STAT 108: Lab 7

Tim Lanthier

3/2/2022

Lab 7: Logistic Regression

Github: https://github.com/talanthier/lab-07

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

In this lab we will be investigating Spotify song attribute data from 2017 from a single user. The raw dataset can be found on the Github Repository.

```r
spotify <- read.csv('data/spotify.csv')
glimpse(spotify)
```

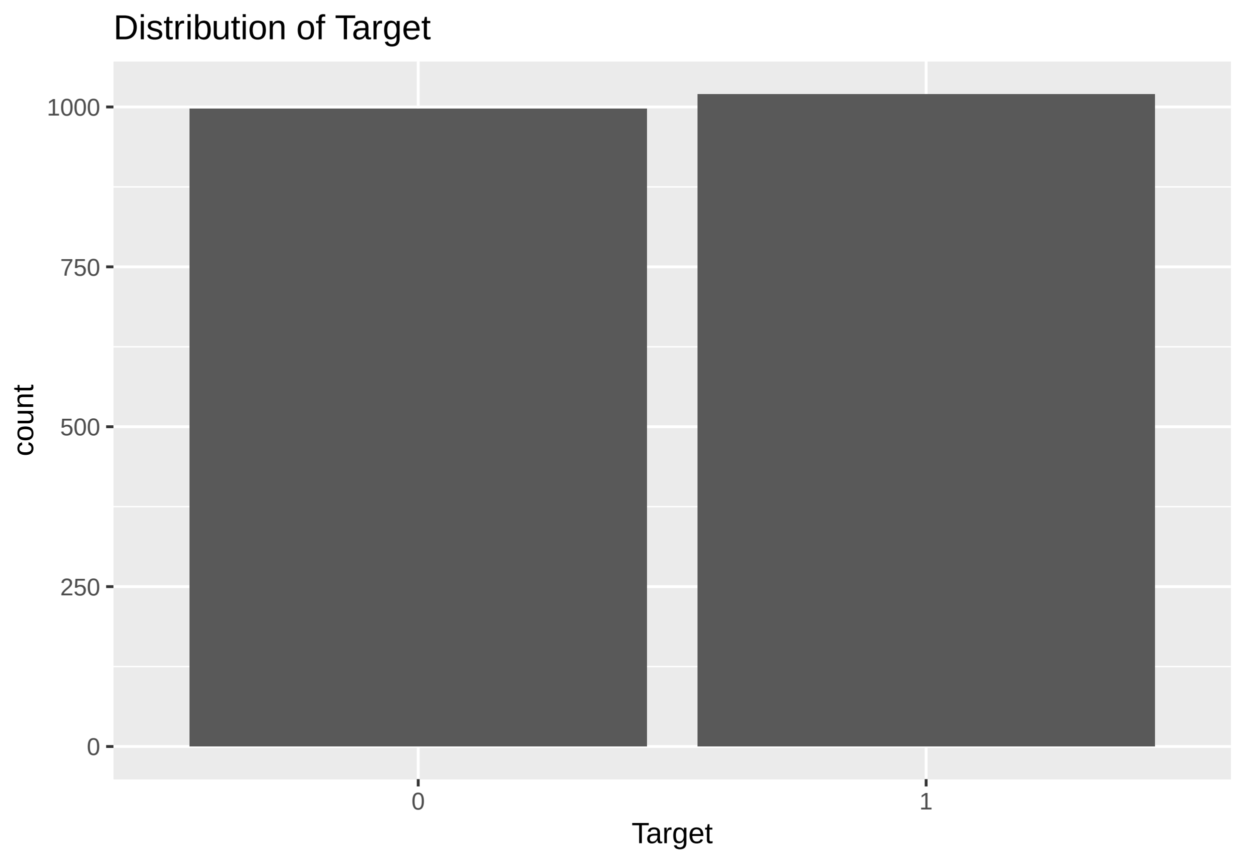
```
## Rows: 2,017
## Columns: 17
## $ X                <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,~
## $ acousticness     <dbl> 0.010200, 0.199000, 0.034400, 0.604000, 0.180000, 0.0~
## $ danceability     <dbl> 0.833, 0.743, 0.838, 0.494, 0.678, 0.804, 0.739, 0.26~
## $ duration_ms      <int> 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              <int> 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             <int> 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           <int> 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"~
```

Data Prep & Modeling

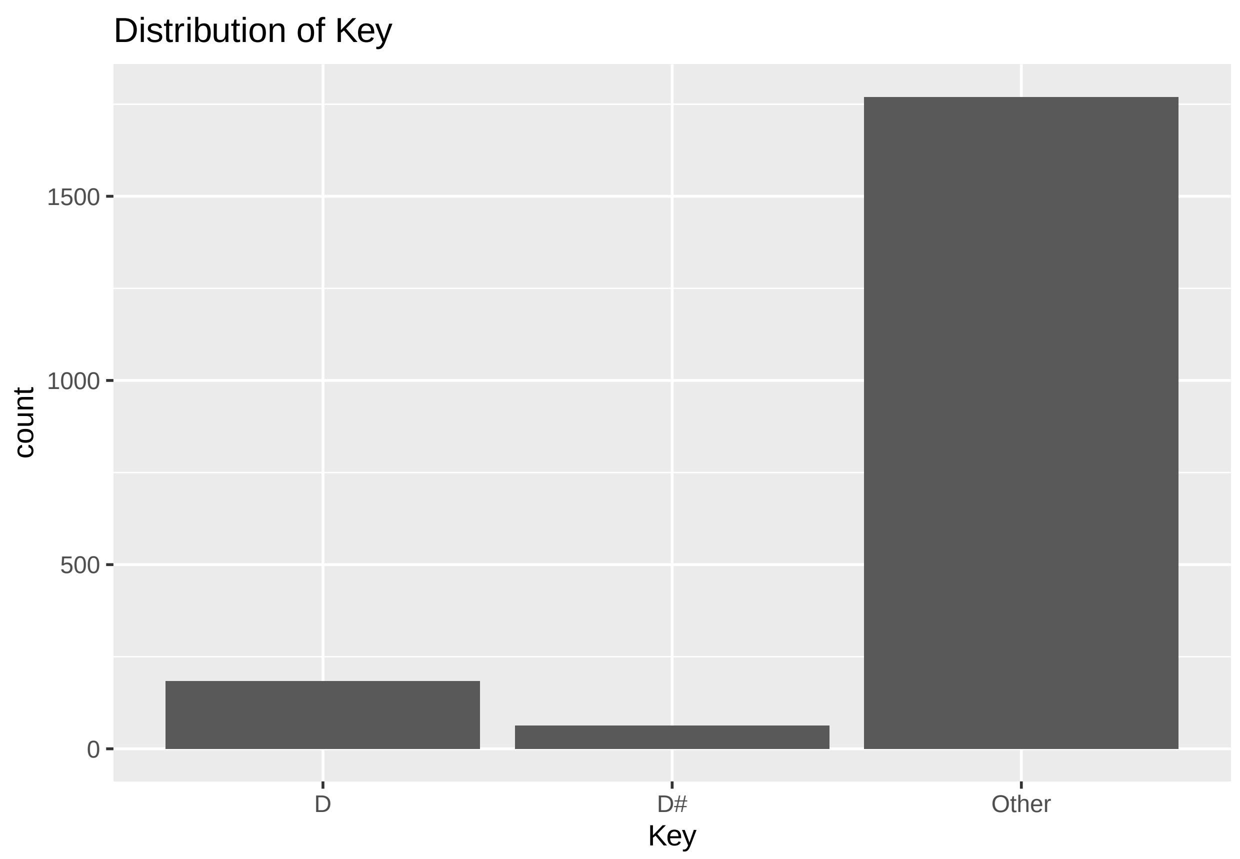
We will be creating a model to predict whether or not the user likes a song or not. Here, the response variable is `target` where 1 means the user liked the song, 0 means they did not like the song. Note that in the glimpse above, `target` is an integer value which we need to change to a factor. We also will need to change `key` to a factor.

```r
spotify <- spotify %>%
  mutate(target = as.factor(target),
         key = ifelse(key == 2, 'D', ifelse(key == 3, 'D#', 'Other')))
```

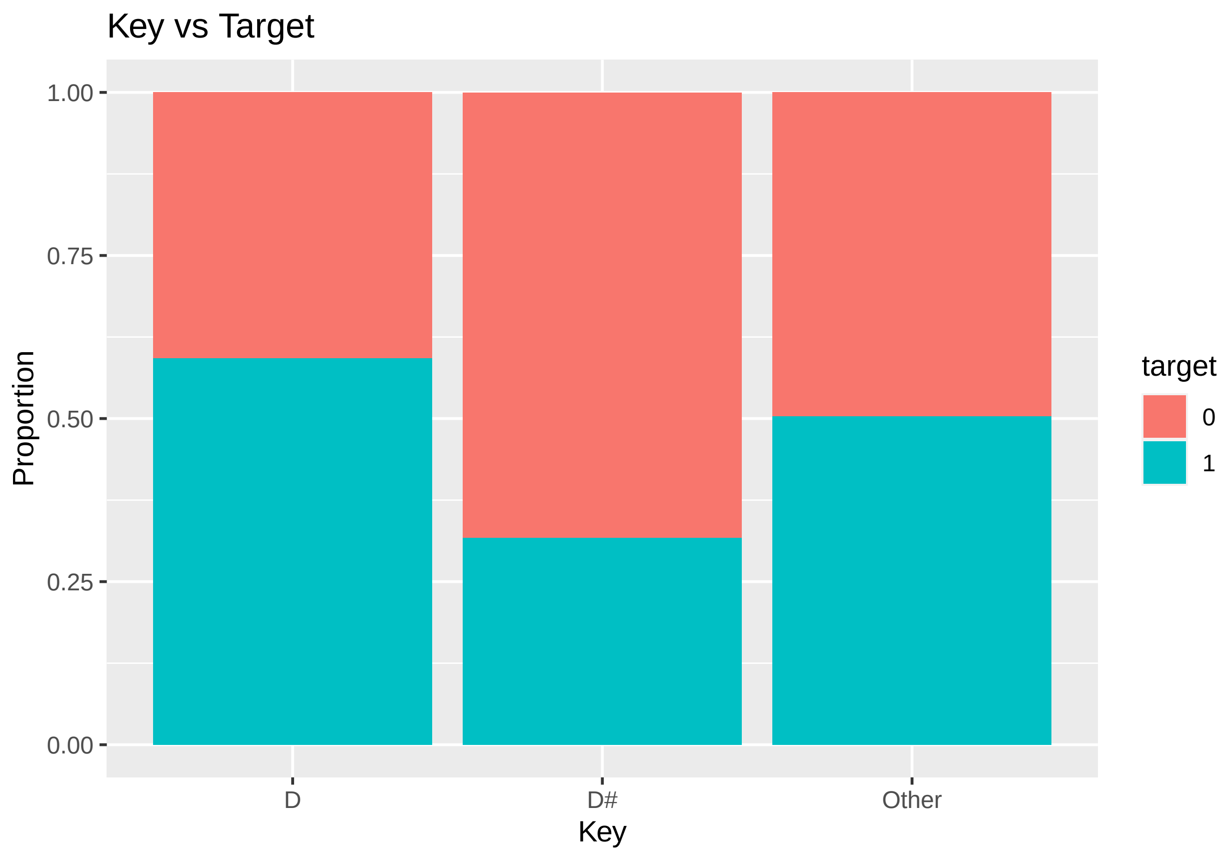
```r
ggplot(data = spotify, aes(x = target)) +
  geom_bar() +
  labs(x = 'Target', title = 'Distribution of Target')
```



```r
ggplot(data = spotify, aes(x = key)) +
  geom_bar() +
  labs(x = 'Key', title = 'Distribution of Key')
```



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



According to the above plot, it looks like the user appears to dislike songs in D# shown by a approximately 30% of the songs in D# being liked by the user. Meanwhile the user likes approximately 60% of the songs in D and around 50% of the songs in other keys. Now we will build a logistic regression model to predict `target` using `acousticness`, `danceability`, `duration_ms`, `instrumentalness`, `loudness`, `speechiness`, and `valence` as our predictors.

```
model <- glm(target ~ acousticness + danceability + duration_ms + instrumentalness + loudness + speechi
             data = spotify,
             family = 'binomial')
tidy(model, conf.int=TRUE) %>%
  kable(digits = 6, format = 'markdown')
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-2.955475	0.276397	-10.692879	0.000000	-3.503773	-2.419782
acousticness	-1.722313	0.239825	-7.181553	0.000000	-2.197434	-1.256776
danceability	1.630404	0.344211	4.736643	0.000002	0.958471	2.308410
duration_ms	0.000003	0.000001	4.225004	0.000024	0.000002	0.000004
instrumentalness	1.352910	0.206588	6.548829	0.000000	0.952356	1.762979
loudness	-0.087437	0.017273	-5.062191	0.000000	-0.121603	-0.053827
speechiness	4.072379	0.583023	6.984931	0.000000	2.946953	5.234202
valence	0.856382	0.223265	3.835727	0.000125	0.419966	1.295508

Now we will consider adding `key` to our model. Consider the model `model_key` shown below which is our original model but including `key` as an additional predictor.

```
model_key <- glm(target ~ key + acousticness + danceability + duration_ms + instrumentalness + loudness
             data = spotify,
             family = 'binomial')
```

We will now conduct a drop-in-deviance test between `model` and `model_key` to check whether we should include `key` in our final model. Note that in `model_key`, we have an additional 2 terms: `keyD#` and `keyOther`. So if we let β_1 and β_2 be the coefficients for `keyD#` and `keyOther` in our model including `key`, then we have the null hypothesis that $\beta_1 = \beta_2 = 0$ and the alternate hypothesis that at least one of β_1 and β_2 is nonzero. Now we may conduct a drop-in-deviance test.

```
dev_model <- glance(model)$deviance
dev_model_key <- glance(model_key)$deviance
test_stat <- dev_model - dev_model_key
```

Now as we noted earlier, we have an additional 2 parameters in `model_key` compared to the original model. So we will be checking the probability that χ^2 is greater than our test statistic where χ^2 follows a chi^2-distribution with 2 degrees of freedom.

```
1-pchisq(test_stat,2)
```

```
## [1] 0.001257663
```

The p-value from our drop-in-deviance test is shown. So at the 0.01 significance level, since our p-value is less than 0.01, we have sufficient evidence to reject the null hypothesis that $\beta_1 = \beta_2 = 0$. So we are confident that at least one of β_1 and β_2 is nonzero. This means that `key` will have some predictive power in our model. Thus we should include `key` in our model. So for the remainder of the lab, we will be using `model_key` shown below.

```
tidy(model_key, conf.int = TRUE) %>%
  kable(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
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
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

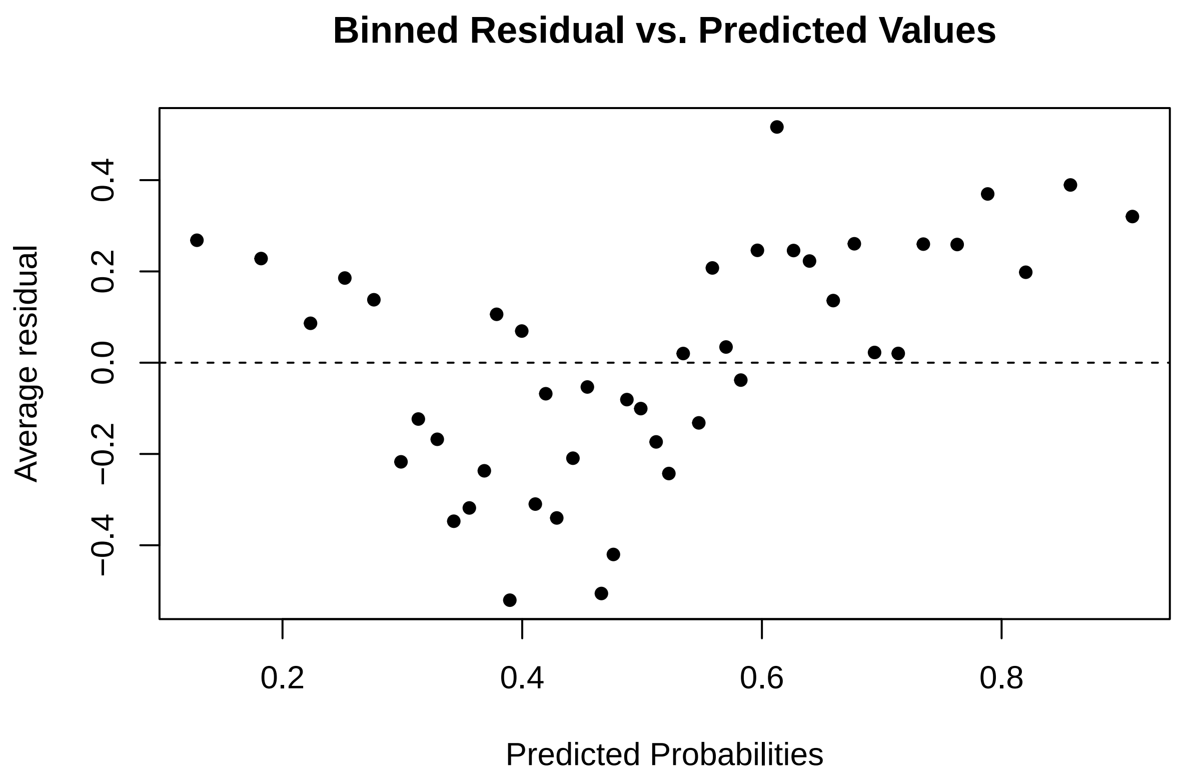
According to the model output, looking at the coefficient for `keyD#`, a song will have a log odds of being liked by the user that is $e^{-1.073} = 0.342$ times the log odds of a song with the same characteristics with the only difference that it is in the D key.

Checking Assumptions

We will now check the assumptions for our model. We will start by checking the binned residuals and the predicted probabilities for our model.

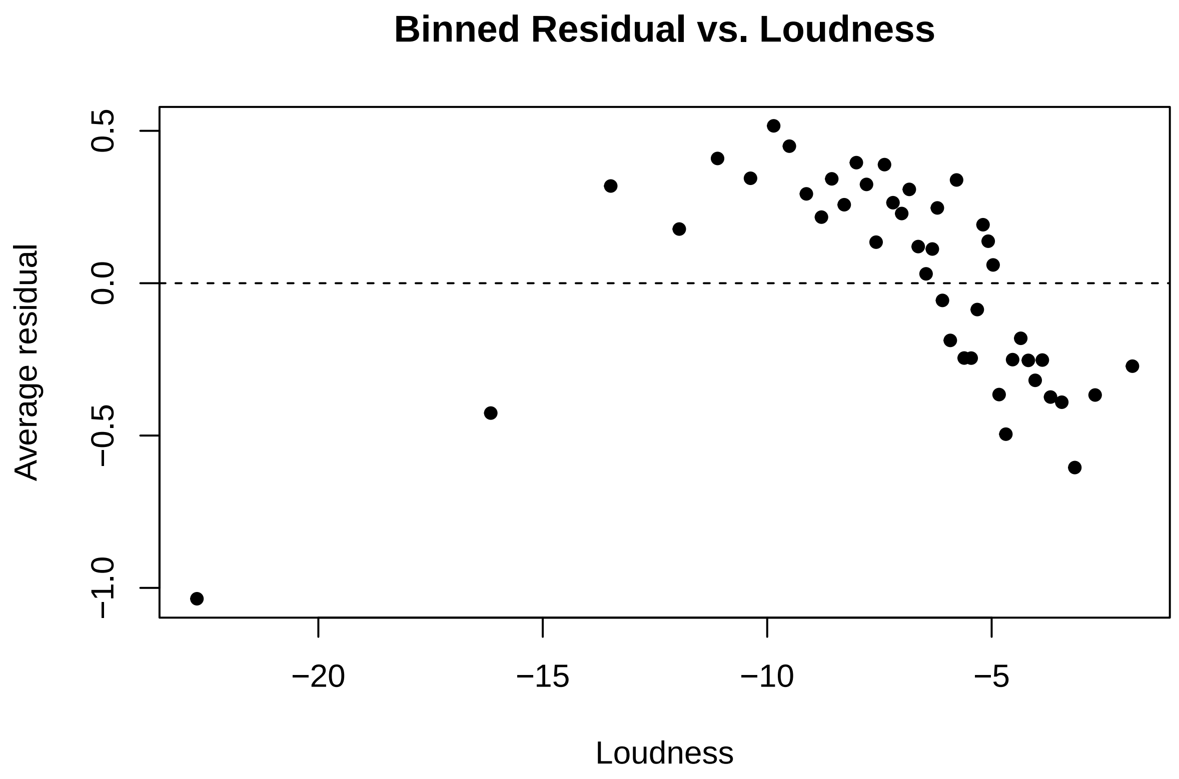
```
spotify_aug <- augment(model_key, type.predict = 'response')

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



We will also examine the binned residual plots for `loudness`.

```
arm::binnedplot(x = spotify_aug$loudness, y = spotify_aug$.resid,
                xlab = "Loudness",
                main = "Binned Residual vs. Loudness",
                col.int = FALSE)
```



Now consider the residuals in each category of `key`.

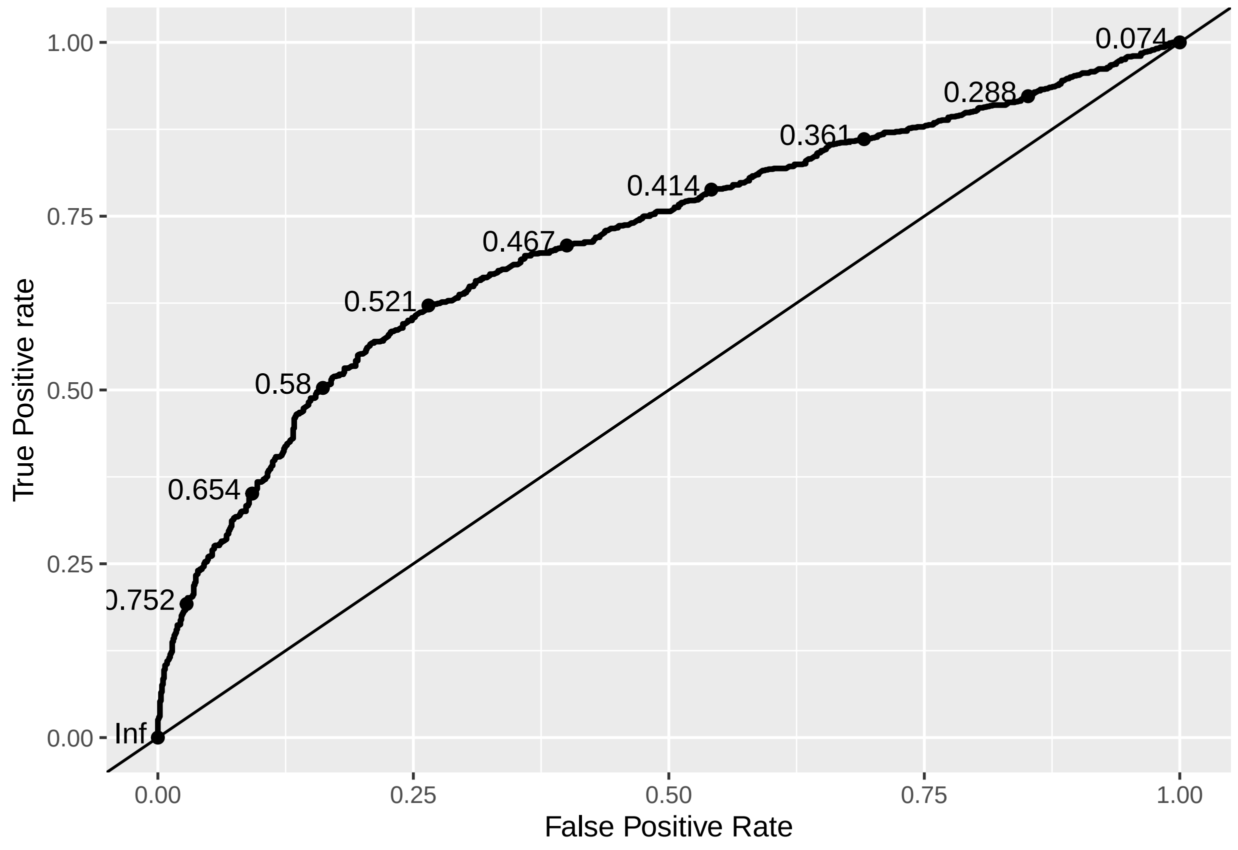
```r
spotify_aug %>%
  group_by(key) %>%
  summarise(mean_resid = mean(.resid))
```

```
## # A tibble: 3 x 2
##   key   mean_resid
##   <chr>      <dbl>
## 1 D        0.0542
## 2 D#      -0.0992
## 3 Other    0.00316
```

Now that we have the above plots and table, we can see that the linearity assumption is not satisfied. Starting with the binned residual plot of Binned residuals versus predicted probabilities, we see a clear slightly V-shaped pattern. Looking at the binned residual plot for `loudness`, we also see a clear pattern. Songs with either low or high loudness on average have residuals below zero. Since we do not have a random scatter in our binned residual plots, we conclude that the linearity assumption is not satisfied. That being said, the mean residuals across the groups for `key` are pretty close to one another (all within 0.1 of 0).

Model Assessment and Prediction

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



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

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

So we have an AUC of 0.714. Since an AUC of close to 1 would indicate a very good fit and an AUC close to 0.5 would indicate a poor fit, I would say that our model does just an okay job of differentiating between the songs the user likes and doesn't like. Looking at are ROC curve, it does not get particularly close to the ideal case where we have a True Positive rate very close to 1 and false positive rate very close to 0.

In this case, we are predicting whether the user likes or dislikes a song. In the context of this problem, a false positive means we classified a song which the user dislikes as a song we think he would like. A false negative would be when we classify a song that the user actually likes as a song that he should dislike. So in the context of this problem, neither are particularly dangerous. So we should choose a threshold value on the ROC curve closest fo the top left corner (high true positive rate, low false positive rate). Looking at our ROC curve, our optimal threshold might be between 0.58 and 0.521. So we might choose a threshold of 0.55.

For the rest of the lab we will use this threshold value of 0.55.

```
threshold <- 0.55

spotify_aug %>%
  mutate(target_pred = if_else(.fitted > threshold, "1: Liked", "0: Disliked")) %>%
  group_by(target, target_pred) %>%
  summarise(n = n()) %>%
  kable(format="markdown")
```

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

target	target_pred	n
0	0: Disliked	784
0	1: Liked	213
1	0: Disliked	439
1	1: Liked	581

The confusion matrix for our model with this threshold value is shown above. Hence we have a specificity of

$$\text{specificity} = \frac{581}{439 + 581} = 0.5696$$

and sensitivity of

$$\text{sensitivity} = \frac{784}{784 + 213} = 0.7864$$

Hence we get a false positive rate of

$$1 - \text{sensitivity} = 0.2136$$

In total, we misclassified $213 + 581 = 794$ songs out of a total of $784 + 213 + 439 + 581 = 2017$ songs. So we have a misclassification rate of $\frac{794}{2017} = 0.394$.