Logistic Regression

Alyssa Shi

2020-06-02

Load packages & data

```
library(tidyverse)
library(knitr)
library(broom)
library(pROC)
library(plotROC)
```

```
spotify <- read.csv("Data/spotify.txt")
```

Data Wrangling and EDA

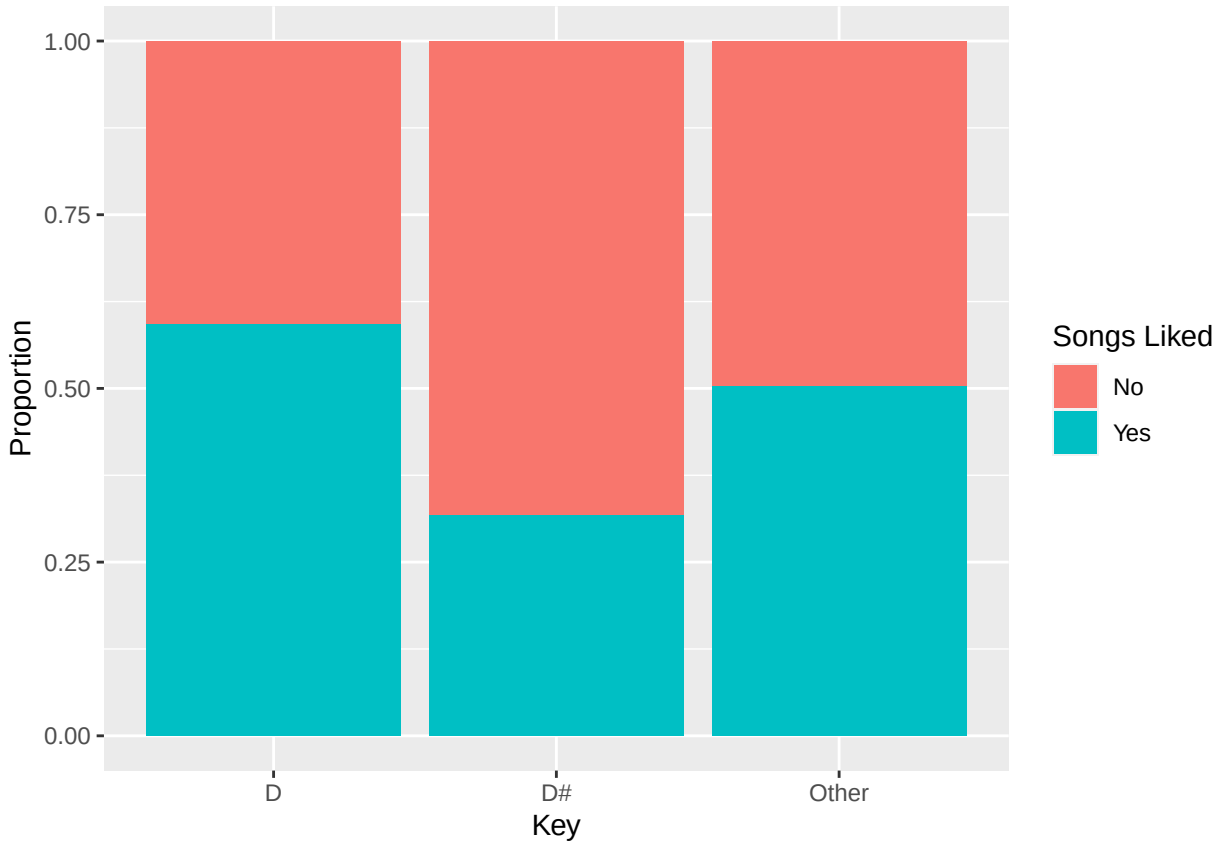
Refactoring variables:

```
spotify <- spotify %>%
  mutate(key = case_when(
    key == 2 ~ "D",
    key == 3 ~ "D#",
    key == 0 | key == 1 | key > 3 ~ "Other"),
    key = as.factor(key),
    like = as.factor(like))

levels(spotify$like) <- c("No", "Yes")
```

EDA:

```
ggplot(data = spotify, mapping = aes(x = key, fill = like)) +
  geom_bar(position = "fill") +
  labs(y = "Proportion", x = 'Key', fill = "Songs Liked")
```




```
spotify %>%
  group_by(key, like) %>%
  summarise(n = n()) %>%
  mutate(prop = n / sum(n)) %>%
  filter(like == "Yes")
```

```
## # A tibble: 3 x 4
## # Groups:   key [3]
##   key   like      n  prop
##   <fct> <fct> <int> <dbl>
## 1 D     Yes     109 0.592
## 2 D#    Yes      20 0.317
## 3 Other Yes     891 0.503
```

Songs in the key D have the highest proportion of likes (59.24%), followed by Other (50.34%) and D# (31.75%).

Model Fitting

Fit a logistic regression model with `like` as the response variable and the following as predictors: `acousticness`, `danceability`, `duration_ms`, `instrumentalness`, `loudness`, `speechiness`, and `valence`:

```
logit_red <- glm(like ~ acousticness + danceability + duration_ms +
                  instrumentalness + loudness + speechiness + valence,
                  spotify, family = binomial)
tidy(logit_red, conf.int = T) %>%
  kable(format = "markdown", digits = 3)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-2.955	0.276	-10.693	0	-3.504	-2.420
acousticness	-1.722	0.240	-7.182	0	-2.197	-1.257
danceability	1.630	0.344	4.737	0	0.958	2.308
duration_ms	0.000	0.000	4.225	0	0.000	0.000
instrumentalness	1.353	0.207	6.549	0	0.952	1.763
loudness	-0.087	0.017	-5.062	0	-0.122	-0.054
speechiness	4.072	0.583	6.985	0	2.947	5.234
valence	0.856	0.223	3.836	0	0.420	1.296

Drop in Deviance

Add the variable `key` to the model created above and use a drop-in deviance test to see if this variable should be included.

The null hypothesis is that all the new coefficients (levels D, D#, or Other) of `key` are equal to zero.

The alternative hypothesis is that at least one of the new coefficients (levels D, D#, or Other) of `key` is not equal to zero.

```
logit_full <- glm(like ~ acousticness + danceability + duration_ms +
                  instrumentalness + loudness + speechiness + valence + key,
                  spotify, family = binomial)
anova(logit_red, logit_full, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: like ~ acousticness + danceability + duration_ms + instrumentalness +
##     loudness + speechiness + valence
## Model 2: like ~ acousticness + danceability + duration_ms + instrumentalness +
##     loudness + speechiness + valence + key
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      2009     2518.5
## 2      2007     2505.2  2   13.357 0.001258 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Because the p-value for the drop-in deviance test is 0.001258 which is less than .05, our model would benefit from adding `key` as a predictor variable. We can double check this conclusion using AIC.

```
logit_red$aic
```

```
## [1] 2534.517
```

```
logit_full$aic
```

```
## [1] 2525.16
```

The AIC for the model from Exercise 3 with the addition of `key` as a predictor variable is 2525.16 compared to the AIC of the model without the addition of `key` at 2534.517. The model with `key` is more ideal because a lower AIC indicates a better fitted model.

Interpretation

```r
tidy(logit_full, conf.int = T) %>%
  kable(format = "markdown", digits = 3)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-2.509	0.311	-8.068	0.000	-3.124	-1.904
acousticness	-1.702	0.241	-7.065	0.000	-2.179	-1.234
danceability	1.649	0.345	4.774	0.000	0.975	2.329
duration_ms	0.000	0.000	4.187	0.000	0.000	0.000
instrumentalness	1.383	0.207	6.667	0.000	0.981	1.795
loudness	-0.087	0.017	-5.018	0.000	-0.121	-0.053
speechiness	4.034	0.585	6.896	0.000	2.905	5.199
valence	0.881	0.224	3.927	0.000	0.442	1.322
keyD#	-1.073	0.335	-3.204	0.001	-1.745	-0.428
keyOther	-0.494	0.169	-2.923	0.003	-0.828	-0.165

The predicted odds that a user likes a song in the key D# are exp{-1.073)} = 0.342 times the odds of a like for songs in the key D (the baseline), holding all else constant.

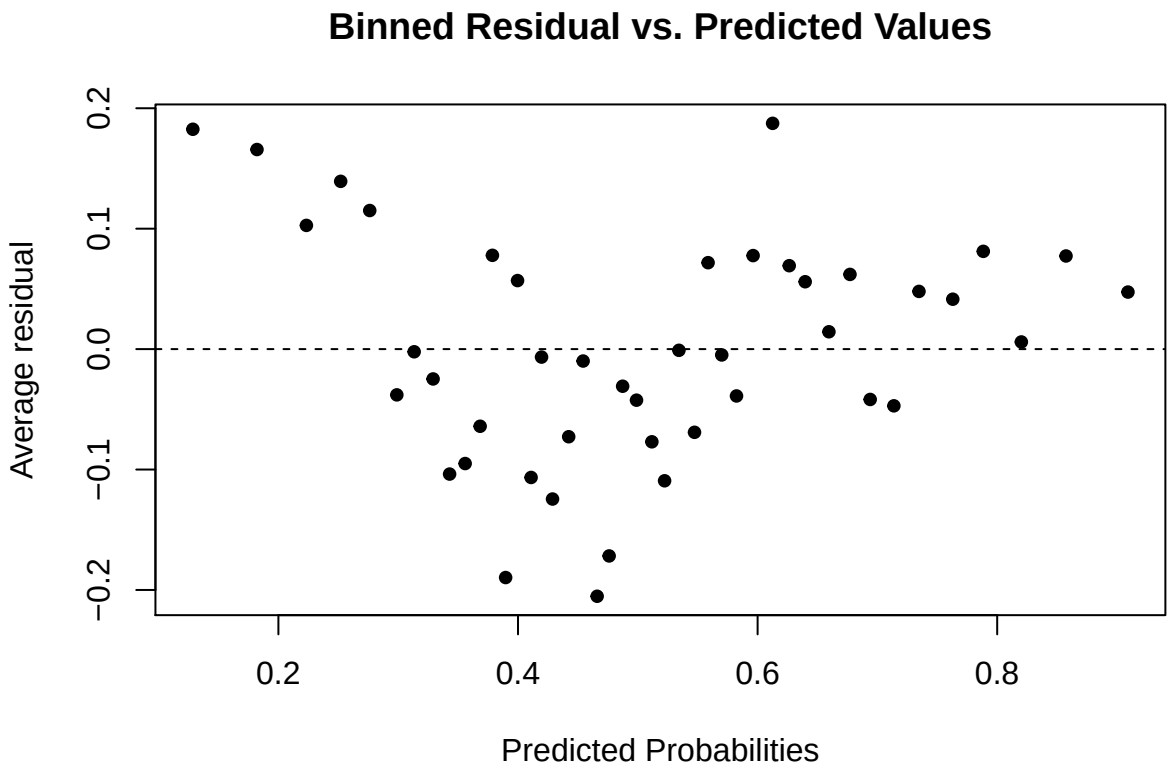
Calculating Predicted Probabilities and Residuals

```r
logit_aug <- augment(logit_full,
                         type.predict = "response",
                         type.residuals = "response")
```

Checking Assumptions (Linearity)

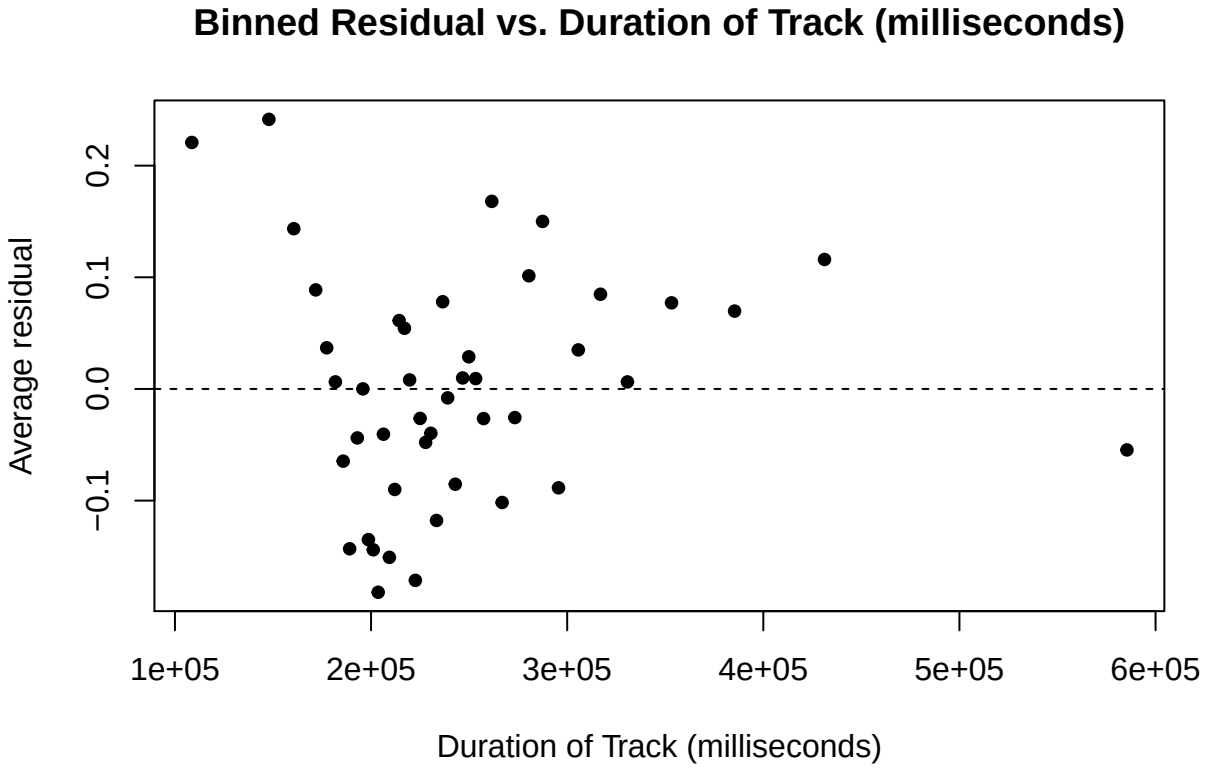
Create a binned plot of the residuals versus the predicted probabilities:

```r
arm::binnedplot(x = logit_aug$.fitted,
                y = logit_aug$.resid,
                xlab = "Predicted Probabilities",
                main = "Binned Residual vs. Predicted Values",
                col.int = FALSE)
```



We chose the quantitative variable `duration_ms` and used a binned residuals plot to examine the residuals versus this predictor variable.

```
arm::binnedplot(x = logit_aug$duration_ms,
                y = logit_aug$.resid,
                col.int = FALSE,
                xlab = "Duration of Track (milliseconds)",
                main = "Binned Residual vs. Duration of Track (milliseconds)")
```



We chose the categorical variable `key` and found the mean value of the residuals for each of the three levels.

```
logit_aug %>%
  group_by(key) %>%
  summarise(mean_resid = mean(.resid))
```

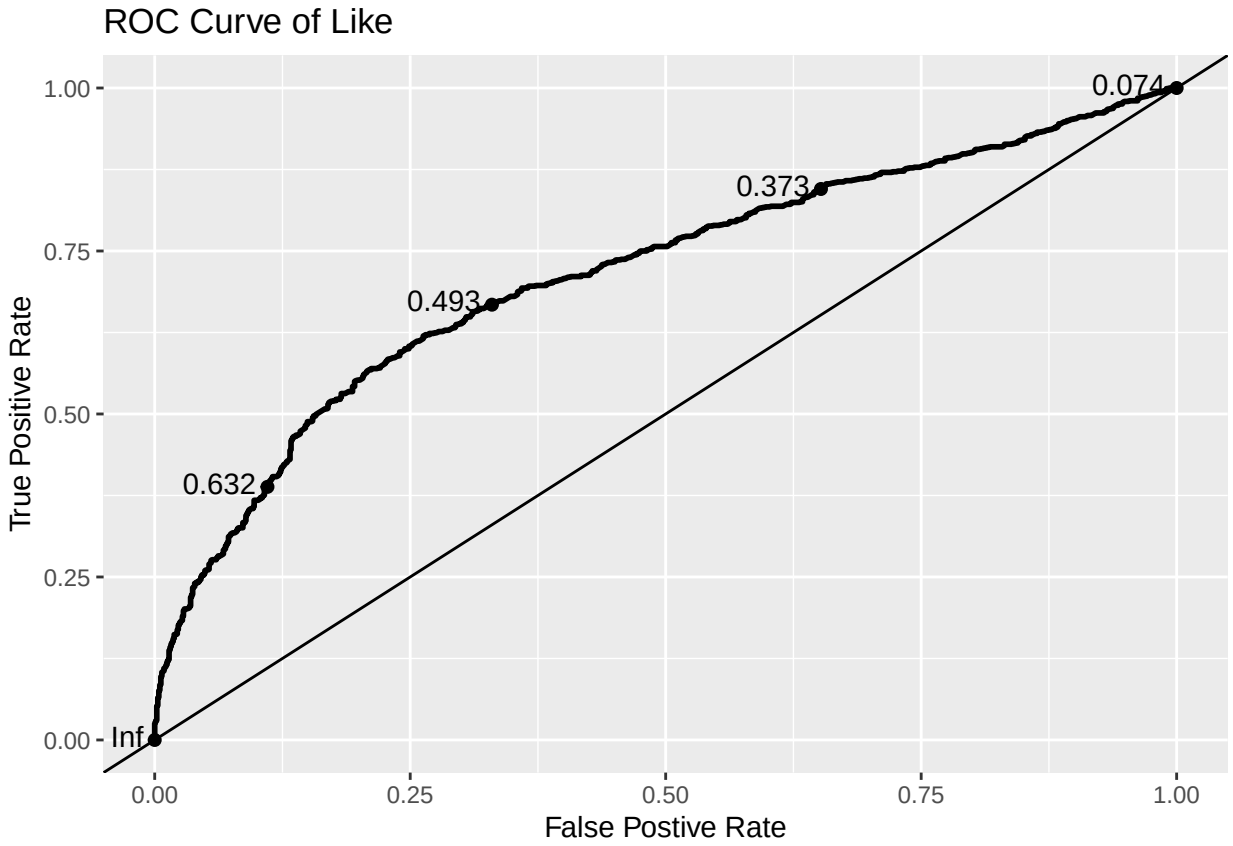
```
## # A tibble: 3 x 2
##   key   mean_resid
##   <fct>      <dbl>
## 1 D      -5.42e-16
## 2 D#     -6.66e-16
## 3 Other  -6.20e-16
```

Based on the plots from exercises 6-7, it seems as though the linearity assumption is satisfied. This is because there is no apparent shape in the plots of the predicted values versus the residuals, nor do the observations follow any sorts of linear lines. Additionally, the mean residuals in exercise 8 are all very close to 0, indicating that the linearity assumption is satisfied for `key`.

ROC and AUC

ROC curve and calculating AUC:

```
(roc_curve <- ggplot(logit_aug,
                     aes(d = as.numeric(like) - 1,
                         m = .fitted)) +
  geom_roc(n.cuts = 5, labelround = 3) +
  geom_abline(intercept = 0) +
  labs(title = "ROC Curve of Like",
       x = "False Postive Rate",
       y = "True Positive Rate") )
```




```
calc_auc(roc_curve)$AUC
```

```
## [1] 0.7137869
```

As seen above, the area under the curve for the ROC curve is approximately 0.714.

Interpretation

Based on the ROC curve and AUC from the previous exercise, it seems as though this model does about an average job of differentiating between songs the user likes and doesn't like. An AUC score closer to 1 is typically a better fit (whereas 0.5 means a very bad fit), and the AUC score of this particular curve was 0.714 which was almost exactly in the middle of the spectrum.

Choosing a Threshold

The threshold value we chose was 0.5. This is because, as we examined the ROC curve, we thought a threshold value of around 0.5 would be the best compromise for a high true positive rate and low false positive rate. After a threshold value of 0.5, the slope of the ROC curve seems to decrease, meaning a lower increase in the true positive rate compared to a greater increase in the false positive rate.

Keeping in mind Spotify's goal of recommending songs a user has a high liklihood of liking, we decided that a threshold of 0.5 would recommend songs that a user has a pretty high probability of liking, without annoying the user by recommending a bunch of songs they don't like.

Creating Confusion Matrix

```
threshold <- 0.5
logit_aug %>%
  mutate(like_predict = if_else(.fitted > threshold, "Yes", "No")) %>%
  group_by(like, like_predict) %>%
  summarise(n = n()) %>%
  kable(format="markdown")
```

like	like_predict	n
No	No	687
No	Yes	310
Yes	No	352
Yes	Yes	668

Misclassification Rates

The proportion of true positives is:

$$668/(668+352) \approx 0.655$$

The proportion of false positives is:

$$1 - 687/(687+310) \approx 0.311$$

The misclassification rate is:

$$(310+352)/(687+310+352+668) \approx 0.328$$