In Class Work: Classification

We'll be again working on a Kaggle-style competition to predict who gets pizza. Using the pizza dataset, find the best fit to the data. However, to qualify as a winner, you need to do the following:

1. Fit a model using logistic regression using the training dataset.
2. Compute the predictions from your model from the testing dataset.
3. Calculate the AUC (from the Receiver Operating Characteristic) for the predictions from your model from the testing dataset. Compare your results to the artice linked below.

```r
library(knitr)
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()
```

```r
library(modelr)
library(yardstick)
```

```
## For binary classification, the first factor level is assumed to be the event.
## Use the argument ‘event_level = "second"‘ to alter this as needed.

##
## Attaching package: 'yardstick'

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

## The following object is masked from 'package:readr':
##
##     spec
```

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

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x broom::bootstrap() masks modelr::bootstrap()
## 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::mae()   masks modelr::mae()
## x yardstick::mape()  masks modelr::mape()
## x yardstick::rmse()  masks modelr::rmse()
## x yardstick::spec()  masks readr::spec()
## x recipes::step()    masks stats::step()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```r
library(probably)
```

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

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

```r
load("za.RData")
```

```r
# Training and testing datasets

za_split<-initial_split(za,prop=.5)

za_train<-training(za_split)

za_test<-testing(za_split)
```

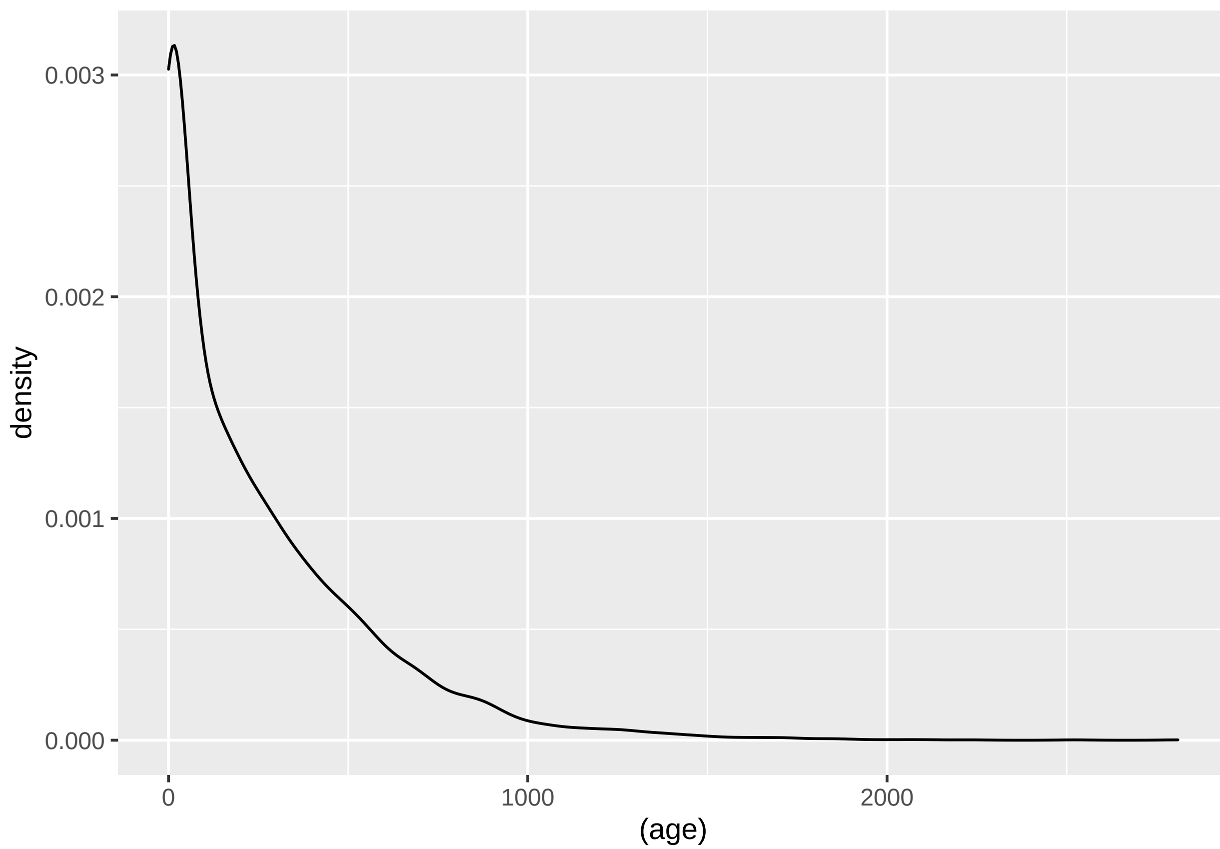
```r
colnames(za)
```

```
##  [1] "got_pizza"      "got_pizza_f"    "karma"          "age"
##  [5] "raop_age"       "pop_request"    "activity"       "total_posts"
##  [9] "raop_posts"     "prev_raop_post" "words"          "poor"
## [13] "student"        "grateful"       "score"
```

```
#  Model terms

za_formula<-as.formula("got_pizza_f~
             prev_raop_post+
             raop_posts+

             age+
             words+
             student+
             karma")



#table(za$got_pizza_f)
```

```
za%>%
  ggplot(aes(x=(age)))+
  geom_density()
```

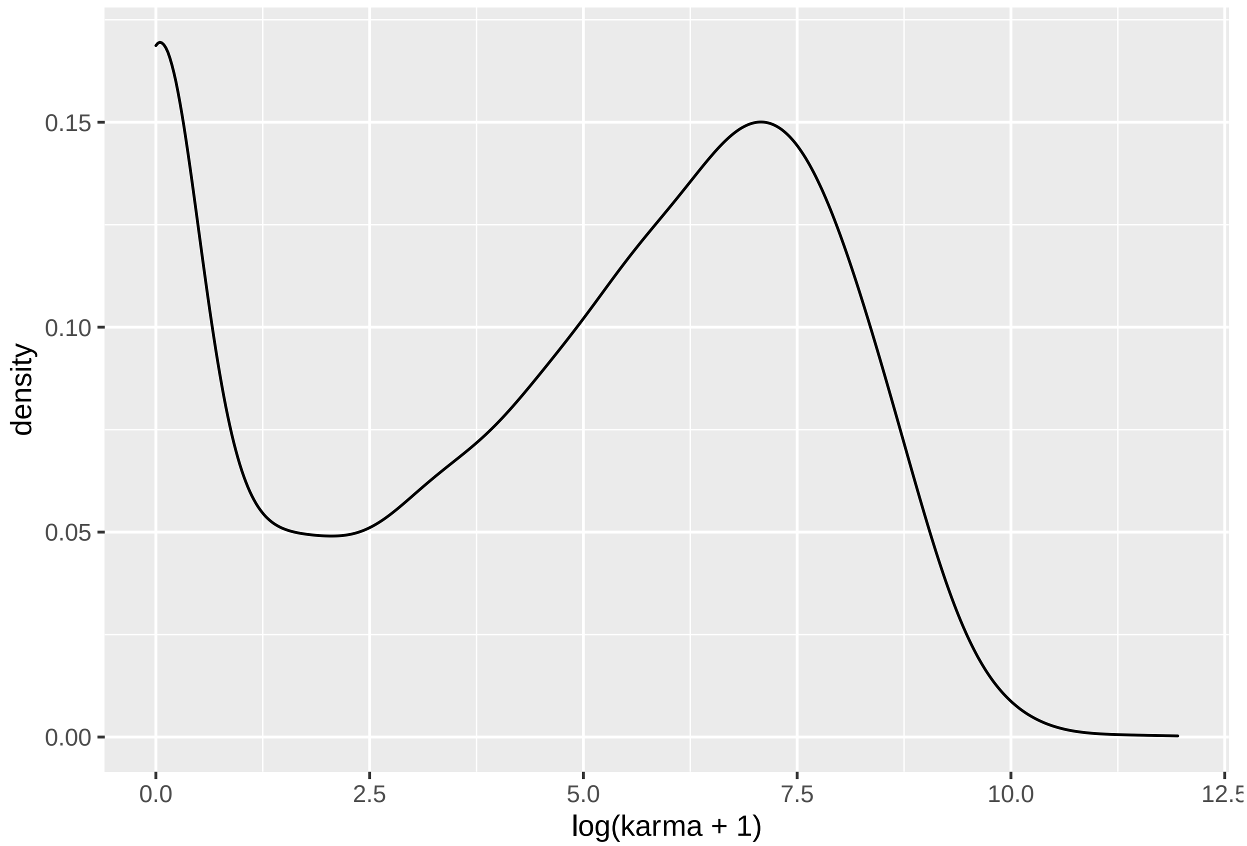
```
## Warning: Removed 2 rows containing non-finite values (stat_density).
```




```
#Do we consider new users? Do we log transform?

za%>%
```

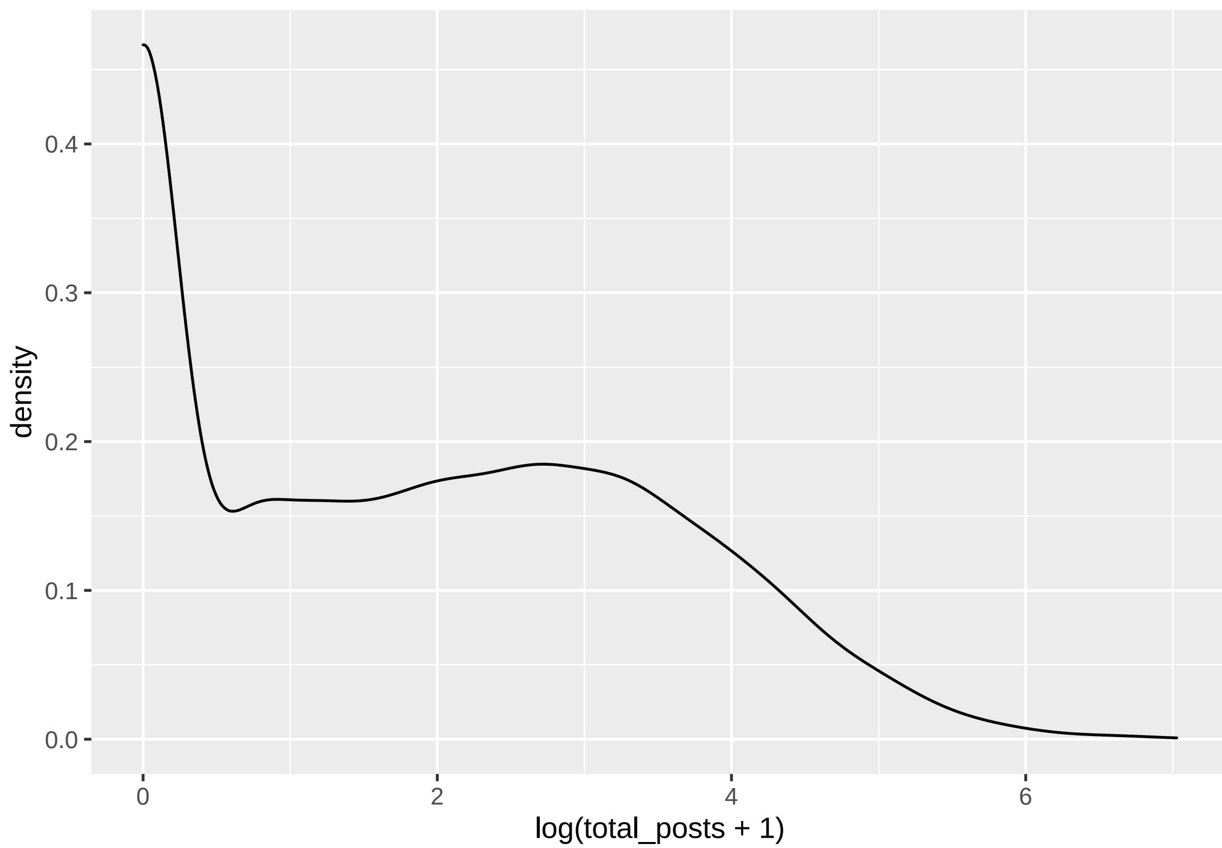
```r
  ggplot(aes(x=log(karma+1)))+
  geom_density()
```

```
## Warning in log(karma + 1): NaNs produced

## Warning in log(karma + 1): NaNs produced

## Warning: Removed 72 rows containing non-finite values (stat_density).
```



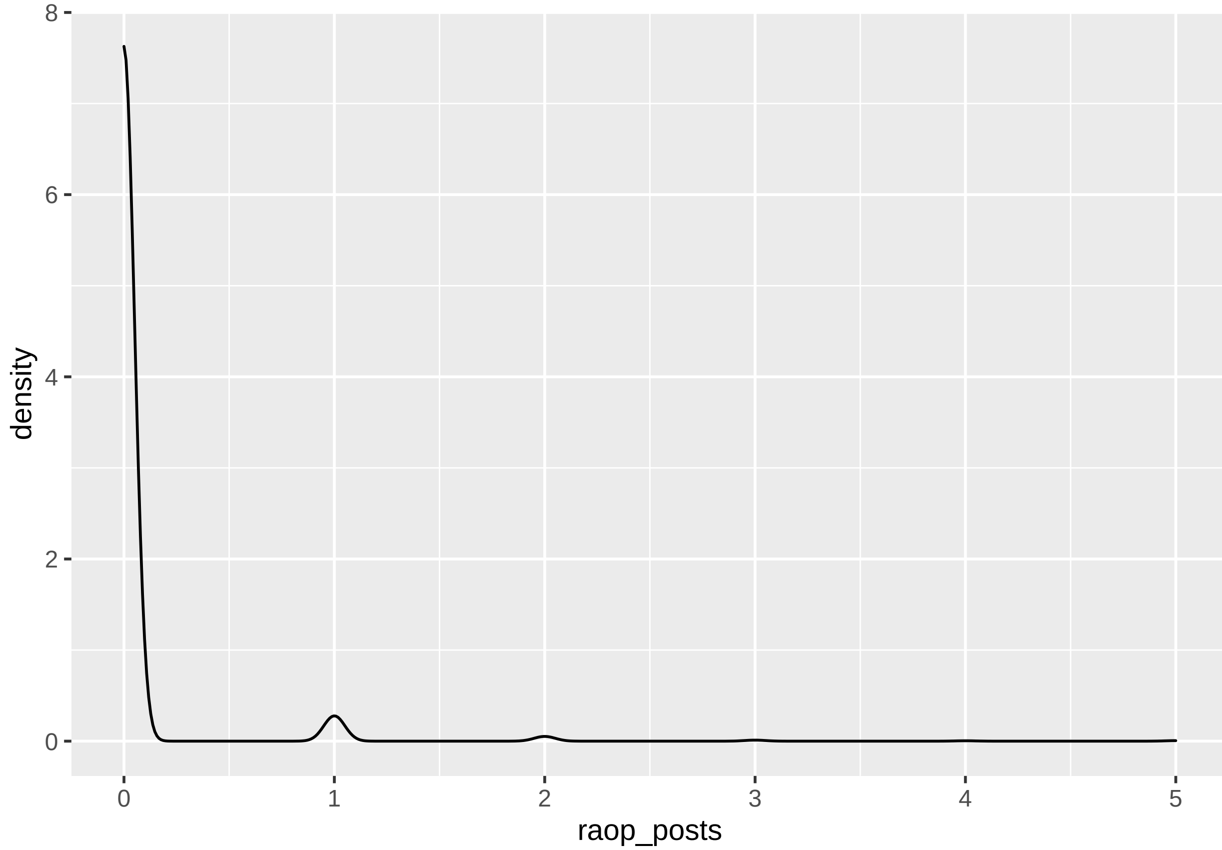

```r
#Do we consider new users? Do we log transform?

za%>%
  ggplot(aes(x=log(total_posts+1)))+
  geom_density()
```

```
## Warning: Removed 2 rows containing non-finite values (stat_density).
```




```
#Do we consider new users? Do we log transform?

za%>%
  ggplot(aes(x=raop_posts))+
  geom_density()
```

```
## Warning: Removed 3 rows containing non-finite values (stat_density).
```

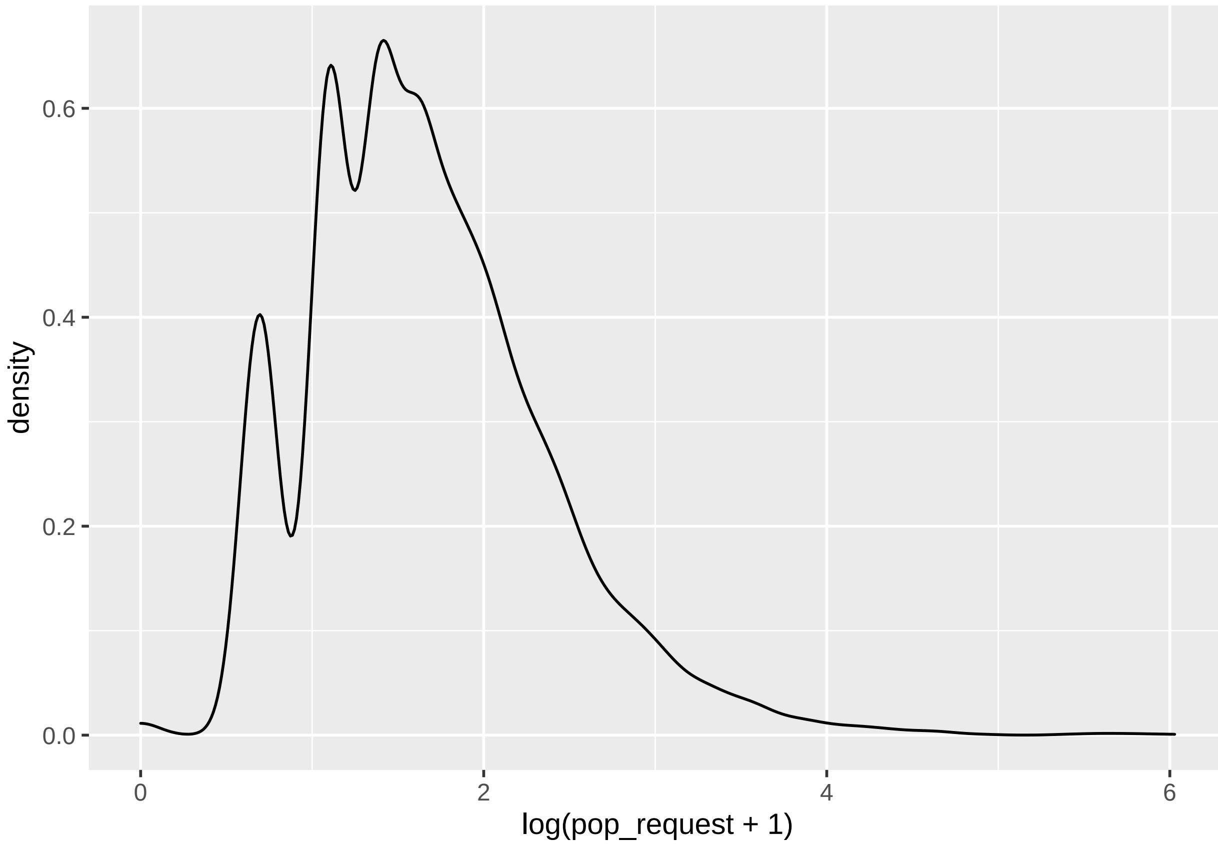



```r
table(za$raop_posts)
```

```
##
##    0    1    2    3    4    5
## 5420  198   37    8    3    4
```

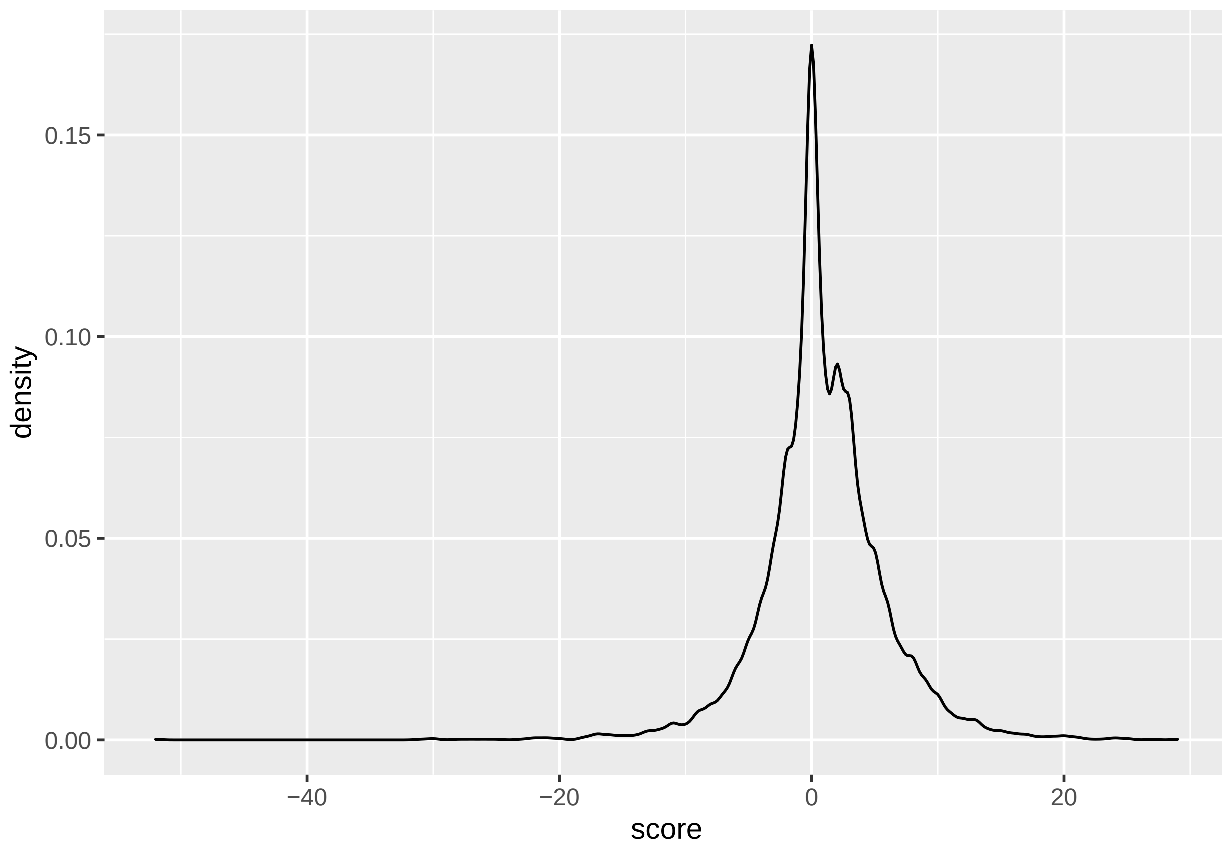
```r
#Does this indicate that our var is binary? Does this *add* to our model?

za%>%
  ggplot(aes(x=log(pop_request+1)))+
  geom_density()
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



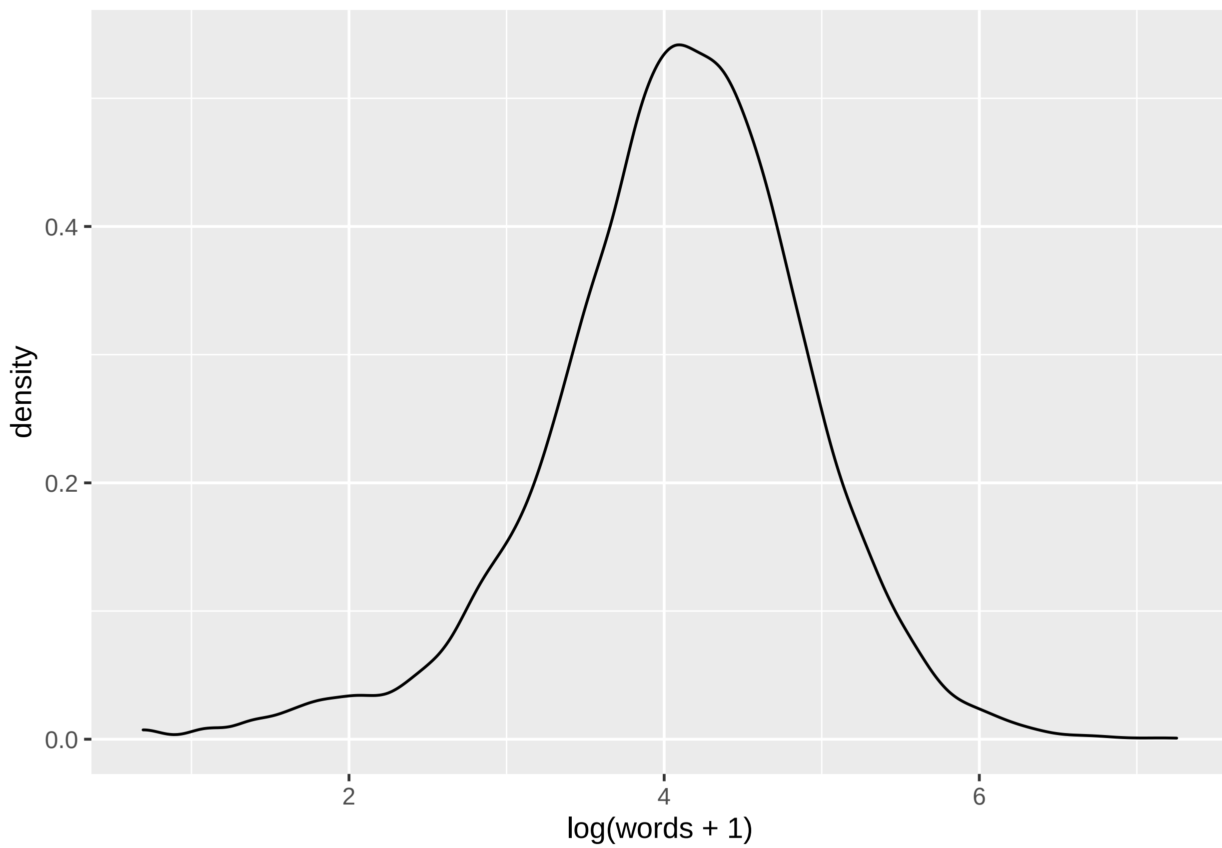

```
#more or less normal

za%>%
  ggplot(aes(x=score))+
  geom_density()
```

```
## Warning: Removed 12 rows containing non-finite values (stat_density).
```



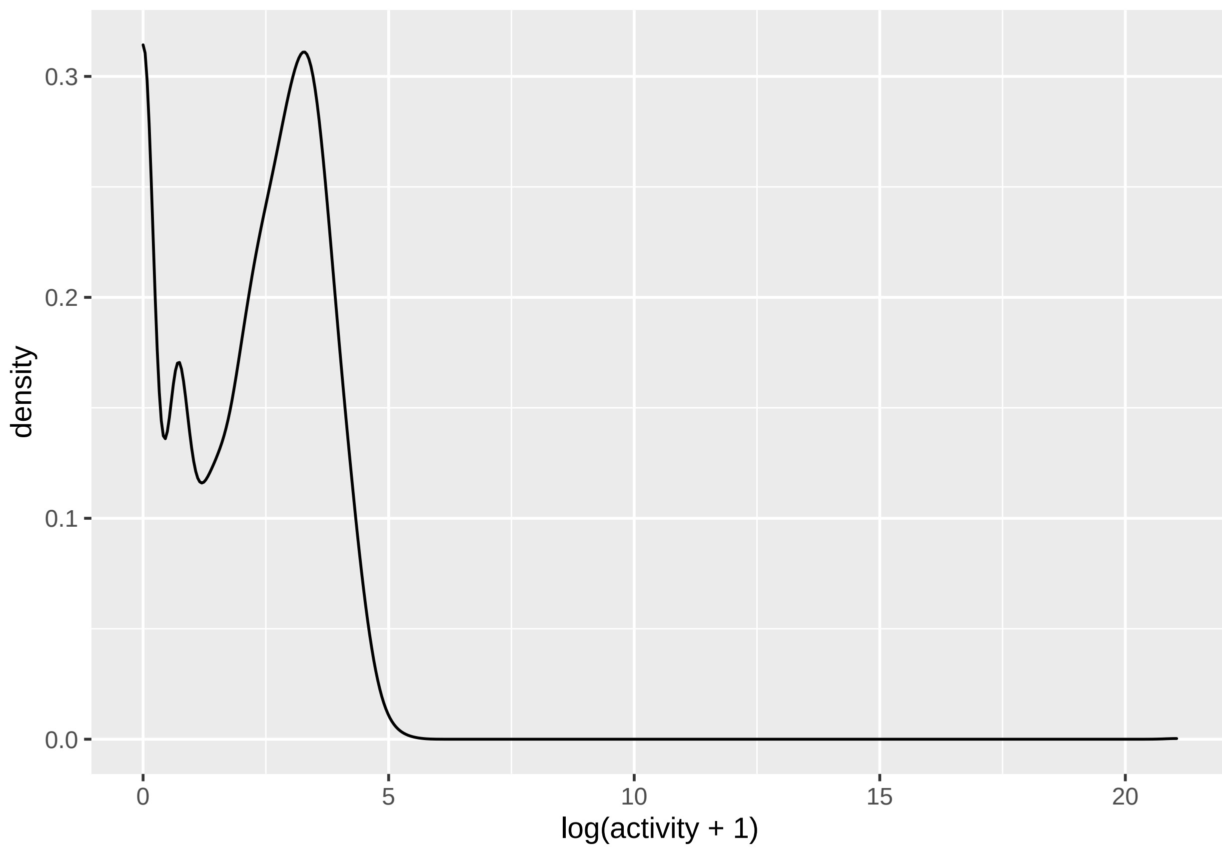

```r
#more or less normal

za%>%
  ggplot(aes(x=log(words+1)))+
  geom_density()
```

```
## Warning: Removed 160 rows containing non-finite values (stat_density).
```




```r
#do we log transform? #normal afterwards

za%>%
  ggplot(aes(x=log(activity+1)))+
  geom_density()
```

```
## Warning: Removed 2 rows containing non-finite values (stat_density).
```




```
logit_rec<-recipe(za_formula, data=za)%>%
  step_log(age,offset = 1)%>%
  step_log(words,offset=1)

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

Put the workflow together

```
logit_wf<-workflow()%>%
  add_recipe(logit_rec)%>%
  add_model(logit_mod)
```

```
logit_results<-fit(logit_wf,data=za_train)
```

```
logit_results%>%
  tidy()
```

```
## # A tibble: 7 x 5
##   term                           estimate std.error statistic  p.value
```

```
##   <chr>                             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)                 -3.96       0.283       -14.0   2.16e-44
## 2 prev_raop_postPosted Before -0.115      0.469        -0.246 8.06e- 1
## 3 raop_posts                   0.980      0.361         2.72  6.55e- 3
## 4 age                          0.104      0.0204        5.08  3.82e- 7
## 5 words                        0.565      0.0607        9.31  1.29e-20
## 6 studentStudent              -0.179      0.166        -1.08  2.79e- 1
## 7 karma                        0.00000940 0.0000163     0.575 5.65e- 1
```

```
logit_final<-last_fit(logit_wf,za_split)

logit_final$.metrics
```

```
## [[1]]
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.752 Preprocessor1_Model1
## 2 roc_auc  binary         0.623 Preprocessor1_Model1
```

2. Compute the predictions from your model from the testing dataset.

3. Calculate the AUC (from the Receiver Operating Characteristic) for the predictions from your model from the testing dataset. Compare your results to the artice linked below.

If we look at our results:

```
logit_results%>%
  tidy()
```

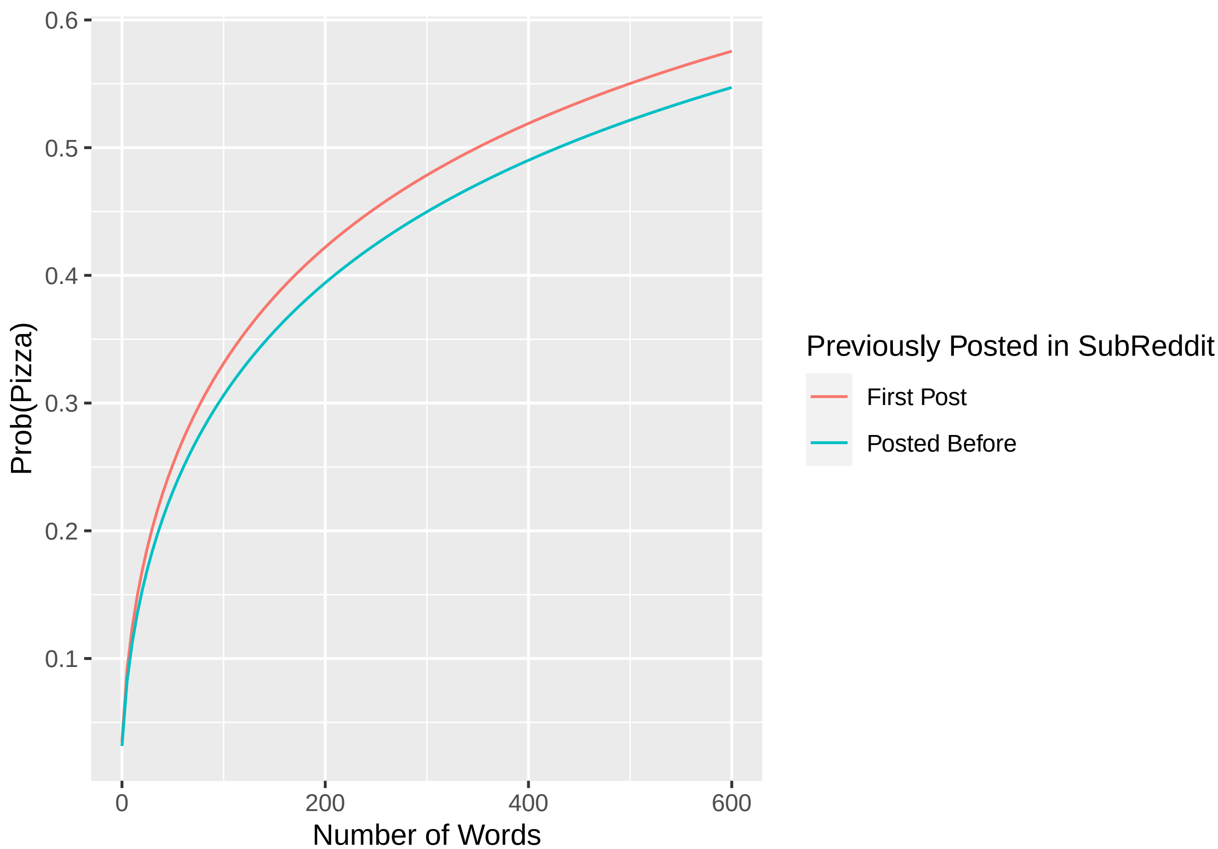
```
## # A tibble: 7 x 5
##   term                        estimate std.error statistic  p.value
##   <chr>                          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)                 -3.96       0.283       -14.0   2.16e-44
## 2 prev_raop_postPosted Before -0.115      0.469        -0.246 8.06e- 1
## 3 raop_posts                   0.980      0.361         2.72  6.55e- 3
## 4 age                          0.104      0.0204        5.08  3.82e- 7
## 5 words                        0.565      0.0607        9.31  1.29e-20
## 6 studentStudent              -0.179      0.166        -1.08  2.79e- 1
## 7 karma                        0.00000940 0.0000163     0.575 5.65e- 1
```

we see that 'raop_posts', 'age', and 'words' were all statistically significant. The next R chunk generates a hypothetical dataset. Arbitrarily, I chose to plot 'words' by 'prev_raop_post' to show the differences based on prediction.

4. Find a way to plot the predictions from your model.

```
#Here we are generating our hypothetical model. To do this we need to fix every value except the ones t


hypo_data<-za_train%>%data_grid(
```

```
  age=mean(age,na.rm=TRUE),# fixes age to be constant
  karma=mean(karma,na.rm=TRUE), #fixes karma to be constant
  raop_posts=mean(raop_posts,na.rm=TRUE),#fixes raop_posts to be constant
  prev_raop_post=as_factor(levels(prev_raop_post)), #generates values for all levels of prev_raop_post
  student=as_factor(levels(student)[1]), #fixes student to be 'No Student'
  words=(seq_range(words,n=100, pretty=TRUE)) #generates values for words within the range of available
)


#The following predicts the probability of Pizza using our generated dataset and the estimates we obtai
plot_data<-logit_results%>%
  predict(hypo_data,type="prob")%>%
  bind_cols(hypo_data)%>%
  rename(`Previously Posted in SubReddit`=prev_raop_post) #this step is just to help with graph below

plot_data%>%
ggplot(aes(x=words,y=.pred_Yes,color=`Previously Posted in SubReddit`))+ #plots words against our Prob(
  geom_line()+
  xlab("Number of Words")+
  ylab("Prob(Pizza)")
```



Was this a good model? We already answered in-class, "no, not really". But we were able to quantify how good it was and even show graphically, that those differences in sensitivity are negligible.

For some ideas, see: http://cs.stanford.edu/~althoff/raop-dataset/altruistic_requests_icwsm.pdf.