08-assignment

Doyle

4/6/2021

Assignment 8

"When life gives you lemons, don't make lemonade. Make life take the lemons back! Get mad!" -Cave Johnson

For this assignment, you'll be using the lemons dataset, which is a subset of the dataset used for a Kaggle competition described here: https://www.kaggle.com/c/DontGetKicked/data. Your job is to predict which cars are most likely to be lemons. Please note

Complete the following steps.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

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

```
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip

## -- Attaching packages --------------------------------------- tidymodels 0.1.4 --

## v broom        0.7.9      v rsample      0.1.0
## v dials        0.0.10     v tune         0.1.6
## v infer        1.0.0      v workflows    0.2.4
## v modeldata    0.1.1      v workflowsets 0.1.0
## v parsnip      0.1.7      v yardstick    0.0.8
## v recipes      0.1.17
```

```
## -- 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 yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(knitr)
library(probably)
```

```
##
## Attaching package: 'probably'

## The following objects are masked from 'package:base':
##
##     as.factor, as.ordered
```

```
library(modelr)
```

```
##
## Attaching package: 'modelr'

## The following objects are masked from 'package:yardstick':
##
##     mae, mape, rmse

## The following object is masked from 'package:broom':
##
##     bootstrap
```

Read just the first 100k rows

```
df<-read_csv("training.csv",n_max=1e6)
```

```
## Rows: 72983 Columns: 34

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (24): PurchDate, Auction, Make, Model, Trim, SubModel, Color, Transmissi...
## dbl (10): RefId, IsBadBuy, VehYear, VehicleAge, VehOdo, BYRNO, VNZIP1, VehBC...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Recode isbadbuy to be a factor, required for `recipe` command and others.

```
df<-df%>%
  mutate(isbadbuy_f=fct_recode(as.factor(IsBadBuy),"Yes"="1","No"="0"))
```

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

```
df%>%
  summarize(mean(IsBadBuy,na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   `mean(IsBadBuy, na.rm = TRUE)`
##                            <dbl>
## 1                          0.123
```

2. Calculate the proportion of lemons by Make.

```
df%>%
  group_by(Make)%>%
    summarize(mean_isbabduy=mean(IsBadBuy,na.rm=TRUE))%>%
    arrange(-mean_isbabduy)%>%
    kable()
```

Make	mean_isbabduy
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
PONTIAC	0.1190700
KIA	0.1175523
GMC	0.1155624
HONDA	0.1086519
DODGE	0.1032373
TOYOTA	0.0996503

Make	mean_isbabduy
CHEVROLET	0.0974606
SCION	0.0852713
ISUZU	0.0671642
HUMMER	0.0000000
TOYOTA SCION	0.0000000
VOLVO	0.0000000

3. Now, predict the probability of being a lemon using a logistic regression, using covariates of your choosing.

```
lemon_split<-initial_split(df)

lemon_train<-training(lemon_split)

lemon_test<-testing(lemon_split)
```

```
lemon_formula<-as.formula("isbadbuy_f~
                          VehicleAge+
                          Make+
                          Color+
                          Transmission+
                          VNST+
                          VehBCost+
                          VehOdo")
```

```
lemon_recipe<-recipe(lemon_formula,lemon_train)%>%
    step_other(Make,threshold = .1)%>%
  step_dummy(all_nominal(),-all_outcomes())%>%
  step_log(VehBCost,VehOdo)%>%
  step_naomit(all_predictors())
```

```
lemon_model<-
  logistic_reg()%>%
  set_engine("glm")%>%
  set_mode("classification")
```

```
lemon_wf<-workflow()%>%
  add_recipe(lemon_recipe)%>%
  add_model(lemon_model)
```

```
lemon_results<-fit(lemon_wf,lemon_train)
```

```
## Warning: There are new levels in a factor: NA
```

```
lemon_results%>%tidy()%>%kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-3.6386270	0.9028186	-4.0302966	0.0000557
VehicleAge	0.2368309	0.0087112	27.1868485	0.0000000
VehBCost	-0.6401235	0.0501717	-12.7586588	0.0000000
VehOdo	0.5565867	0.0677927	8.2101232	0.0000000
Make_CHRYSLER	0.3684968	0.0492235	7.4861946	0.0000000
Make_DODGE	0.1432616	0.0461048	3.1073043	0.0018880
Make_FORD	0.2713561	0.0437272	6.2056547	0.0000000
Make_other	0.2958799	0.0384427	7.6966545	0.0000000
Color_BLACK	-0.0294653	0.0961004	-0.3066102	0.7591401
Color_BLUE	-0.1070260	0.0937829	-1.1412096	0.2537827
Color_BROWN	-0.1691377	0.1963080	-0.8615935	0.3889112
Color_GOLD	-0.0266789	0.0982454	-0.2715533	0.7859655
Color_GREEN	-0.1817415	0.1064813	-1.7067927	0.0878606
Color_GREY	-0.0529460	0.0953967	-0.5550086	0.5788888
Color_MAROON	-0.1572522	0.1168365	-1.3459164	0.1783295
Color_NOT.AVAIL	0.9714216	0.2851865	3.4062677	0.0006586
Color_NULL.	-11.0304221	156.0526980	-0.0706840	0.9436493
Color_ORANGE	-0.2830763	0.2366762	-1.1960489	0.2316775
Color_OTHER	0.0592729	0.2406316	0.2463221	0.8054329
Color_PURPLE	0.0128076	0.1932020	0.0662912	0.9471460
Color_RED	0.0038923	0.0969323	0.0401544	0.9679700
Color_SILVER	-0.0943466	0.0911478	-1.0350950	0.3006245
Color_WHITE	-0.0295066	0.0919076	-0.3210462	0.7481754
Color_YELLOW	0.0167474	0.2273892	0.0736506	0.9412884
Transmission_Manual	-10.4216384	324.7437018	-0.0320919	0.9743988
Transmission_MANUAL	-0.3522185	0.0745840	-4.7224418	0.0000023
Transmission_NULL.	NA	NA	NA	NA
VNST_AR	0.2426446	0.3524718	0.6884086	0.4911955
VNST_AZ	-0.1750035	0.1404107	-1.2463687	0.2126291
VNST_CA	0.0077523	0.1382546	0.0560727	0.9552839
VNST_CO	-0.2047855	0.1413802	-1.4484740	0.1474845
VNST_FL	-0.4099423	0.1374310	-2.9828955	0.0028554
VNST_GA	-0.3373105	0.1522222	-2.2159080	0.0266978
VNST_IA	0.0144436	0.2041287	0.0707574	0.9435909
VNST_ID	-0.4077044	0.3236066	-1.2598769	0.2077138
VNST_IL	0.2217841	0.2018921	1.0985280	0.2719740
VNST_IN	-0.0341339	0.2010705	-0.1697608	0.8651982
VNST_KY	-1.0276489	0.3877331	-2.6504027	0.0080396
VNST_LA	-0.0199041	0.2224410	-0.0894803	0.9287002
VNST_MA	-0.1650743	0.8175064	-0.2019242	0.8399760
VNST_MD	0.0597107	0.1649455	0.3620027	0.7173500
VNST_MI	-0.4455183	1.0711754	-0.4159154	0.6774719
VNST_MN	-1.0294273	0.7401904	-1.3907601	0.1642982
VNST_MO	-0.2071018	0.1899002	-1.0905821	0.2754568
VNST_MS	-0.2354194	0.2232497	-1.0545116	0.2916488
VNST_NC	-0.2972089	0.1394451	-2.1313684	0.0330588
VNST_NE	-10.3382967	75.8953632	-0.1362178	0.8916491
VNST_NH	0.4040507	0.4483845	0.9011256	0.3675216
VNST_NJ	-0.0407101	0.2383952	-0.1707671	0.8644069
VNST_NM	-0.2616645	0.2749556	-0.9516610	0.3412689
VNST_NV	0.0544950	0.1864483	0.2922792	0.7700732
VNST_NY	-10.1130078	144.8629109	-0.0698109	0.9443442

term	estimate	std.error	statistic	p.value
VNST_OH	-0.3246171	0.1959052	-1.6570107	0.0975173
VNST_OK	0.1330610	0.1494537	0.8903157	0.3732964
VNST_OR	-0.5989209	0.3427207	-1.7475481	0.0805423
VNST_PA	0.0531999	0.1713976	0.3103891	0.7562651
VNST_SC	-0.1812445	0.1427626	-1.2695516	0.2042444
VNST_TN	-0.2451770	0.1588569	-1.5433832	0.1227378
VNST_TX	-0.0186293	0.1353163	-0.1376719	0.8904997
VNST_UT	-0.1090544	0.1781348	-0.6122012	0.5404047
VNST_VA	0.0548987	0.1544961	0.3553407	0.7223344
VNST_WA	-0.6639591	0.4175466	-1.5901439	0.1118024
VNST_WV	-0.3492630	0.2602650	-1.3419515	0.1796117

4. Make predictions from the logit model. Make sure these are probabilities.

```
lemon_results%>%
  predict(lemon_test)%>%
  bind_cols(lemon_test)
```

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

```
## # A tibble: 18,246 x 36
##    .pred_class RefId IsBadBuy PurchDate  Auction VehYear VehicleAge Make   Model
##    <fct>       <dbl>    <dbl> <chr>      <chr>     <dbl>      <dbl> <chr>  <chr>
##  1 No              5        0 12/7/2009  ADESA      2005          4 FORD   FOCUS
##  2 No              7        0 12/7/2009  ADESA      2004          5 KIA    SPEC~
##  3 No              8        0 12/7/2009  ADESA      2005          4 FORD   TAUR~
##  4 No             16        0 12/14/2009 ADESA      2003          6 CHEVR~ CAVA~
##  5 No             17        0 12/14/2009 ADESA      2005          4 CHEVR~ TRAI~
##  6 No             39        0 1/4/2010   ADESA      2005          5 CHRYS~ 300
##  7 No             43        0 1/11/2010  ADESA      2004          6 CHRYS~ PACI~
##  8 No             49        1 1/11/2010  ADESA      2004          6 FORD   FOCUS
##  9 No             53        0 1/18/2010  ADESA      2006          4 CHEVR~ HHR
## 10 No             65        0 1/18/2010  ADESA      2005          5 DODGE  CARA~
## # ... with 18,236 more rows, and 27 more variables: Trim <chr>, SubModel <chr>,
## #   Color <chr>, Transmission <chr>, WheelTypeID <chr>, WheelType <chr>,
## #   VehOdo <dbl>, Nationality <chr>, Size <chr>, TopThreeAmericanName <chr>,
## #   MMRAcquisitionAuctionAveragePrice <chr>,
## #   MMRAcquisitionAuctionCleanPrice <chr>,
## #   MMRAcquisitionRetailAveragePrice <chr>,
## #   MMRAcquisitonRetailCleanPrice <chr>, ...
```

5. Calculate the accuracy, sensitivity and specificity of your model using a threshold of .5.

```
lemon_results%>%
  predict(lemon_test)%>%
  bind_cols(lemon_test)%>%
   metrics(truth=isbadbuy_f,estimate=.pred_class)
```

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

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary      0.878
## 2 kap      binary      0.000240
```

```
lemon_results%>%
  predict(lemon_test)%>%
  bind_cols(lemon_test)%>%
  sens(truth=isbadbuy_f,estimate=.pred_class,event_level="second")
```

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

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary      0.000449
```

```
lemon_results%>%
  predict(lemon_test)%>%
  bind_cols(lemon_test)%>%
  spec(truth=isbadbuy_f,estimate=.pred_class,event_level="second")
```

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

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary          1.00
```

5. Calculate the AUC for the predictions from the ROC based on the logit model.

```
lemon_results%>%
  predict(lemon_test,type="prob")%>%
  bind_cols(lemon_test)%>%
  roc_auc(truth=isbadbuy_f,estimate=.pred_Yes,event_level="second")
```

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

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.664
```

```
lemon_results%>%
  predict(lemon_test,type="prob")%>%
  bind_cols(lemon_test)%>%
  threshold_perf(truth=isbadbuy_f,
                 estimate=.pred_Yes,
                 thresholds=.12,metrics=c("sens","spec"))
```

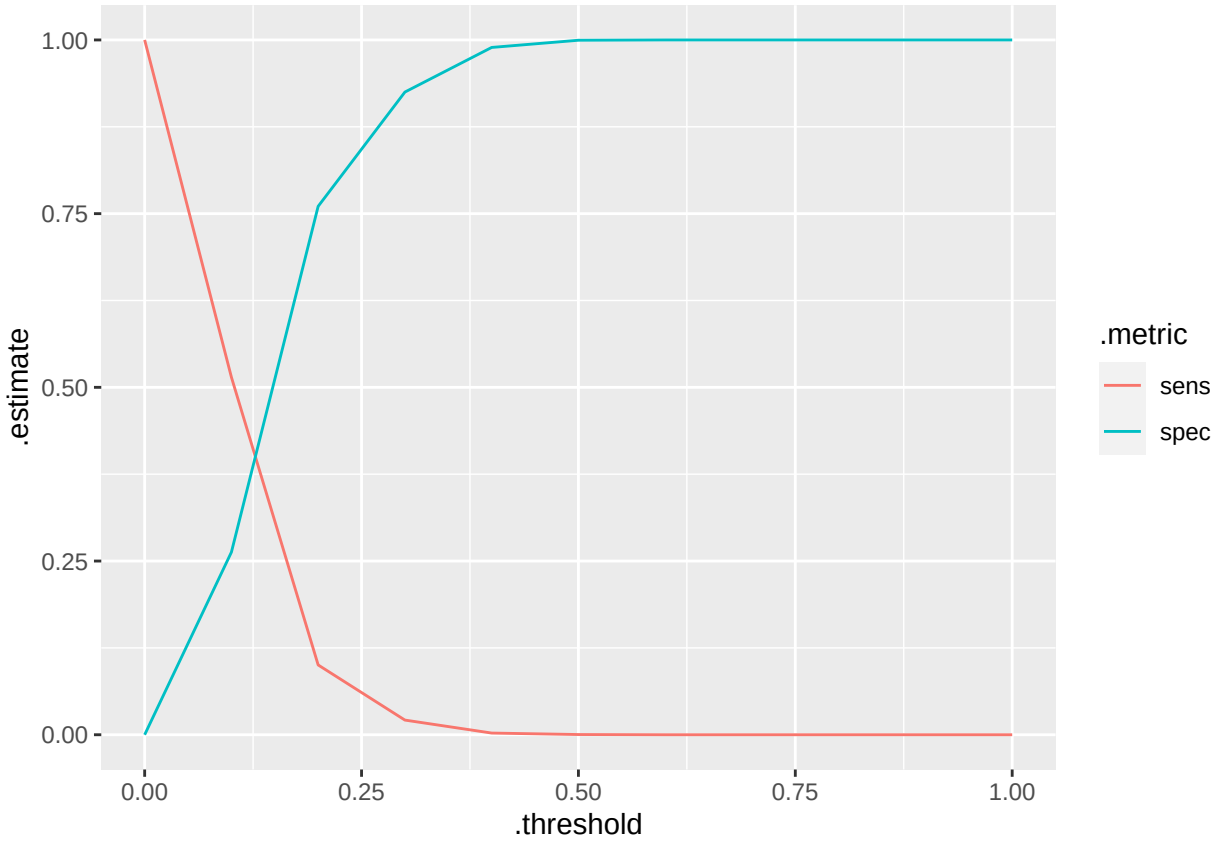
```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## # A tibble: 4 x 4
##   .threshold .metric  .estimator .estimate
##        <dbl> <chr>    <chr>          <dbl>
## 1       0.12 sens     binary         0.369
## 2       0.12 spec     binary         0.383
## 3       0.12 j_index  binary        -0.248
## 4       0.12 distance binary         0.779
```

```
th<-lemon_results%>%
  predict(lemon_test,type="prob")%>%
  bind_cols(lemon_test)%>%
   threshold_perf(truth=isbadbuy_f,
                  estimate=.pred_Yes,
                  thresholds=seq(0,1,by=.1),metrics=c("sens","spec"))
```

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

```
ggplot(filter(th,.metric%in%c("sens","spec")),
       aes(x=.threshold,y=.estimate,color=.metric))+
  geom_line()
```



```
lemon_mod<-extract_model(lemon_results)
```

```
## Warning: ‘extract_model()‘ was deprecated in tune 0.1.6.
## Please use ‘extract_fit_engine()‘ instead.
```

```
lemon_data<-lemon_recipe%>%
  prep()%>%
  juice()
```

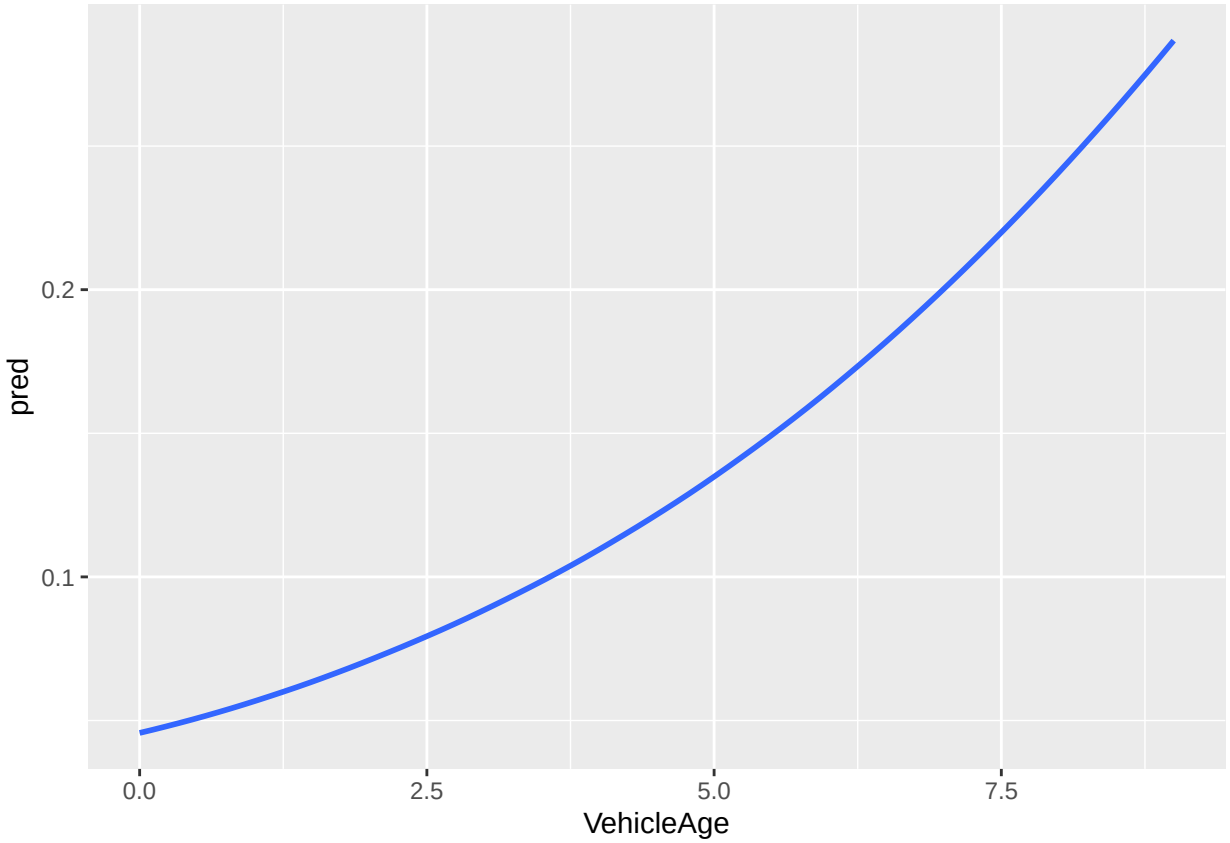
```
## Warning: There are new levels in a factor: NA
```

```
hypo_data<-
  data_grid(
            data=lemon_data,
            .model=lemon_mod,
            VehicleAge=seq_range(VehicleAge,n=10)
  )%>%
  add_predictions(lemon_mod,type="response")
```

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

```
hypo_data%>%
  ggplot(aes(x=VehicleAge,y=pred))+
  geom_smooth()
```

```
## ‘geom_smooth()‘ using method = 'loess' and formula 'y ~ x'
```


0.2
0.1
pred
0.0
2.5
5.0
7.5
VehicleAge