lab07

Dylan Scoble

3/4/2022

```r
spotify <- read_csv("spotify.csv")
```

```
## New names:
## * `` -> ...1
```

```
## Rows: 2017 Columns: 17
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): song_title, artist
## dbl (15): ...1, acousticness, danceability, duration_ms, energy, instrumenta...
```

```
##
## 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.
```

```r
glimpse(spotify)
```

```
## Rows: 2,017
## Columns: 17
## $ ...1             <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,~
## $ acousticness     <dbl> 0.010200, 0.199000, 0.034400, 0.604000, 0.180000, 0.0~
## $ danceability     <dbl> 0.833, 0.743, 0.838, 0.494, 0.678, 0.804, 0.739, 0.26~
## $ duration_ms      <dbl> 204600, 326933, 185707, 199413, 392893, 251333, 24140~
## $ energy           <dbl> 0.434, 0.359, 0.412, 0.338, 0.561, 0.560, 0.472, 0.34~
## $ instrumentalness <dbl> 2.19e-02, 6.11e-03, 2.34e-04, 5.10e-01, 5.12e-01, 0.0~
## $ key              <dbl> 2, 1, 2, 5, 5, 8, 1, 10, 11, 7, 5, 10, 0, 0, 9, 6, 1,~
## $ liveness         <dbl> 0.1650, 0.1370, 0.1590, 0.0922, 0.4390, 0.1640, 0.207~
## $ loudness         <dbl> -8.795, -10.401, -7.148, -15.236, -11.648, -6.682, -1~
## $ mode             <dbl> 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,~
## $ speechiness      <dbl> 0.4310, 0.0794, 0.2890, 0.0261, 0.0694, 0.1850, 0.156~
## $ tempo            <dbl> 150.062, 160.083, 75.044, 86.468, 174.004, 85.023, 80~
## $ time_signature   <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4,~
## $ valence          <dbl> 0.286, 0.588, 0.173, 0.230, 0.904, 0.264, 0.308, 0.39~
## $ target           <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ song_title       <chr> "Mask Off", "Redbone", "Xanny Family", "Master Of Non~
## $ artist           <chr> "Future", "Childish Gambino", "Future", "Beach House"~
```

Part 1: Data Prep & Modeling

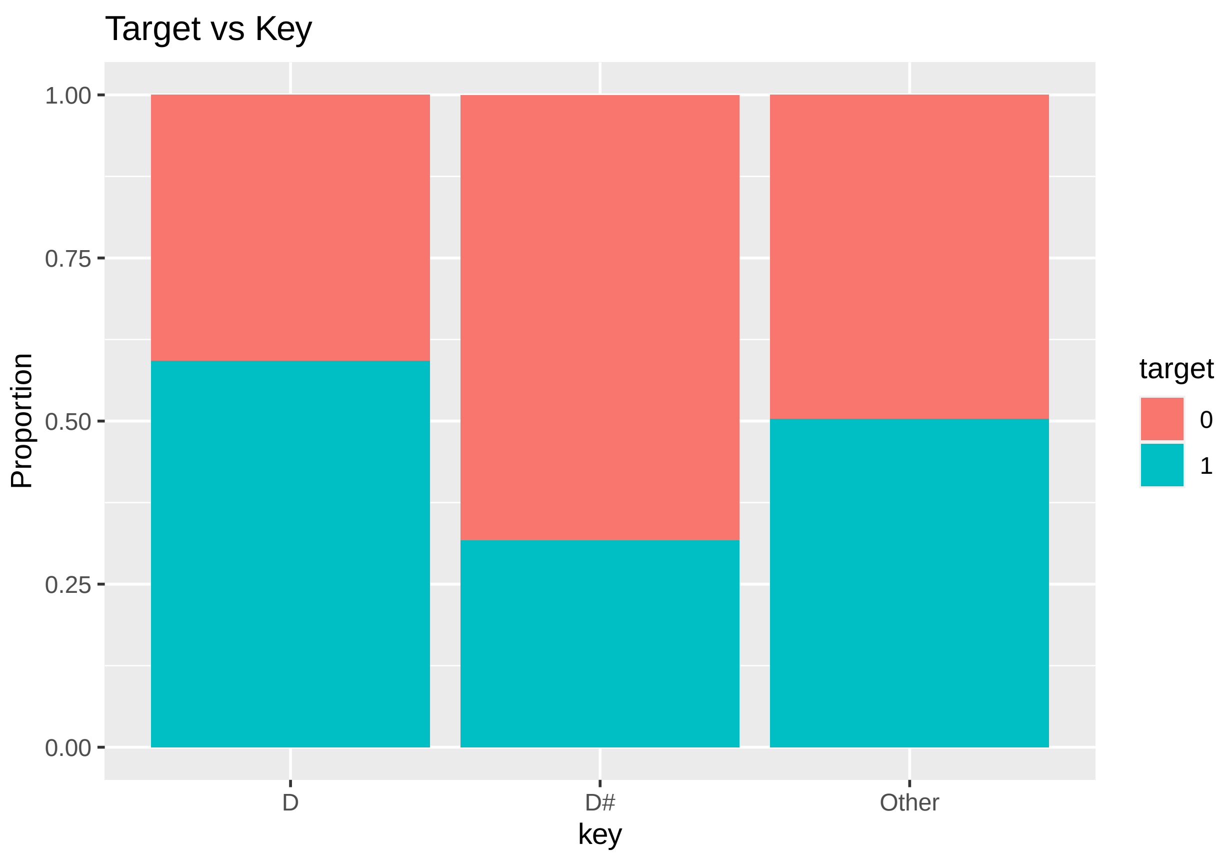
Exercise 1

```
spotify <- spotify %>%
  drop_na() %>%
  mutate(target = as.factor(target),
         key = case_when(
           key == 2 ~ "D",
           key == 3 ~ "D#",
           TRUE ~ "Other"
         ))

glimpse(spotify)
```

```
## Rows: 2,017
## Columns: 17
## $ ...1             <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,~
## $ acousticness     <dbl> 0.010200, 0.199000, 0.034400, 0.604000, 0.180000, 0.0~
## $ danceability     <dbl> 0.833, 0.743, 0.838, 0.494, 0.678, 0.804, 0.739, 0.26~
## $ duration_ms      <dbl> 204600, 326933, 185707, 199413, 392893, 251333, 24140~
## $ energy           <dbl> 0.434, 0.359, 0.412, 0.338, 0.561, 0.560, 0.472, 0.34~
## $ instrumentalness <dbl> 2.19e-02, 6.11e-03, 2.34e-04, 5.10e-01, 5.12e-01, 0.0~
## $ key              <chr> "D", "Other", "D", "Other", "Other", "Other", "Other"~
## $ liveness         <dbl> 0.1650, 0.1370, 0.1590, 0.0922, 0.4390, 0.1640, 0.207~
## $ loudness         <dbl> -8.795, -10.401, -7.148, -15.236, -11.648, -6.682, -1~
## $ mode             <dbl> 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,~
## $ speechiness      <dbl> 0.4310, 0.0794, 0.2890, 0.0261, 0.0694, 0.1850, 0.156~
## $ tempo            <dbl> 150.062, 160.083, 75.044, 86.468, 174.004, 85.023, 80~
## $ time_signature   <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4,~
## $ valence          <dbl> 0.286, 0.588, 0.173, 0.230, 0.904, 0.264, 0.308, 0.39~
## $ target           <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ song_title       <chr> "Mask Off", "Redbone", "Xanny Family", "Master Of Non~
## $ artist           <chr> "Future", "Childish Gambino", "Future", "Beach House"~
```

```
ggplot(data = spotify, aes(x = key, fill = target)) +
  geom_bar(position = "fill") +
  labs(title="Target vs Key",
       y="Proportion")
```



The plot above describes the relationship between Key and Target. For all songs in the key of D, about 60% of them have a target value of 1. For all songs in the key of D#, about 30% of them have a target value of 1. For all other songs, about half of them have a target value of 1.

Exercise 2

```
model <- glm(target ~ acousticness + danceability + duration_ms + instrumentalness + loudness + speechi

tidy(model, conf.int = TRUE, exponentiate = FALSE) %>%
  kable(format="markdown", digits = 5)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-2.95548	0.27640	-10.69288	0.00000	-3.50377	-2.41978
acousticness	-1.72231	0.23982	-7.18155	0.00000	-2.19743	-1.25678
danceability	1.63040	0.34421	4.73664	0.00000	0.95847	2.30841
duration_ms	0.00000	0.00000	4.22500	0.00002	0.00000	0.00000
instrumentalness	1.35291	0.20659	6.54883	0.00000	0.95236	1.76298
loudness	-0.08744	0.01727	-5.06219	0.00000	-0.12160	-0.05383
speechiness	4.07238	0.58302	6.98493	0.00000	2.94695	5.23420
valence	0.85638	0.22326	3.83573	0.00013	0.41997	1.29551

Exercise 3

```
model_with_key <- glm(target ~ acousticness + danceability + duration_ms + instrumentalness + loudness +

summary(model)$aic
```

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

```
summary(model_with_key)$aic
```

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

Adding key to the model lowers the model's AIC, making it better at predicting target. Therefore, we will continue to use the model with key.

Exercise 4

```
tidy(model_with_key, conf.int = TRUE, exponentiate = FALSE) %>%
  kable(format="markdown", digits = 5)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-2.50934	0.31102	-8.06817	0.00000	-3.12416	-1.90422
acousticness	-1.70210	0.24091	-7.06521	0.00000	-2.17930	-1.23436
danceability	1.64880	0.34536	4.77417	0.00000	0.97468	2.32911
duration_ms	0.00000	0.00000	4.18744	0.00003	0.00000	0.00000
instrumentalness	1.38316	0.20745	6.66741	0.00000	0.98104	1.79505
loudness	-0.08662	0.01726	-5.01848	0.00000	-0.12075	-0.05301
speechiness	4.03380	0.58491	6.89650	0.00000	2.90456	5.19917
valence	0.88094	0.22434	3.92682	0.00009	0.44248	1.32224
keyD#	-1.07319	0.33497	-3.20385	0.00136	-1.74517	-0.42805
keyOther	-0.49390	0.16898	-2.92288	0.00347	-0.82793	-0.16468

If the key of the observation is D#, the target value's prediction will increase, by a value of exp(-1.07319).

Part 2: Checking Assumptions

Exercise 5

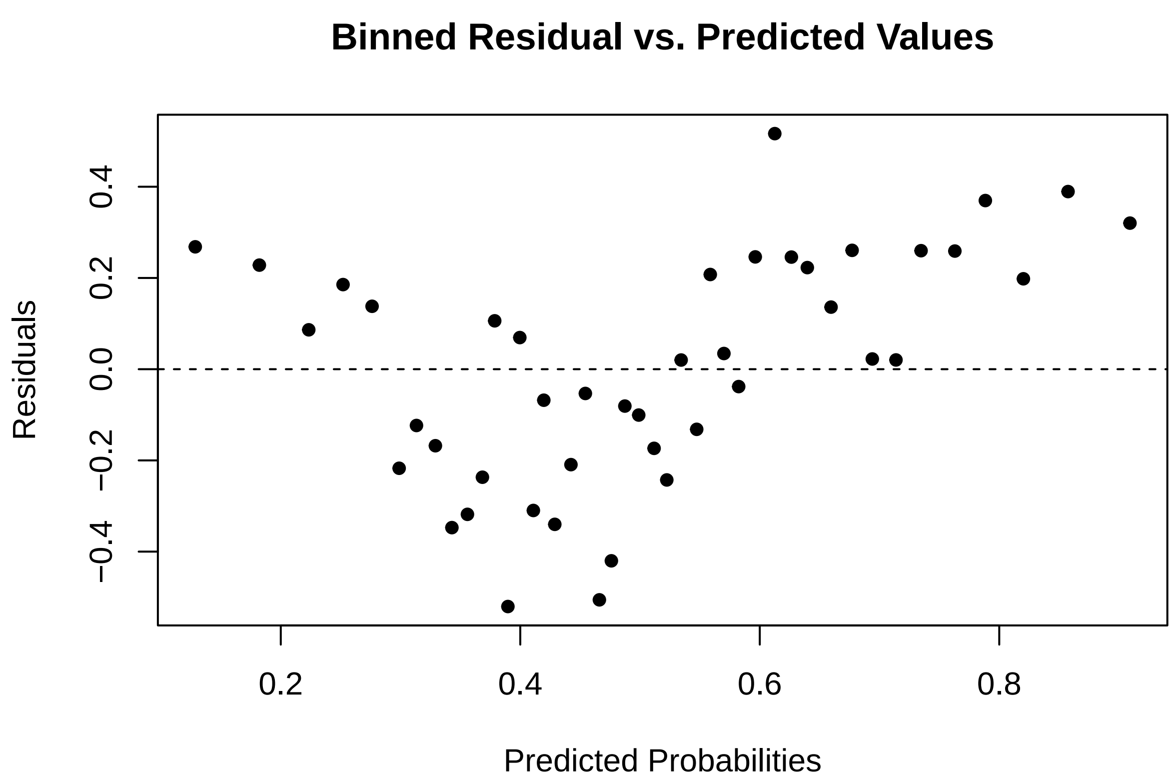
```
df <- augment(model_with_key, type.predict = "response",type.residuals = "deviance")
glimpse(df)
```

```
## Rows: 2,017
## Columns: 15
## $ target           <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
```

```
## $ acousticness     <dbl> 0.010200, 0.199000, 0.034400, 0.604000, 0.180000, 0.0~
## $ danceability     <dbl> 0.833, 0.743, 0.838, 0.494, 0.678, 0.804, 0.739, 0.26~
## $ duration_ms      <dbl> 204600, 326933, 185707, 199413, 392893, 251333, 24140~
## $ instrumentalness <dbl> 2.19e-02, 6.11e-03, 2.34e-04, 5.10e-01, 5.12e-01, 0.0~
## $ loudness         <dbl> -8.795, -10.401, -7.148, -15.236, -11.648, -6.682, -1~
## $ speechiness      <dbl> 0.4310, 0.0794, 0.2890, 0.0261, 0.0694, 0.1850, 0.156~
## $ valence          <dbl> 0.286, 0.588, 0.173, 0.230, 0.904, 0.264, 0.308, 0.39~
## $ key              <chr> "D", "Other", "D", "Other", "Other", "Other", "Other"~
## $ .fitted          <dbl> 0.9015924, 0.6379788, 0.7829667, 0.4223972, 0.8489462~
## $ .resid           <dbl> 0.4551764, 0.9481036, 0.6995214, 1.3128664, 0.5722926~
## $ .std.resid       <dbl> 0.4567745, 0.9493120, 0.7028629, 1.3163187, 0.5733889~
## $ .hat             <dbl> 0.006985200, 0.002544191, 0.009485687, 0.005238444, 0~
## $ .sigma           <dbl> 1.117466, 1.117311, 1.117402, 1.117126, 1.117439, 1.1~
## $ .cooksd          <dbl> 7.731894e-05, 1.451076e-04, 2.679972e-04, 7.238900e-0~
```

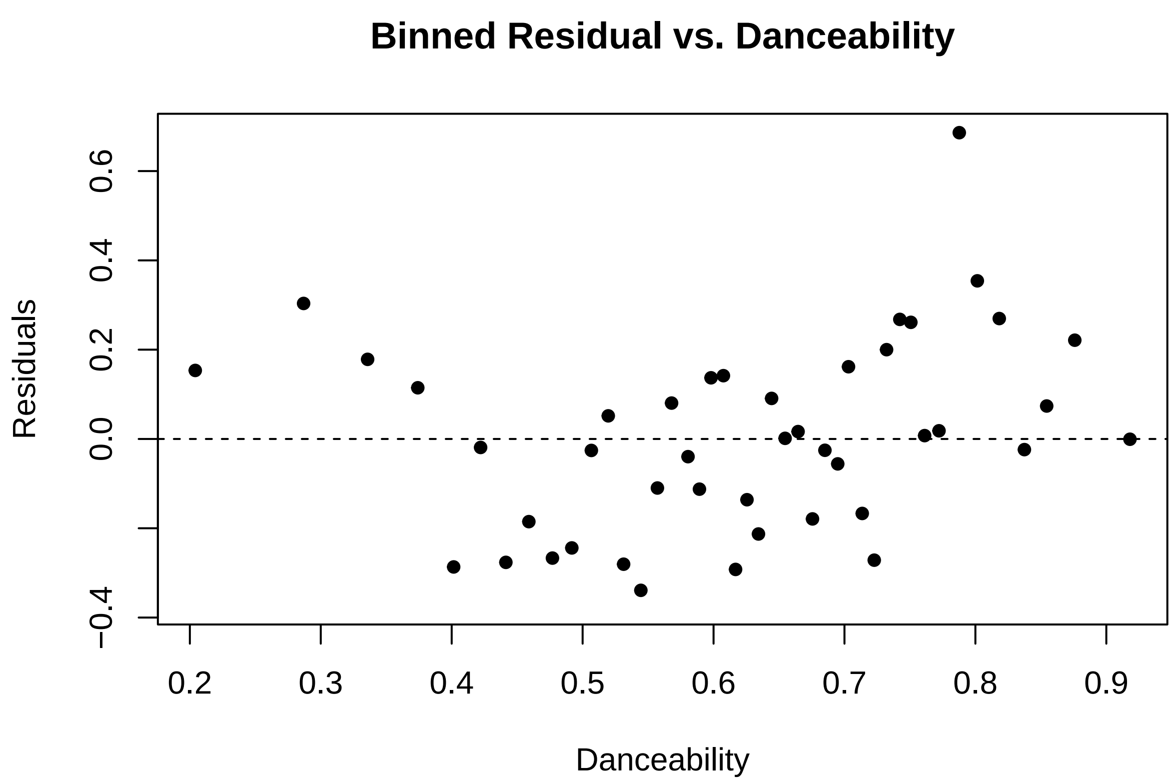
Exercise 6

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



Exercise 7

```
arm::binnedplot(x = df$danceability, y = df$.resid,
                xlab = "Danceability",
                ylab = "Residuals",
                main = "Binned Residual vs. Danceability",
                col.int = FALSE)
```



Exercise 8

```
df %>%
  mutate(key_resid = if_else(.resid > 1 | .resid < -1, "High Residual", "Low Residual")) %>%
  group_by(key, key_resid) %>%
  summarise(n = n()) %>%
  kable(format="markdown")
```

```
## `summarise()` has grouped output by 'key'. You can override using the `.groups` argument.
```

key	key_resid	n
D	High Residual	98

key	key_resid	n
D	Low Residual	86
D#	High Residual	21
D#	Low Residual	42
Other	High Residual	1039
Other	Low Residual	731

Exercise 9

The linearity assumption is not satisfied because there is no clear linear relationship between our fitted values and our predictors, as shown by the plot in exercise 6.

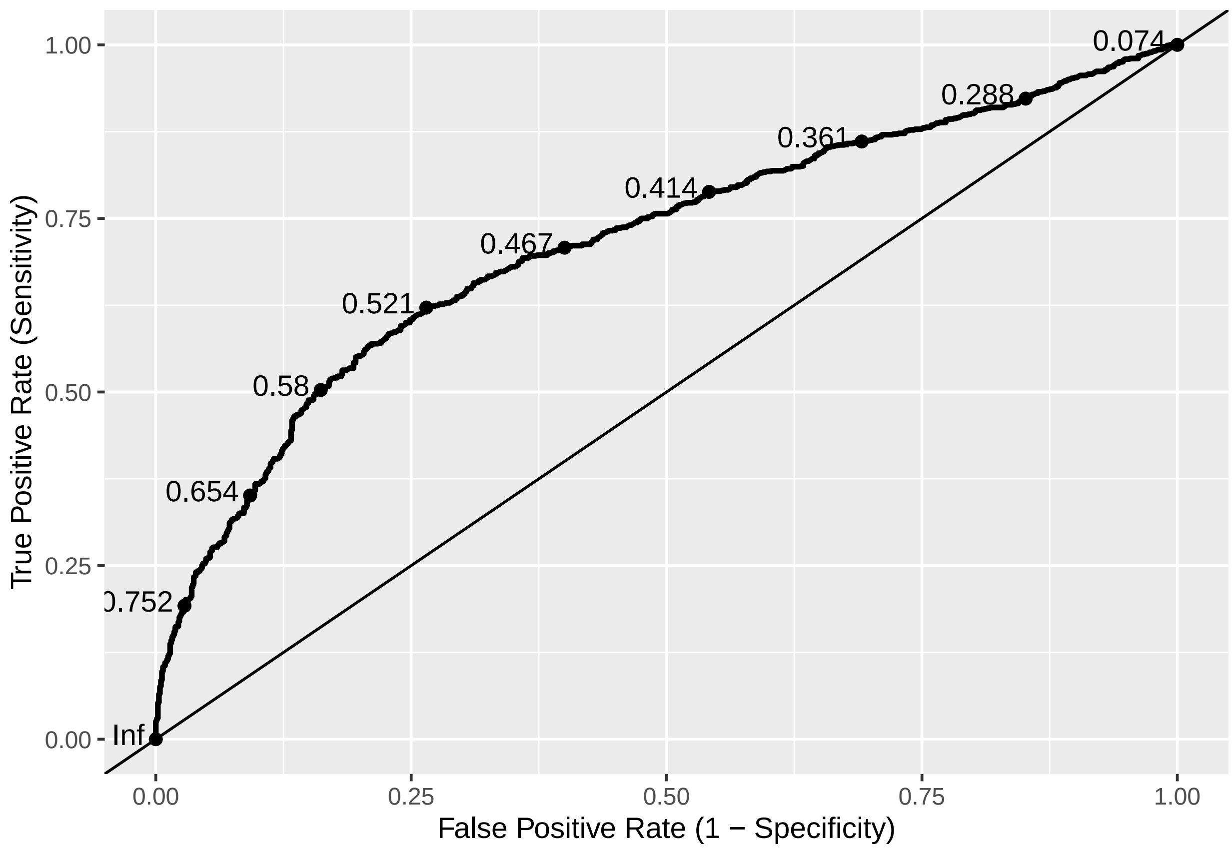
Part 3: Model Assessment & Prediction

Exercise 10

```
roc_curve <- ggplot(df, aes(d = as.numeric(target), m = .fitted)) +
  geom_roc(n.cuts = 10, labelround = 3) +
  geom_abline(intercept = 0) +
  labs(x = "False Positive Rate (1 - Specificity)",
       y = "True Positive Rate (Sensitivity)")

roc_curve
```

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming 1 = 0 and 2 = 1!
```




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

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming 1 = 0 and 2 = 1!
```

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

Exercise 11

This model does effectively differentiate between the songs the user likes versus those he or she doesn't. The ROC Curve is entirely above the line x=y, which represents an entirely random classifier. Additionally, the area under the ROC curve exceeds 0.5.

Exercise 12.

I would choose a threshold of 0.58 because it maximizes the TPR (true positive rate) while keeping FPR (false positive rate) as small as possible. In terms of the ROC curve, the 0.58 threshold seems to have the largest euclidean distance from the line x=y.

Exercise 13

```r
threshold = 0.58
df %>%
  mutate(prediction = if_else(.fitted < threshold, "0:No","1:Yes")) %>%
  group_by(target, prediction) %>%
  summarise(n = n()) %>%
  kable(format="markdown")
```

```
## `summarise()` has grouped output by 'target'. You can override using the `.groups` argument.
```

target	prediction	n
0	0:No	836
0	1:Yes	161
1	0:No	508
1	1:Yes	512

Exercise 14

The proportion of true positives is $(512/512 + 508) = 0.502$

The proportion of false positives is $(508/512 + 508) = 0.498$

The missclassification rate is $(508 + 161/508 + 161 + 512 + 836) = 0.302$