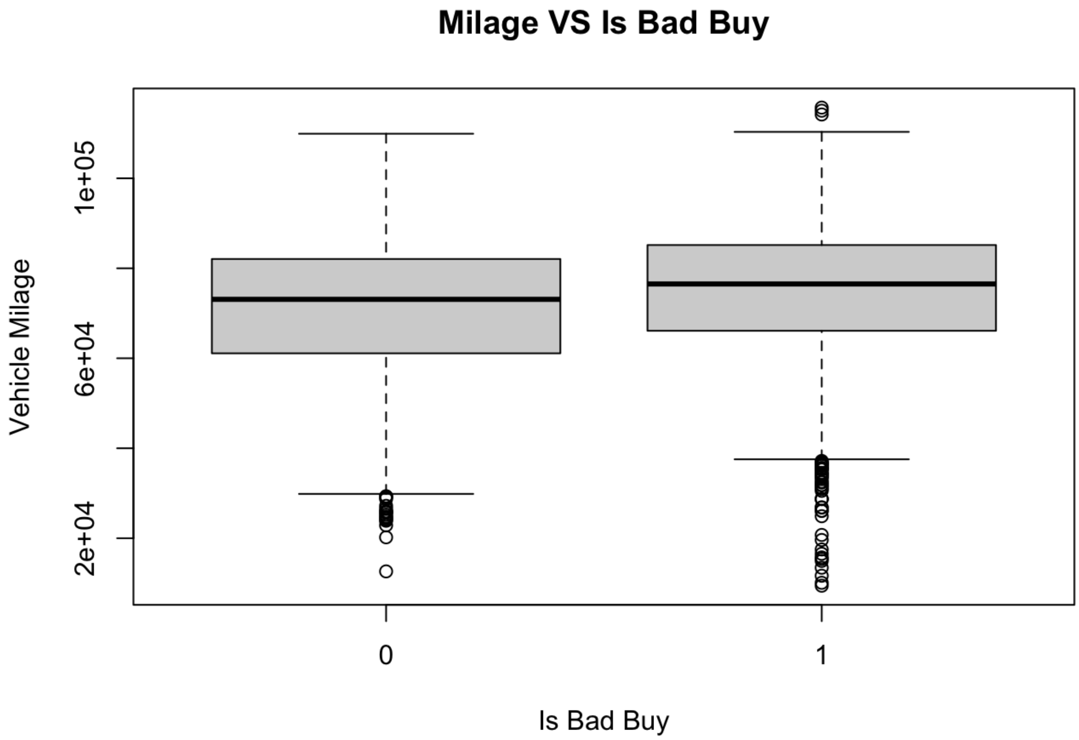
```
age_mean <- car %>%
  summarise(mean_age = mean(VehicleAge))
```

1: EDA and Data Cleaning

a. Construct and report boxplots of VehOdo and VehAge (broken up by values of IsBadBuy). Does it appear there is a relationship between either of these numerical variables and IsBadBuy?

ANSWER TO QUESTION 1a HERE: There is no specific relationship between these 2 numeric variables and and whether a car is a bad buy or not. The graph only points out that the median Vehodo and the median VehAge is higher when IsBadBuy = 1, which may indicate that older cars traveled more and they could be potentially bad buys. However, since we just compared medians, we cannot establish strong relationship between both sides.

```
#Boxplots of VehOdo VS IsBadGuy
VehOdo_boxplot=boxplot(car$VehOdo~car$IsBadBuy,main="Milage VS Is Bad Buy",
xlab="Is Bad Buy", ylab="Vehicle Milage")
```



b. Construct a two-way table of IsBadBuy by Make. Does it appear that any vehicle makes are particularly problematic?

ANSWER TO QUESTION 1b HERE: All the Lexus, Plymouth cars in this data set have only bad buys, which means 100% bad buys. Other very problematic makes include Infiniti (8 out of 10 bad buys), OLDSMOBILE (31 out of 43), Subaru (3 out of 4), etc.

```
table(car$IsBadBuy,car$Make)
```

```
##
##     ACURA BUICK CADILLAC CHEVROLET CHRYSLER DODGE FORD  GMC HONDA HYUNDAI
##   0     4    43        1      1191      604   911  774   42    41     115
##   1     5    60        2       930      613   742  990   43    36     124
##
##     INFINITI ISUZU JEEP  KIA LEXUS LINCOLN MAZDA MERCURY MINI MITSUBISHI NISSAN
##   0        2    10  108  203     0       7    73      61    3         81    138
##   1        8     5  134  169     8      16    95      91    5         65    191
##
##     OLDSMOBILE PLYMOUTH PONTIAC SATURN SCION SUBARU SUZUKI TOYOTA VOLKSWAGEN
##   0         12        0     317    132    11      1     84     78          8
##   1         31        1     280    165     7      3    110     65         10
##
##     VOLVO
##   0     3
##   1     0
```

c. Construct the following new variables :

- MPYind = 1 when the miles/year is above the median and 0 otherwise
- VehType which has the following values:
  - SUV when Size is LARGE SUV, MEDIUM SUV, or SMALL SUV
  - Truck when Size is Large Truck, Medium Truck, or Small Truck

- Regular when Size is VAN, CROSSOVER, LARGE, or MEDIUM
- Small when size is COMPACT, SPECIALTY, or SPORT Hint: there are lots of ways to do this one, but case_when might be a useful function that's part of the tidyverse
- Price0 which is 1 when either the MMRAcquisitionRetailAveragePrice or MMRAcquisitionAuctionAveragePrice are equal to 0, and 0 otherwise

Also, modify these two existing variables:

- The value of Make should be replaced with "other_make" when there are fewer than 20 cars with that make
- The value of Color should be replaced with "other_color" when there are fewer than 20 cars with that color

ANSWER TO QUESTION 1c HERE:

```
car_clean = car %>%
  mutate(MilesPerYear = VehOdo/VehicleAge,
         medianPerYear=median(MilesPerYear))%>%
  mutate(MPYind = ifelse(MilesPerYear > median(MilesPerYear), 1, 0), # When the miles per year is above t
he median, the variable MPYind should be marked as 1, otherwise 0.
         VehType = case_when(Size == "LARGE SUV" | Size == "MEDIUM SUV" | Size == "SMALL SUV" ~ "SUV",
                             Size == "LARGE TRUCK" | Size == "MEDIUM TRUCK" | Size == "SMALL TRUCK" ~ "Tr
uck",
                             Size == "VAN" | Size == "CROSSOVER" | Size == "LARGE" | Size == "MEDIUM" ~
"Regular",
                             Size == "COMPACT" | Size == "SPECIALTY" | Size == "SPORTS" ~ "Small"),
        Price0 = ifelse(MMRAcquisitionRetailAveragePrice == 0 | MMRAcquisitionAuctionAveragePrice == 0,1,
0)) %>%
  group_by(Make) %>%
  mutate(count = n(),
         Make = ifelse(count < 20, "other_make",Make)) %>%
  ungroup() %>%
  group_by(Color) %>%
  mutate(count = n(),
         Color = ifelse(count < 20, "other_color", Color))%>%
  ungroup()
summary(car_clean)
```

```
##    Auction           VehicleAge       Make             Color          
##  Length:10062       Min.   :1.000   Length:10062       Length:10062      
##  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             VehOdo           Size          
##  Length:10062       Min.   :  9446   Length:10062      
##  Class :character   1st Qu.: 63489   Class :character  
##  Mode  :character   Median : 74942   Mode  :character  
##                     Mean   : 72904                     
##                     3rd Qu.: 83662                     
##                     Max.   :115717                     
##  MMRAcquisitionAuctionAveragePrice MMRAcquisitionRetailAveragePrice
##  Min.   :    0                     Min.   :    0                   
##  1st Qu.: 3876                     1st Qu.: 5872                   
##  Median : 5587                     Median : 8052                   
##  Mean   : 5812                     Mean   : 8171                   
##  3rd Qu.: 7450                     3rd Qu.:10315                   
##  Max.   :35722                     Max.   :39080                   
##     IsBadBuy       MilesPerYear   medianPerYear       MPYind   
##  Min.   :0.0000   Min.   : 2137   Min.   :16765   Min.   :0.0  
##  1st Qu.:0.0000   1st Qu.:12987   1st Qu.:16765   1st Qu.:0.0  
##  Median :0.0000   Median :16765   Median :16765   Median :0.5  
##  Mean   :0.4973   Mean   :19035   Mean   :16765   Mean   :0.5  
##  3rd Qu.:1.0000   3rd Qu.:22212   3rd Qu.:16765   3rd Qu.:1.0  
##  Max.   :1.0000   Max.   :92996   Max.   :16765   Max.   :1.0  
##    VehType              Price0            count     
##  Length:10062       Min.   :0.00000   Min.   :   2  
##  Class :character   1st Qu.:0.00000   1st Qu.: 881  
##  Mode  :character   Median :0.00000   Median :1386  
##                     Mean   :0.01222   Mean   :1275  
##                     3rd Qu.:0.00000   3rd Qu.:1653  
##                     Max.   :1.00000   Max.   :2081
```

```
table(car_clean$Make)
```

```
## 
##      BUICK  CHEVROLET   CHRYSLER      DODGE       FORD        GMC      HONDA 
##        103       2121       1217       1653       1764         85         77 
##    HYUNDAI       JEEP        KIA    LINCOLN      MAZDA    MERCURY MITSUBISHI 
##        239        242        372         23        168        152        146 
##     NISSAN OLDSMOBILE other_make    PONTIAC     SATURN     SUZUKI     TOYOTA 
##        329         43         97        597        297        194        143
```

```
table(car_clean$Color)
```

```
## 
## 'NOT AVAIL'       BEIGE       BLACK        BLUE       BROWN        GOLD 
##          26         237        1013        1386          65         767 
##       GREEN        GREY      MAROON      ORANGE       OTHER other_color 
##         442        1054         281          43          37           2 
##      PURPLE         RED      SILVER       WHITE      YELLOW 
##          57         881        2081        1653          37
```

d. The rows where MMRAcquisitionRetailAveragePrice or MMRAcquisitionAuctionAveragePrice are equal to 0 are suspicious - it seems like those values might not be correct. Replace the two prices with the average grouped by vehicle make. Be sure to remove the 0's from the average calculation! Hint: this one is a little tricky. Consider using the special character NA to replace the 0's.

ANSWER TO QUESTION 1d HERE:

```
car_clean = car_clean %>%
        #Replace 0 with mean value
  mutate(MMRAcquisitionRetailAveragePrice = ifelse(MMRAcquisitionRetailAveragePrice == 0 ,mean(MMRAcquisi
tionRetailAveragePrice),MMRAcquisitionRetailAveragePrice),
         MMRAcquisitionAuctionAveragePrice = ifelse(MMRAcquisitionAuctionAveragePrice == 0,  mean(MMRAcqu
isitionAuctionAveragePrice),MMRAcquisitionAuctionAveragePrice),
         #Replace NA with mean value
         MMRAcquisitionRetailAveragePrice = ifelse(is.na(MMRAcquisitionRetailAveragePrice) ,mean(MMRAcqui
sitionRetailAveragePrice),MMRAcquisitionRetailAveragePrice),
         MMRAcquisitionAuctionAveragePrice = ifelse(is.na(MMRAcquisitionAuctionAveragePrice),  mean(MMRAc
quisitionAuctionAveragePrice),MMRAcquisitionAuctionAveragePrice))
```

```
summary(car_clean$MMRAcquisitionAuctionAveragePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     884    3956    5684    5883    7450   35722
```

```
summary(car_clean$MMRAcquisitionRetailAveragePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1455    5981    8150    8271   10315   39080
```

2: Linear Regression

a. Train a linear regression to predict IsBadBuy using the variables listed below. Report the R^2.

- Auction
- VehicleAge
- Make
- Color
- WheelType
- VehOdo
- MPYind
- VehType
- MMRAcquisitionAuctionAveragePrice
- MMRAcquisitionRetailAveragePrice

ANSWER TO QUESTION 2a HERE: Multiple R-squared: 0.1894, Adjusted R-squared: 0.1854

```
car_clean = car_clean %>%
  mutate(Auction = as.factor(Auction),
         Make = as.factor(Make),
         Color = as.factor(Color),
         WheelType = as.factor(WheelType),
         Size=as.factor(Size),
         VehType = as.factor(VehType),
         IsBadBuy = as.numeric(IsBadBuy)) #make IsBadBuy an numeric variable so it could be the dependent
variable in linear regression
```

```
linreg1 = lm(data = car_clean, IsBadBuy ~ Auction + VehicleAge + Make + Color + WheelType + VehOdo + MPYi
nd + VehType + MMRAcquisitionAuctionAveragePrice + MMRAcquisitionRetailAveragePrice)
summary(linreg1)
```

```
##
## Call:
## lm(formula = IsBadBuy ~ Auction + VehicleAge + Make + Color +
##     WheelType + VehOdo + MPYind + VehType + MMRAcquisitionAuctionAveragePrice +
##     MMRAcquisitionRetailAveragePrice, data = car_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2697 -0.3950 -0.1620  0.4688  0.9560
##
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       -1.270e-02  1.091e-01  -0.116 0.907300
## AuctionMANHEIM                     4.284e-02  1.201e-02   3.568 0.000361 ***
## AuctionOTHER                       7.337e-03  1.367e-02   0.537 0.591418
## VehicleAge                         5.026e-02  5.517e-03   9.111  < 2e-16 ***
## MakeCHEVROLET                     -3.849e-02  4.595e-02  -0.837 0.402339
## MakeCHRYSLER                       4.944e-02  4.685e-02   1.055 0.291318
## MakeDODGE                          4.746e-03  4.643e-02   0.102 0.918577
## MakeFORD                           2.673e-02  4.617e-02   0.579 0.562673
## MakeGMC                           -3.755e-02  6.694e-02  -0.561 0.574852
## MakeHONDA                         -1.229e-01  6.842e-02  -1.796 0.072494 .
## MakeHYUNDAI                        8.449e-03  5.367e-02   0.157 0.874912
## MakeJEEP                           9.915e-03  5.437e-02   0.182 0.855300
## MakeKIA                            2.576e-02  5.110e-02   0.504 0.614167
## MakeLINCOLN                        6.727e-02  1.045e-01   0.644 0.519713
## MakeMAZDA                          3.541e-02  5.680e-02   0.623 0.533010
## MakeMERCURY                        4.231e-02  5.779e-02   0.732 0.464084
## MakeMITSUBISHI                    -1.113e-01  5.850e-02  -1.903 0.057054 .
## MakeNISSAN                         3.383e-02  5.123e-02   0.660 0.509117
## MakeOLDSMOBILE                     8.039e-02  8.224e-02   0.978 0.328319
## Makeother_make                     4.915e-02  6.549e-02   0.751 0.452959
## MakePONTIAC                       -1.001e-02  4.856e-02  -0.206 0.836728
## MakeSATURN                         3.882e-02  5.202e-02   0.746 0.455535
## MakeSUZUKI                         1.415e-01  5.628e-02   2.514 0.011945 *
## MakeTOYOTA                        -2.325e-02  5.885e-02  -0.395 0.692831
## ColorBEIGE                        -7.511e-03  9.360e-02  -0.080 0.936048
## ColorBLACK                         1.465e-02  9.008e-02   0.163 0.870765
## ColorBLUE                          5.836e-03  8.978e-02   0.065 0.948174
## ColorBROWN                         2.807e-02  1.052e-01   0.267 0.789634
## ColorGOLD                          4.684e-02  9.050e-02   0.518 0.604730
## ColorGREEN                        -7.624e-03  9.153e-02  -0.083 0.933617
## ColorGREY                          7.381e-03  9.007e-02   0.082 0.934694
## ColorMAROON                        8.210e-02  9.296e-02   0.883 0.377167
## ColorORANGE                       -1.621e-02  1.127e-01  -0.144 0.885639
## ColorOTHER                        -1.531e-01  1.157e-01  -1.323 0.185998
## Colorother_color                  -4.843e-01  3.320e-01  -1.459 0.144684
## ColorPURPLE                        5.833e-02  1.073e-01   0.543 0.586809
## ColorRED                           3.190e-02  9.027e-02   0.353 0.723832
## ColorSILVER                        3.332e-02  8.950e-02   0.372 0.709686
## ColorWHITE                         2.865e-02  8.966e-02   0.320 0.749349
## ColorYELLOW                       -8.274e-02  1.163e-01  -0.712 0.476756
## WheelTypeCovers                   -2.524e-02  1.110e-02  -2.275 0.022940 *
## WheelTypeNULL                      5.193e-01  1.508e-02  34.448  < 2e-16 ***
## WheelTypeSpecial                  -1.037e-02  4.584e-02  -0.226 0.820983
## VehOdo                             2.410e-06  3.967e-07   6.075 1.28e-09 ***
## MPYind                            -1.113e-02  1.512e-02  -0.736 0.461643
## VehTypeSmall                       6.806e-02  1.375e-02   4.949 7.58e-07 ***
## VehTypeSUV                         1.237e-02  1.600e-02   0.773 0.439345
## VehTypeTruck                      -2.927e-02  2.205e-02  -1.327 0.184444
## MMRAcquisitionAuctionAveragePrice -2.344e-06  5.396e-06  -0.434 0.663942
```

```
## MMRAcquisitionRetailAveragePrice   3.210e-07  3.592e-06   0.089 0.928804
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4513 on 10012 degrees of freedom
## Multiple R-squared:  0.1894, Adjusted R-squared:  0.1854
## F-statistic: 47.75 on 49 and 10012 DF,  p-value: < 2.2e-16
```

b. What is the predicted value of IsBadBuy for a MANHEIM Auction, 4-year-old Compact Blue Volvo with 32000 miles, WheelType = Special, an MMR Auction Price of $8000, and an MMR Retail Price of $12000? What would be your predicted classification for the car, using a cutoff of 0.5?

ANSWER TO QUESTION 2b HERE: The predicted value for IsBadBuy for the test data is 0.4060868. Having a cutoff being 0.5, the car would be classified as 0, which means the car is a good buy.

```
test1 = data.frame(Auction = "MANHEIM",VehicleAge = 4,Make = "other_make",Color = "BLUE",WheelType = "Spe
cial",VehOdo = 32000,MPYind = 0,VehType = "Small",MMRAcquisitionAuctionAveragePrice = 8000,MMRAcquisition
RetailAveragePrice = 12000) #fill in the values mentioned above to the test1 (since the miles per years i
n below the median level, which is 16765, the MPYind=0)
IsBadBuylinreg = predict(linreg1,newdata=test1) #using the values filled to predict if the car is a good
 buy
IsBadBuylinreg
```

```
##         1
## 0.4060868
```

c. Do you have any reservations about this predicted IsBadBuy? That is, would you feel sufficiently comfortable with this prediction in order to take action based on it? Why or why not?

ANSWER TO QUESTION 2c HERE: It's not sufficient to predict categorical variables using a linear regression since the values of predicted model could be out of the range 0 to 1. So 0.4060868 is not a sufficient prediction.

3: Logistic Regression

a. Train a Logistic Regression model using the same variables as in 2a. Report the AIC of your model.

ANSWER TO QUESTION 3a HERE: The AIC value for the logistic model is 11778.

```
log_model1= glm(data=car_clean, IsBadBuy ~ Auction + VehicleAge + Make + Color + WheelType + VehOdo + MPY
ind + VehType +MMRAcquisitionAuctionAveragePrice + MMRAcquisitionRetailAveragePrice,family="binomial")

summary(log_model1)
```

```
##
## Call:
## glm(formula = IsBadBuy ~ Auction + VehicleAge + Make + Color +
##     WheelType + VehOdo + MPYind + VehType + MMRAcquisitionAuctionAveragePrice +
##     MMRAcquisitionRetailAveragePrice, family = "binomial", data = car_clean)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1104  -0.9833  -0.5302   1.0960   2.1348
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -2.705e+00  6.476e-01  -4.177 2.95e-05 ***
## AuctionMANHEIM                    1.914e-01  5.978e-02   3.202  0.00137 **
## AuctionOTHER                      2.404e-02  7.198e-02   0.334  0.73839
## VehicleAge                        2.597e-01  2.778e-02   9.348  < 2e-16 ***
## MakeCHEVROLET                    -1.983e-01  2.295e-01  -0.864  0.38752
## MakeCHRYSLER                      2.174e-01  2.340e-01   0.929  0.35300
## MakeDODGE                        -2.782e-03  2.317e-01  -0.012  0.99042
## MakeFORD                          1.173e-01  2.305e-01   0.509  0.61081
## MakeGMC                          -2.128e-01  3.253e-01  -0.654  0.51292
## MakeHONDA                        -6.113e-01  3.515e-01  -1.739  0.08200 .
## MakeHYUNDAI                       2.965e-02  2.675e-01   0.111  0.91173
## MakeJEEP                          2.930e-02  2.699e-01   0.109  0.91354
## MakeKIA                           1.188e-01  2.552e-01   0.465  0.64172
## MakeLINCOLN                       2.665e-01  5.259e-01   0.507  0.61229
## MakeMAZDA                         1.441e-01  2.820e-01   0.511  0.60929
## MakeMERCURY                       1.866e-01  2.857e-01   0.653  0.51361
## MakeMITSUBISHI                   -5.976e-01  2.957e-01  -2.021  0.04331 *
## MakeNISSAN                        1.507e-01  2.548e-01   0.591  0.55419
## MakeOLDSMOBILE                    3.832e-01  4.222e-01   0.908  0.36408
## Makeother_make                    2.403e-01  3.247e-01   0.740  0.45927
## MakePONTIAC                      -5.656e-02  2.422e-01  -0.234  0.81532
## MakeSATURN                        1.847e-01  2.586e-01   0.714  0.47517
## MakeSUZUKI                        6.932e-01  2.804e-01   2.472  0.01344 *
## MakeTOYOTA                       -1.725e-01  2.923e-01  -0.590  0.55496
## ColorBEIGE                        1.988e-02  5.848e-01   0.034  0.97288
## ColorBLACK                        1.585e-01  5.694e-01   0.278  0.78075
## ColorBLUE                         1.048e-01  5.681e-01   0.184  0.85371
## ColorBROWN                        2.196e-01  6.241e-01   0.352  0.72492
## ColorGOLD                         3.155e-01  5.709e-01   0.553  0.58052
## ColorGREEN                        5.893e-02  5.747e-01   0.103  0.91832
## ColorGREY                         1.214e-01  5.693e-01   0.213  0.83120
## ColorMAROON                       4.979e-01  5.802e-01   0.858  0.39087
## ColorORANGE                       7.697e-03  6.710e-01   0.011  0.99085
## ColorOTHER                       -1.165e+00  7.317e-01  -1.592  0.11144
## Colorother_color                 -3.246e+00  1.597e+00  -2.032  0.04211 *
## ColorPURPLE                       4.656e-01  6.435e-01   0.724  0.46936
## ColorRED                          2.377e-01  5.701e-01   0.417  0.67665
## ColorSILVER                       2.466e-01  5.671e-01   0.435  0.66362
## ColorWHITE                        2.237e-01  5.677e-01   0.394  0.69358
## ColorYELLOW                      -3.500e-01  6.720e-01  -0.521  0.60242
## WheelTypeCovers                  -6.587e-02  5.278e-02  -1.248  0.21204
## WheelTypeNULL                     3.469e+00  1.368e-01  25.364  < 2e-16 ***
## WheelTypeSpecial                 -5.133e-02  2.108e-01  -0.244  0.80761
## VehOdo                            1.257e-05  1.977e-06   6.358 2.05e-10 ***
## MPYind                           -4.337e-02  7.386e-02  -0.587  0.55704
## VehTypeSmall                      3.419e-01  6.807e-02   5.023 5.10e-07 ***
## VehTypeSUV                        5.561e-02  7.840e-02   0.709  0.47815
## VehTypeTruck                     -1.436e-01  1.069e-01  -1.344  0.17895
## MMRAcquisitionAuctionAveragePrice -2.731e-06  2.685e-05  -0.102  0.91897
```

```
## MMRAcquisitionRetailAveragePrice  -9.533e-07  1.773e-05  -0.054  0.95712
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13949  on 10061  degrees of freedom
## Residual deviance: 11678  on 10012  degrees of freedom
## AIC: 11778
##
## Number of Fisher Scoring iterations: 5
```

b. What is the coefficient for VehicleAge? Provide a precise (numerical) interpretation of the coefficient.

ANSWER TO QUESTION 3b HERE: The coefficient for VehAge is 0.2597 which means that with everything else holding constant, generally if the VehAge increases by 1 unit, the odds that Vehicle becoming a bad buy increases by a factor e^(0.2597) and the corresponding probability also increases.

c. What is the coefficient for VehType = Small? Provide a precise (numerical) interpretation of this coefficient.

ANSWER TO QUESTION 3c HERE: The coefficient for VehType = Small is 0.3419, which means that with everything else holding constant, generally if the VehAge increases by 1 unit, the odds that Vehicle becoming a bad buy would increase by a factor e^(0.3419)

d. Compute the predicted probability that the same car as in #2b is a bad buy. Hint: you should use the predict function, but you need to specify type = “response” when predicting probabilities from logistic regression (otherwise, it will predict the value of logit). For example: predict(mymodel, newdata = mydata, type = “response”).

ANSWER TO QUESTION 3d HERE: The predicted value for the test data is 0.3845428

```
IsBadBuylogreg = predict(log_model1, newdata=test1, type = "response")
print(IsBadBuylogreg)
```

```
##         1
## 0.3845428
```

e. If you were to pick one model to use for the purposes of inference (explaining the relationship between the features and the target variable) which would it be, and why?

ANSWER TO QUESTION 3e HERE: I would prefer to pick the logistic model to use for the purposes of inference since for the logistic model, the values can be within the range of 0 and 1, which is more accurate to used in categorical prediction.

4: Classification and Evaluation

a. Split the data into 70% training and 30% validation sets, retrain the linear and logistic regression models using the training data only, and report the resulting R^2 and AIC, respectively.

ANSWER TO QUESTION 4a HERE: For linear model, the R2 is 0.1931 and adjusted R2 is 0.1874 . For logistic model, the AIC value for is 8242.9

```
set.seed(1)

train_insts = sample(nrow(car_clean), .7*nrow(car_clean)) #Split the data into 70% training and 30% valid
ation sets

data_train = car_clean[train_insts,] #assign the 70% data into training data set
data_valid = car_clean[-train_insts,] #assign the rest into validation data set

lm_model2 = lm(data = data_train, IsBadBuy ~ factor(Auction) + VehicleAge + factor(Make) + factor(Color)
 + factor(WheelType) + VehOdo + MPYind + factor(VehType) + MMRAcquisitionAuctionAveragePrice + MMRAcquisi
tionRetailAveragePrice)

lm_predscore_data_valid=predict(lm_model2,newdata =data_valid)

log_model2 = glm(data = data_train, IsBadBuy ~ Auction + VehicleAge + Make + Color + WheelType + VehOdo +
MPYind + VehType + MMRAcquisitionAuctionAveragePrice + MMRAcquisitionRetailAveragePrice, family = "binomi
al")

log_predscore_data_valid=predict(log_model2,newdata =data_valid)

summary(lm_model2)
```

```
##
## Call:
## lm(formula = IsBadBuy ~ factor(Auction) + VehicleAge + factor(Make) +
##     factor(Color) + factor(WheelType) + VehOdo + MPYind + factor(VehType) +
##     MMRAcquisitionAuctionAveragePrice + MMRAcquisitionRetailAveragePrice,
##     data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2463 -0.3898 -0.1573  0.4655  0.9509
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -2.413e-02  1.327e-01  -0.182 0.855645
## factor(Auction)MANHEIM            5.347e-02  1.441e-02   3.710 0.000209 ***
## factor(Auction)OTHER              2.038e-02  1.637e-02   1.245 0.213156
## VehicleAge                        4.345e-02  6.588e-03   6.595 4.57e-11 ***
## factor(Make)CHEVROLET            -4.125e-02  5.229e-02  -0.789 0.430139
## factor(Make)CHRYSLER              4.251e-02  5.336e-02   0.797 0.425754
## factor(Make)DODGE                -3.433e-03  5.289e-02  -0.065 0.948256
## factor(Make)FORD                  5.160e-02  5.251e-02   0.983 0.325795
## factor(Make)GMC                  -2.844e-02  7.801e-02  -0.365 0.715449
## factor(Make)HONDA                -1.318e-01  7.722e-02  -1.707 0.087860 .
## factor(Make)HYUNDAI               1.229e-02  6.220e-02   0.198 0.843357
## factor(Make)JEEP                  1.100e-02  6.206e-02   0.177 0.859277
## factor(Make)KIA                   4.131e-02  5.889e-02   0.702 0.483011
## factor(Make)LINCOLN               4.096e-02  1.240e-01   0.330 0.741208
## factor(Make)MAZDA                 2.954e-02  6.688e-02   0.442 0.658676
## factor(Make)MERCURY               5.801e-02  6.723e-02   0.863 0.388259
## factor(Make)MITSUBISHI           -1.561e-01  6.874e-02  -2.271 0.023202 *
## factor(Make)NISSAN                4.603e-02  5.898e-02   0.780 0.435141
## factor(Make)OLDSMOBILE            7.478e-02  9.273e-02   0.806 0.420020
## factor(Make)other_make            7.366e-02  7.459e-02   0.988 0.323396
## factor(Make)PONTIAC               3.010e-03  5.545e-02   0.054 0.956716
## factor(Make)SATURN                5.232e-02  6.060e-02   0.863 0.387967
## factor(Make)SUZUKI                1.289e-01  6.545e-02   1.970 0.048872 *
## factor(Make)TOYOTA               -5.875e-02  6.911e-02  -0.850 0.395357
## factor(Color)BEIGE                3.413e-03  1.158e-01   0.029 0.976496
## factor(Color)BLACK                3.771e-02  1.113e-01   0.339 0.734794
## factor(Color)BLUE                 2.374e-02  1.110e-01   0.214 0.830675
## factor(Color)BROWN                5.513e-02  1.266e-01   0.435 0.663337
## factor(Color)GOLD                 7.312e-02  1.117e-01   0.654 0.512826
## factor(Color)GREEN                5.267e-02  1.131e-01   0.466 0.641385
## factor(Color)GREY                 3.183e-02  1.113e-01   0.286 0.774849
## factor(Color)MAROON               9.704e-02  1.146e-01   0.847 0.397270
## factor(Color)ORANGE               2.203e-02  1.377e-01   0.160 0.872914
## factor(Color)OTHER               -1.409e-01  1.411e-01  -0.998 0.318103
## factor(Color)other_color         -8.392e-01  4.644e-01  -1.807 0.070828 .
## factor(Color)PURPLE               9.109e-02  1.303e-01   0.699 0.484631
## factor(Color)RED                  5.668e-02  1.115e-01   0.508 0.611319
## factor(Color)SILVER               6.701e-02  1.107e-01   0.605 0.544870
## factor(Color)WHITE                4.948e-02  1.108e-01   0.446 0.655308
## factor(Color)YELLOW              -7.250e-02  1.427e-01  -0.508 0.611348
## factor(WheelType)Covers          -2.513e-02  1.328e-02  -1.893 0.058399 .
## factor(WheelType)NULL             5.220e-01  1.798e-02  29.034  < 2e-16 ***
## factor(WheelType)Special          9.338e-03  5.332e-02   0.175 0.860974
## VehOdo                            2.449e-06  4.729e-07   5.180 2.29e-07 ***
## MPYind                           -2.553e-02  1.806e-02  -1.414 0.157513
## factor(VehType)Small              8.111e-02  1.646e-02   4.927 8.53e-07 ***
## factor(VehType)SUV                2.371e-02  1.907e-02   1.243 0.213972
## factor(VehType)Truck             -2.382e-02  2.619e-02  -0.909 0.363127
```

```
## MMRAcquisitionAuctionAveragePrice -7.363e-06  6.458e-06  -1.140 0.254299
## MMRAcquisitionRetailAveragePrice   4.233e-06  4.289e-06   0.987 0.323748
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4507 on 6993 degrees of freedom
## Multiple R-squared:  0.1931, Adjusted R-squared:  0.1874
## F-statistic: 34.15 on 49 and 6993 DF,  p-value: < 2.2e-16
```

```
summary(log_model2)
```

```
##
## Call:
## glm(formula = IsBadBuy ~ Auction + VehicleAge + Make + Color +
##     WheelType + VehOdo + MPYind + VehType + MMRAcquisitionAuctionAveragePrice +
##     MMRAcquisitionRetailAveragePrice, family = "binomial", data = data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0771  -0.9737  -0.5153   1.0901   2.1339
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -2.723e+00  8.227e-01  -3.310 0.000934 ***
## AuctionMANHEIM                     2.473e-01  7.217e-02   3.426 0.000613 ***
## AuctionOTHER                       1.015e-01  8.679e-02   1.169 0.242264
## VehicleAge                         2.288e-01  3.317e-02   6.898 5.28e-12 ***
## MakeCHEVROLET                     -1.985e-01  2.597e-01  -0.764 0.444706
## MakeCHRYSLER                       1.963e-01  2.651e-01   0.740 0.459019
## MakeDODGE                         -3.005e-02  2.626e-01  -0.114 0.908901
## MakeFORD                           2.416e-01  2.607e-01   0.927 0.354088
## MakeGMC                           -1.525e-01  3.770e-01  -0.405 0.685796
## MakeHONDA                         -6.414e-01  3.958e-01  -1.620 0.105150
## MakeHYUNDAI                        5.235e-02  3.087e-01   0.170 0.865344
## MakeJEEP                           4.642e-02  3.072e-01   0.151 0.879880
## MakeKIA                            2.063e-01  2.921e-01   0.706 0.480025
## MakeLINCOLN                        1.610e-01  6.311e-01   0.255 0.798586
## MakeMAZDA                          1.218e-01  3.300e-01   0.369 0.712055
## MakeMERCURY                        2.665e-01  3.318e-01   0.803 0.421988
## MakeMITSUBISHI                    -8.138e-01  3.515e-01  -2.315 0.020610 *
## MakeNISSAN                         2.186e-01  2.923e-01   0.748 0.454615
## MakeOLDSMOBILE                     3.585e-01  4.731e-01   0.758 0.448677
## Makeother_make                     3.941e-01  3.730e-01   1.057 0.290659
## MakePONTIAC                        1.981e-02  2.752e-01   0.072 0.942607
## MakeSATURN                         2.708e-01  2.998e-01   0.903 0.366369
## MakeSUZUKI                         6.310e-01  3.256e-01   1.938 0.052626 .
## MakeTOYOTA                        -3.335e-01  3.430e-01  -0.972 0.330921
## ColorBEIGE                        -1.875e-02  7.592e-01  -0.025 0.980301
## ColorBLACK                         1.903e-01  7.401e-01   0.257 0.797112
## ColorBLUE                          1.146e-01  7.389e-01   0.155 0.876722
## ColorBROWN                         2.617e-01  7.921e-01   0.330 0.741098
## ColorGOLD                          3.713e-01  7.416e-01   0.501 0.616579
## ColorGREEN                         2.794e-01  7.462e-01   0.374 0.708061
## ColorGREY                          1.658e-01  7.401e-01   0.224 0.822690
## ColorMAROON                        4.930e-01  7.518e-01   0.656 0.511965
## ColorORANGE                        1.091e-01  8.513e-01   0.128 0.898008
## ColorOTHER                        -1.165e+00  9.068e-01  -1.285 0.198671
## Colorother_color                  -1.415e+01  1.970e+02  -0.072 0.942744
## ColorPURPLE                        5.492e-01  8.155e-01   0.674 0.500606
## ColorRED                           2.877e-01  7.408e-01   0.388 0.697715
## ColorSILVER                        3.366e-01  7.376e-01   0.456 0.648099
## ColorWHITE                         2.436e-01  7.383e-01   0.330 0.741476
## ColorYELLOW                       -3.873e-01  8.549e-01  -0.453 0.650475
## WheelTypeCovers                   -6.565e-02  6.332e-02  -1.037 0.299841
## WheelTypeNULL                      3.484e+00  1.626e-01  21.422  < 2e-16 ***
## WheelTypeSpecial                   4.481e-02  2.455e-01   0.183 0.855184
## VehOdo                             1.286e-05  2.356e-06   5.458 4.80e-08 ***
## MPYind                            -1.166e-01  8.844e-02  -1.318 0.187518
## VehTypeSmall                       4.136e-01  8.155e-02   5.072 3.95e-07 ***
## VehTypeSUV                         1.039e-01  9.383e-02   1.108 0.268051
## VehTypeTruck                      -1.235e-01  1.279e-01  -0.965 0.334312
## MMRAcquisitionAuctionAveragePrice -2.635e-05  3.214e-05  -0.820 0.412304
```

```
## MMRAcquisitionRetailAveragePrice   1.828e-05  2.124e-05   0.861 0.389376
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9763.5  on 7042  degrees of freedom
## Residual deviance: 8142.9  on 6993  degrees of freedom
## AIC: 8242.9
##
## Number of Fisher Scoring iterations: 10
```

b. Compute the RMSE in the training and validation sets for the linear model (do not do the classifications, just use the predicted score). Which is better, and does this make sense? Why or why not?

ANSWER TO QUESTION 4b HERE: The RMSE for training data set is 0.449139, which is smaller than the one for validation data set (0.4541079). It seems that the linear model fits better for training data.It makes sense since the model is created based on training data.

```
lm_predscore_data_valid=predict(lm_model2,newdata =data_valid)
lm_RMSE_data_valid=sqrt(mean((lm_predscore_data_valid-data_valid$IsBadBuy)^2))


lm_predscore_data_train=predict(lm_model2,newdata =data_train)
lm_RMSE_data_train=sqrt(mean((lm_predscore_data_train-data_train$IsBadBuy)^2))

data.frame(lm_RMSE_data_valid,lm_RMSE_data_train)
```

lm_RMSE_data_valid <dbl>	lm_RMSE_data_train <dbl>
0.4541079	0.449139

1 row

c. For each model, display the confusion matrix resulting from using a cutoff of 0.5 to do the classifications in the validation data set. Report the accuracy, TPR, and FPR. Which model is the most accurate?

ANSWER TO QUESTION 4c HERE:Given cutoff 0.5, for linear model, the accuracy is 0.6710831, TPR is 0.5516322, FPR is 0.2108037; for logistic model, the accuracy is 0.6558463, TPR is 0.3744171, FPR is 0.06587615. So the linear model is more accurate.

```
classify = function(scores, cutoff){
  classifications = ifelse(scores > cutoff, 1 ,0)  # Define a function that uses scores to classify based on a cutoff c
  return(classifications)}

classification_linear = classify(lm_predscore_data_valid, 0.5) #cutoff c=0.5
classification_logistic = classify(log_predscore_data_valid,0.5)


CM_linear = table(data_valid$IsBadBuy, classification_linear)
CM_linear
```

```
##    classification_linear
##        0    1
##   0 1198  320
##   1  673  828
```

```
TP_Linear = CM_linear[2,2]
FP_Linear = CM_linear[1,2]
TN_Linear = CM_linear[1,1]
FN_Linear = CM_linear[2,1]
data.frame(TP_Linear,FP_Linear,TN_Linear,FN_Linear)
```

TP_Linear <int>	FP_Linear <int>	TN_Linear <int>	FN_Linear <int>
828	320	1198	673

1 row

```
TPR_Linear = TP_Linear/(TP_Linear + FN_Linear)
TNR_Linear = TN_Linear/(TN_Linear + FP_Linear)
FPR_linear = 1 - TNR_Linear
accuracy_linear = (TP_Linear+TN_Linear)/(sum(CM_linear))
data.frame(TPR_Linear,TNR_Linear,FPR_linear,accuracy_linear)
```

TPR_Linear <dbl>	TNR_Linear <dbl>	FPR_linear <dbl>	accuracy_linear <dbl>
0.5516322	0.7891963	0.2108037	0.6710831

1 row

```
CM_log = table(data_valid$IsBadBuy, classification_logistic)
CM_log
```

```
##    classification_logistic
##        0    1
##   0 1418  100
##   1  939  562
```

```
TP_Log = CM_log[2,2]
FP_Log = CM_log[1,2]
TN_Log = CM_log[1,1]
FN_Log = CM_log[2,1]
data.frame(TP_Log,FP_Log,TN_Log,FN_Log)
```

TP_Log <int>	FP_Log <int>	TN_Log <int>	FN_Log <int>
562	100	1418	939

1 row

```
TPR_Log = TP_Log/(TP_Log + FN_Log)
TNR_Log = TN_Log/(TN_Log + FP_Log)
FPR_Log = 1 - TNR_Log
accuracy_Log = (TP_Log+TN_Log)/(sum(CM_log))
data.frame(TPR_Log,TNR_Log,FPR_Log,accuracy_Log)
```

TPR_Log <dbl>	TNR_Log <dbl>	FPR_Log <dbl>	accuracy_Log <dbl>
0.3744171	0.9341238	0.06587615	0.6558463

1 row

d. For the more accurate model, compute the accuracy, TPR, and FPR using cutoffs of .25 and .75 in the validation data. Which cutoff has the highest accuracy, highest TPR, and highest FPR?

ANSWER TO QUESTION 4d HERE: For cutoff 0.25, Accuracy is 0.5382577, TPR is 0.9626915, FPR is 0.8814229; For cutoff of .75, Accuracy is 0.6127857, TPR is 0.2345103, FPR is 0.01317523. So there's higher accuracy for 0.75 cutoff model.

```
classification_lm_25 = classify(lm_predscore_data_valid, 0.25) #cutoff c=0.25
CM_linear_25 = table(data_valid$IsBadBuy,classification_lm_25)
CM_linear_25
```

```
##    classification_lm_25
##        0    1
##   0  180 1338
##   1   56 1445
```

```
TP_Linear_25 = CM_linear_25[2,2]
FP_Linear_25 = CM_linear_25[1,2]
TN_Linear_25 = CM_linear_25[1,1]
FN_Linear_25 = CM_linear_25[2,1]
data.frame(TP_Linear_25,FP_Linear_25,TN_Linear_25,FN_Linear_25)
```

TP_Linear_25	FP_Linear_25	TN_Linear_25	FN_Linear_25
<int>	<int>	<int>	<int>
1445	1338	180	56

1 row

```
TPR_Linear_25 = TP_Linear_25/(TP_Linear_25 + FN_Linear_25)
TNR_Linear_25 = TN_Linear_25/(TN_Linear_25 + FP_Linear_25)
FPR_linear_25 = 1 - TNR_Linear_25
accuracy_linear_25 = (TP_Linear_25+TN_Linear_25)/(sum(CM_linear_25))
data.frame(TPR_Linear_25,TNR_Linear_25,FPR_linear_25,accuracy_linear_25)
```

TPR_Linear_25	TNR_Linear_25	FPR_linear_25	accuracy_linear_25
<dbl>	<dbl>	<dbl>	<dbl>
0.9626915	0.1185771	0.8814229	0.5382577

1 row

```
classification_lm_75 = classify(lm_predscore_data_valid,0.75) #cutoff c=0.75
CM_linear_75= table(data_valid$IsBadBuy,classification_lm_75)
CM_linear_75
```

```
##    classification_lm_75
##        0    1
##   0 1498   20
##   1 1149  352
```

```
TP_Linear_75 = CM_linear_75[2,2]
FP_Linear_75 = CM_linear_75[1,2]
TN_Linear_75 = CM_linear_75[1,1]
FN_Linear_75 = CM_linear_75[2,1]
data.frame(TP_Linear_75,FP_Linear_75,TN_Linear_75,FN_Linear_75)
```

TP_Linear_75 <int>	FP_Linear_75 <int>	TN_Linear_75 <int>	FN_Linear_75 <int>
352	20	1498	1149

1 row

```
TPR_Linear_75 = TP_Linear_75/(TP_Linear_75 + FN_Linear_75)
TNR_Linear_75 = TN_Linear_75/(TN_Linear_75 + FP_Linear_75)
FPR_linear_75 = 1 - TNR_Linear_75
accuracy_linear_75 = (TP_Linear_75+TN_Linear_75)/(sum(CM_linear_75))
data.frame(TPR_Linear_75,TNR_Linear_75,FPR_linear_75,accuracy_linear_75)
```

TPR_Linear_75 <dbl>	TNR_Linear_75 <dbl>	FPR_linear_75 <dbl>	accuracy_linear_75 <dbl>
0.2345103	0.9868248	0.01317523	0.6127857

1 row

e. In your opinion, which cutoff of the three yields the best results for this application? Explain your reasoning.

```
data.frame(accuracy_linear_25,accuracy_linear,accuracy_linear_75)
```

accuracy_linear_25 <dbl>	accuracy_linear <dbl>	accuracy_linear_75 <dbl>
0.5382577	0.6710831	0.6127857

1 row

ANSWER TO QUESTION 4e HERE: The cutoff for 0.5 model has the highest accuracy value, which is 0.6710831. We should choose this model