week 10 mod 8+9

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages --------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

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

```r
library(ModelMetrics)
```

```
##
## Attaching package: 'ModelMetrics'

## The following object is masked from 'package:base':
##
##     kappa
```

```r
library(modelr)
```

```
## Warning: package 'modelr' was built under R version 3.5.3

##
## Attaching package: 'modelr'

## The following objects are masked from 'package:ModelMetrics':
##
##     mae, mse, rmse
```

```r
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.5.3
```

NB: you need to download this zip file to your working directory for the class

```r
#unzip("DontGetKicked.zip")
```

```r
lemon<-read_csv("training.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   RefId = col_double(),
##   IsBadBuy = col_double(),
##   VehYear = col_double(),
##   VehicleAge = col_double(),
##   VehOdo = col_double(),
##   BYRNO = col_double(),
##   VNZIP1 = col_double(),
##   VehBCost = col_double(),
##   IsOnlineSale = col_double(),
##   WarrantyCost = col_double()
## )

## See spec(...) for full column specifications.
```

1. Calculate the proportion of lemons in the training dataset using the IsBadBuy variable.

```r
lemon%>%summarise(mean(IsBadBuy))
```

```
## # A tibble: 1 x 1
##   `mean(IsBadBuy)`
##              <dbl>
## 1            0.123
```

```r
prop.table(table(lemon$IsBadBuy))
```

```
##
##         0         1
## 0.8770125 0.1229875
```

2. Calculate the proportion of lemons by Make.

```r
prop.table(table(lemon$Make,lemon$IsBadBuy),margin = 1 )
```

```
##
##                          0          1
##   ACURA         0.72727273 0.27272727
##   BUICK         0.84305556 0.15694444
##   CADILLAC      0.84848485 0.15151515
##   CHEVROLET     0.90253942 0.09746058
##   CHRYSLER      0.87143826 0.12856174
##   DODGE         0.89676270 0.10323730
##   FORD          0.84590889 0.15409111
##   GMC           0.88443760 0.11556240
##   HONDA         0.89134809 0.10865191
##   HUMMER        1.00000000 0.00000000
##   HYUNDAI       0.87134180 0.12865820
##   INFINITI      0.66666667 0.33333333
##   ISUZU         0.93283582 0.06716418
##   JEEP          0.84549878 0.15450122
##   KIA           0.88244767 0.11755233
##   LEXUS         0.64516129 0.35483871
##   LINCOLN       0.70103093 0.29896907
```

```
##   MAZDA        0.83861083 0.16138917
##   MERCURY      0.83023001 0.16976999
##   MINI         0.66666667 0.33333333
##   MITSUBISHI   0.88058252 0.11941748
##   NISSAN       0.84028777 0.15971223
##   OLDSMOBILE   0.79835391 0.20164609
##   PLYMOUTH     0.50000000 0.50000000
##   PONTIAC      0.88093001 0.11906999
##   SATURN       0.85852982 0.14147018
##   SCION        0.91472868 0.08527132
##   SUBARU       0.78571429 0.21428571
##   SUZUKI       0.85316265 0.14683735
##   TOYOTA       0.90034965 0.09965035
##   TOYOTA SCION 1.00000000 0.00000000
##   VOLKSWAGEN   0.85820896 0.14179104
##   VOLVO        1.00000000 0.00000000
```

```r
lemon%>%
  group_by(Make)%>%
  summarise(mean_lemon=mean(IsBadBuy))%>%
  arrange(-mean_lemon)%>%
  print(n=100)
```

```
## # A tibble: 33 x 2
##    Make         mean_lemon
##    <chr>             <dbl>
##  1 PLYMOUTH         0.5
##  2 LEXUS            0.355
##  3 INFINITI         0.333
##  4 MINI             0.333
##  5 LINCOLN          0.299
##  6 ACURA            0.273
##  7 SUBARU           0.214
##  8 OLDSMOBILE       0.202
##  9 MERCURY          0.170
## 10 MAZDA            0.161
## 11 NISSAN           0.160
## 12 BUICK            0.157
## 13 JEEP             0.155
## 14 FORD             0.154
## 15 CADILLAC         0.152
## 16 SUZUKI           0.147
## 17 VOLKSWAGEN       0.142
## 18 SATURN           0.141
## 19 HYUNDAI          0.129
## 20 CHRYSLER         0.129
## 21 MITSUBISHI       0.119
## 22 PONTIAC          0.119
## 23 KIA              0.118
## 24 GMC              0.116
## 25 HONDA            0.109
## 26 DODGE            0.103
## 27 TOYOTA           0.0997
## 28 CHEVROLET        0.0975
## 29 SCION            0.0853
```

```
## 30 ISUZU            0.0672
## 31 HUMMER           0
## 32 TOYOTA SCION     0
## 33 VOLVO            0
```

3. Now, predict the probability of being a lemon using a linear model (`lm(y~x)`, with covariates of your choosing from the training dataset.
4. Make predictions from the linear model.

```r
lin_mod<-lm(IsBadBuy~VehicleAge+VehBCost,data=lemon)

lemon%>%add_predictions(lin_mod)->lemon

# Hint -- lower threshold to improve predicted class distribution
lemon%>%mutate(lin_mod_out=ifelse(pred>.25,1,0))->lemon
```

5 + 6. Now, predict the probability of being a lemon using a logistic regression.

```r
logit_mod<-glm(IsBadBuy~
                VehicleAge+
                 VehBCost,
               data=lemon,
               family=binomial(link="logit"))
summary(logit_mod)
```

```
##
## Call:
## glm(formula = IsBadBuy ~ VehicleAge + VehBCost, family = binomial(link = "logit"),
##     data = lemon)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0368  -0.5387  -0.4523  -0.3705   3.5062
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.525e+00  6.379e-02  -39.58   <2e-16 ***
## VehicleAge   2.569e-01  6.867e-03   37.41   <2e-16 ***
## VehBCost    -9.091e-05  6.877e-06  -13.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 54421  on 72982  degrees of freedom
## Residual deviance: 52264  on 72980  degrees of freedom
## AIC: 52270
##
## Number of Fisher Scoring iterations: 5
```

```r
lemon%>%
  mutate(pred_logit=predict(logit_mod,type="response"))->lemon

## Classifying 1s and 0s
# Hint -- lower threshold to improve predicted class distribution
```

```r
lemon%>%mutate(pred_logit_out=ifelse(pred_logit>.25,1,0))->lemon
```

#Confusion Matrix 7. Create confusion matrix and compare.

```r
## Linear Model
caret::confusionMatrix(data=as.factor(as.character(lemon$pred_logit_out)),
                       reference=as.factor(as.character(lemon$IsBadBuy)),positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 61623  8006
##          1  2384   970
##
##                Accuracy : 0.8576
##                  95% CI : (0.8551, 0.8602)
##     No Information Rate : 0.877
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0969
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.10807
##             Specificity : 0.96275
##          Pos Pred Value : 0.28921
##          Neg Pred Value : 0.88502
##              Prevalence : 0.12299
##          Detection Rate : 0.01329
##    Detection Prevalence : 0.04596
##       Balanced Accuracy : 0.53541
##
##        'Positive' Class : 1
##
```

```r
ModelMetrics::recall(actual=lemon$IsBadBuy,predicted=lemon$lin_mod_out)
```

```
## [1] 0.06038324
```

```r
ModelMetrics::tnr(actual=lemon$IsBadBuy,predicted=lemon$lin_mod_out)
```

```
## [1] 0.9805178
```

```r
## Logit Model
caret::confusionMatrix(data=as.factor(lemon$pred_logit_out),
                       reference=as.factor(lemon$IsBadBuy),
                       positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 61623  8006
##          1  2384   970
##
##                Accuracy : 0.8576
```

```
##                  95% CI : (0.8551, 0.8602)
##     No Information Rate : 0.877
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0969
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.10807
##             Specificity : 0.96275
##          Pos Pred Value : 0.28921
##          Neg Pred Value : 0.88502
##              Prevalence : 0.12299
##          Detection Rate : 0.01329
##    Detection Prevalence : 0.04596
##       Balanced Accuracy : 0.53541
##
##        'Positive' Class : 1
##
```

```
ModelMetrics::recall(actual=lemon$IsBadBuy,predicted=lemon$pred_logit_out)
```

```
## [1] 0.108066
```

```
ModelMetrics::tnr(actual=lemon$IsBadBuy,predicted=lemon$lin_mod_out)
```

```
## [1] 0.9805178
```

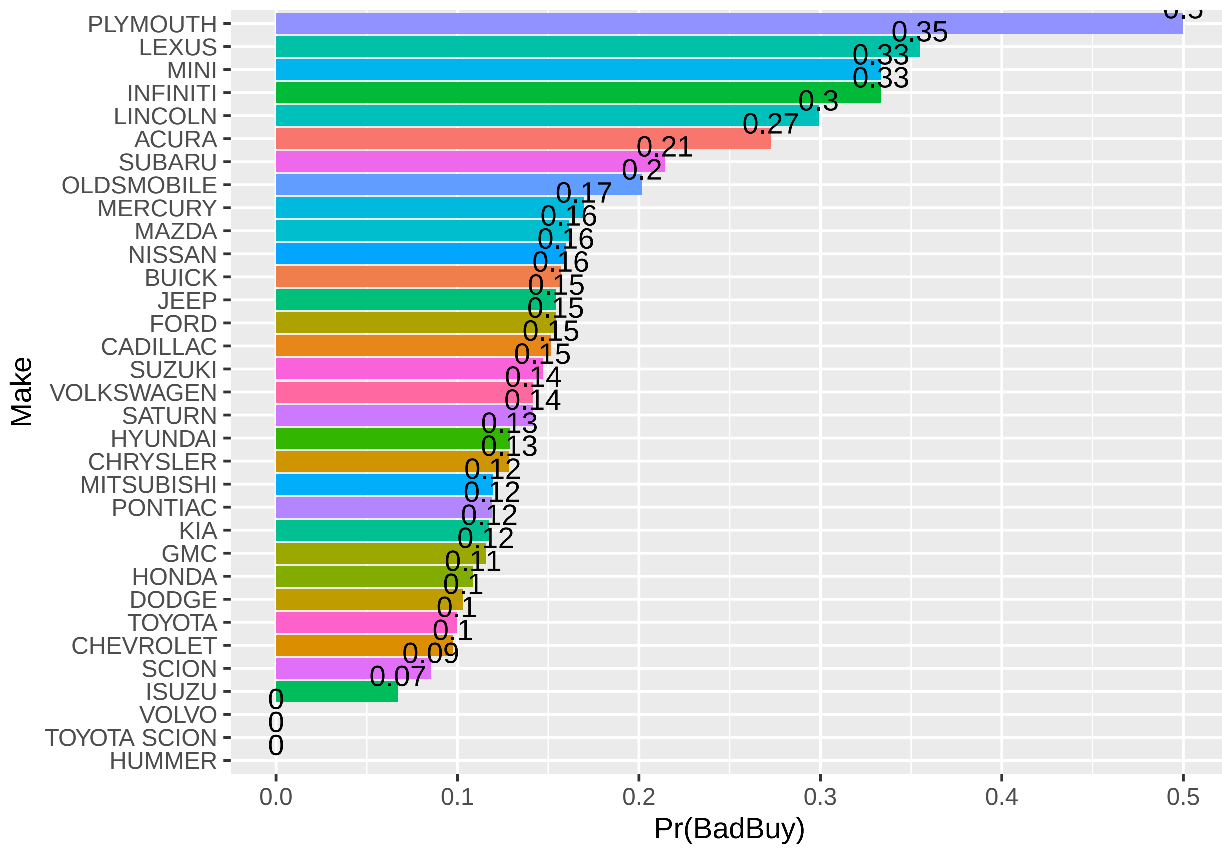
8. Plot distribution of lemon by factors.

```
lemon_sum<-lemon%>%
  group_by(Make)%>%
  summarize(prob_badbuy=mean(IsBadBuy))%>%
  arrange(-prob_badbuy)

gg1<-ggplot(lemon_sum,aes(y=prob_badbuy,
                          x=fct_reorder(.f=as.factor(Make),.x=prob_badbuy),
                          fill=Make))


gg1<-gg1+geom_bar(stat="identity",position="dodge")
gg1<-gg1+xlab("Make")+ylab("Pr(BadBuy)")
gg1<-gg1+theme(legend.title=element_blank(),legend.position = "none")
gg1<-gg1+coord_flip()

gg1<-gg1+geom_text(aes(label=round(prob_badbuy,2)),
                   position=position_dodge(width=.9),
                   vjust=-.25)
gg1
```



9. Create a table that shows the probability of a car being a bad buy by make.

```
kable(lemon_sum)
```

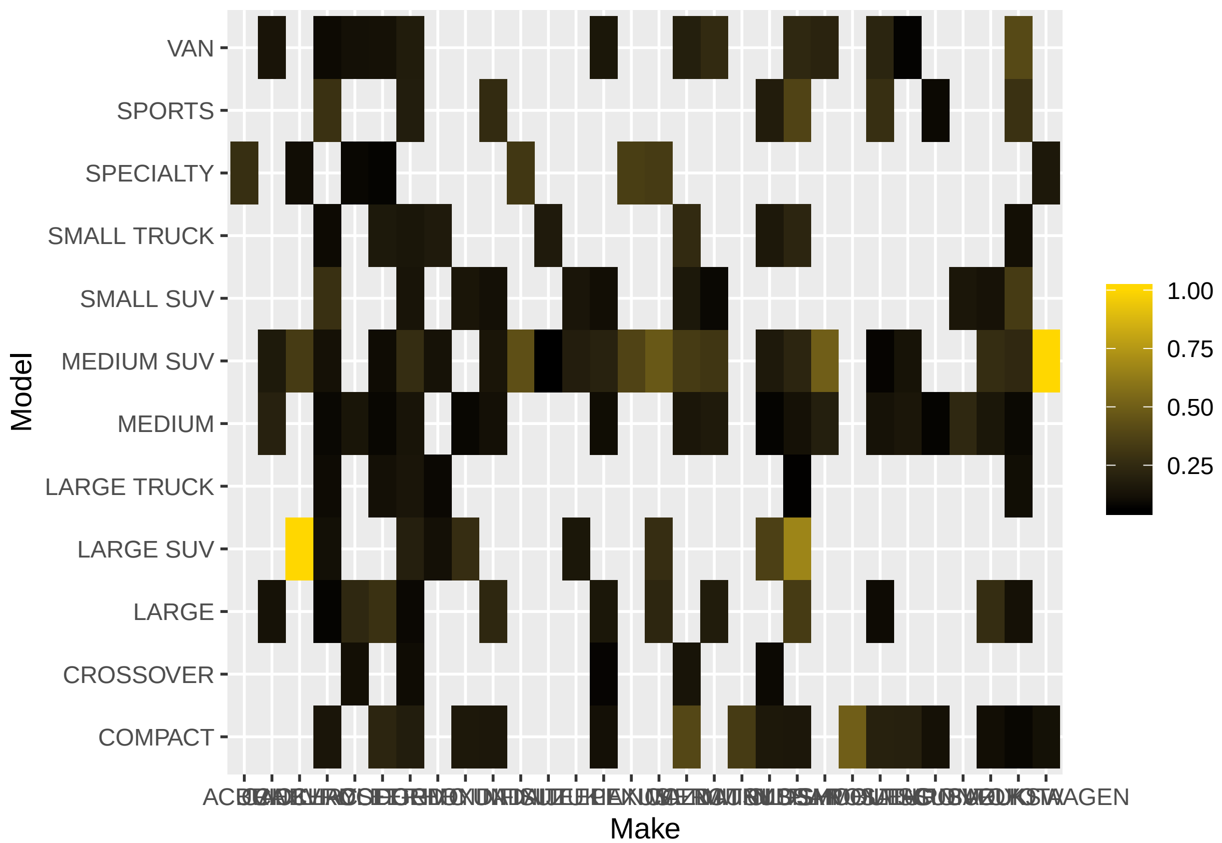
Make	prob_badbuy
PLYMOUTH	0.5000000
LEXUS	0.3548387
INFINITI	0.3333333
MINI	0.3333333
LINCOLN	0.2989691
ACURA	0.2727273
SUBARU	0.2142857
OLDSMOBILE	0.2016461
MERCURY	0.1697700
MAZDA	0.1613892
NISSAN	0.1597122
BUICK	0.1569444
JEEP	0.1545012
FORD	0.1540911
CADILLAC	0.1515152
SUZUKI	0.1468373
VOLKSWAGEN	0.1417910
SATURN	0.1414702
HYUNDAI	0.1286582
CHRYSLER	0.1285617
MITSUBISHI	0.1194175

Make	prob_badbuy
PONTIAC	0.1190700
KIA	0.1175523
GMC	0.1155624
HONDA	0.1086519
DODGE	0.1032373
TOYOTA	0.0996503
CHEVROLET	0.0974606
SCION	0.0852713
ISUZU	0.0671642
HUMMER	0.0000000
TOYOTA SCION	0.0000000
VOLVO	0.0000000

Bonus. Create a heatmap of the probability of a car being a bad buy by make and size.

```
lemon_sum<-lemon%>%
  group_by(Make,Size)%>%
  summarize(prob_badbuy=mean(IsBadBuy))%>%
  arrange(-prob_badbuy)%>%
  filter(prob_badbuy>0)

gg<-ggplot(lemon_sum,
           aes(x=as.factor(Make),
               y=as.factor(Size),fill=prob_badbuy))
gg<-gg+geom_tile()
gg<-gg+scale_fill_gradient(low="black",high="gold")
gg<-gg+xlab("Make")+ylab("Model")
gg<-gg+theme(legend.title=element_blank())
gg
```


VAN
SPORTS
SPECIALTY
SMALL TRUCK
SMALL SUV
MEDIUM SUV
MEDIUM
LARGE TRUCK
LARGE SUV
LARGE
CROSSOVER
COMPACT
Model
Make
1.00
0.75
0.50
0.25