Using recipes in tidymodels

Andrew Noecker

3/1/2022

Using recipes in `tidymodels`

In today's class we will explore how to create better models by creating new variables from old ones. This process is sometimes called *feature engineering* and we will learn how to do using `recipes` from `tidymodels`. In this worksheet we will be covering most of the material from https://www.tmwr.org/recipes.html

Using the ames dataset

Let's start by loading the appropriate libraries, datasets and converting our variable `price` so that is in on log-scale. Also let's make sure that we create our testing and training dataset

```
library(tidymodels)
tidymodels_prefer()

library(modeldata)
data(ames)
ames <- ames %>%
  mutate(Sale_Price=log10(Sale_Price))

set.seed(12345)
ames.split <- initial_split(ames, prop=0.8)
ames.train <- training(ames.split)
ames.test <- testing(ames.split)
```

Initial modeling

We are interested in predicting the price of the property adding the following variables:

- `Neighborhood`
- `Gr_Liv_Area`, corresponding to the gross above-grade living area.
- `Year_built`
- `Bldg_type` corresponding to the building type

1. What is the type of each of these four variables? If a variable is categorical how many different values (levels) it has

```
levels(ames$Neighborhood)
```

```
##  [1] "North_Ames"
##  [2] "College_Creek"
##  [3] "Old_Town"
##  [4] "Edwards"
##  [5] "Somerset"
##  [6] "Northridge_Heights"
##  [7] "Gilbert"
##  [8] "Sawyer"
##  [9] "Northwest_Ames"
## [10] "Sawyer_West"
## [11] "Mitchell"
## [12] "Brookside"
## [13] "Crawford"
## [14] "Iowa_DOT_and_Rail_Road"
## [15] "Timberland"
## [16] "Northridge"
## [17] "Stone_Brook"
## [18] "South_and_West_of_Iowa_State_University"
## [19] "Clear_Creek"
## [20] "Meadow_Village"
## [21] "Briardale"
## [22] "Bloomington_Heights"
## [23] "Veenker"
## [24] "Northpark_Villa"
## [25] "Blueste"
## [26] "Greens"
## [27] "Green_Hills"
## [28] "Landmark"
## [29] "Hayden_Lake"
```

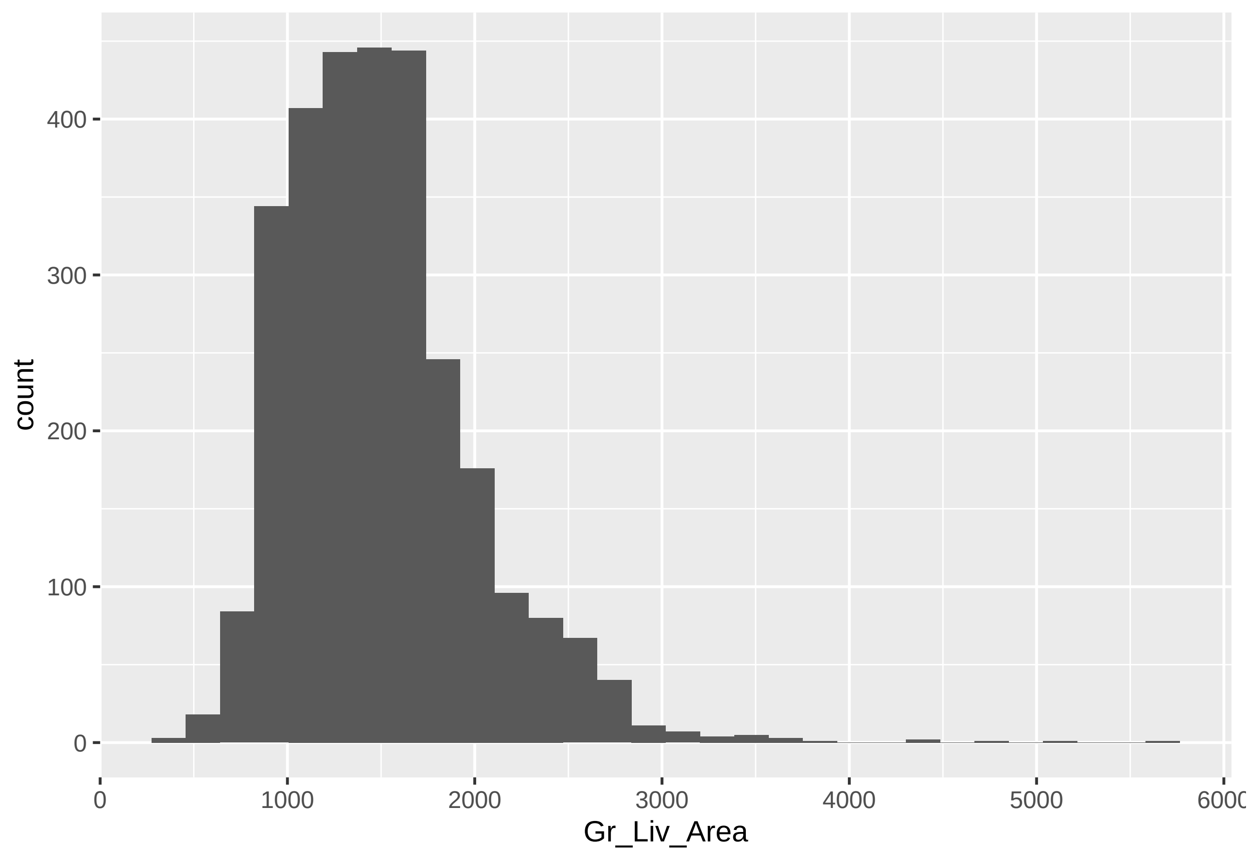
```
levels(ames$Bldg_Type)
```

```
## [1] "OneFam"   "TwoFmCon" "Duplex"   "Twnhs"    "TwnhsE"
```

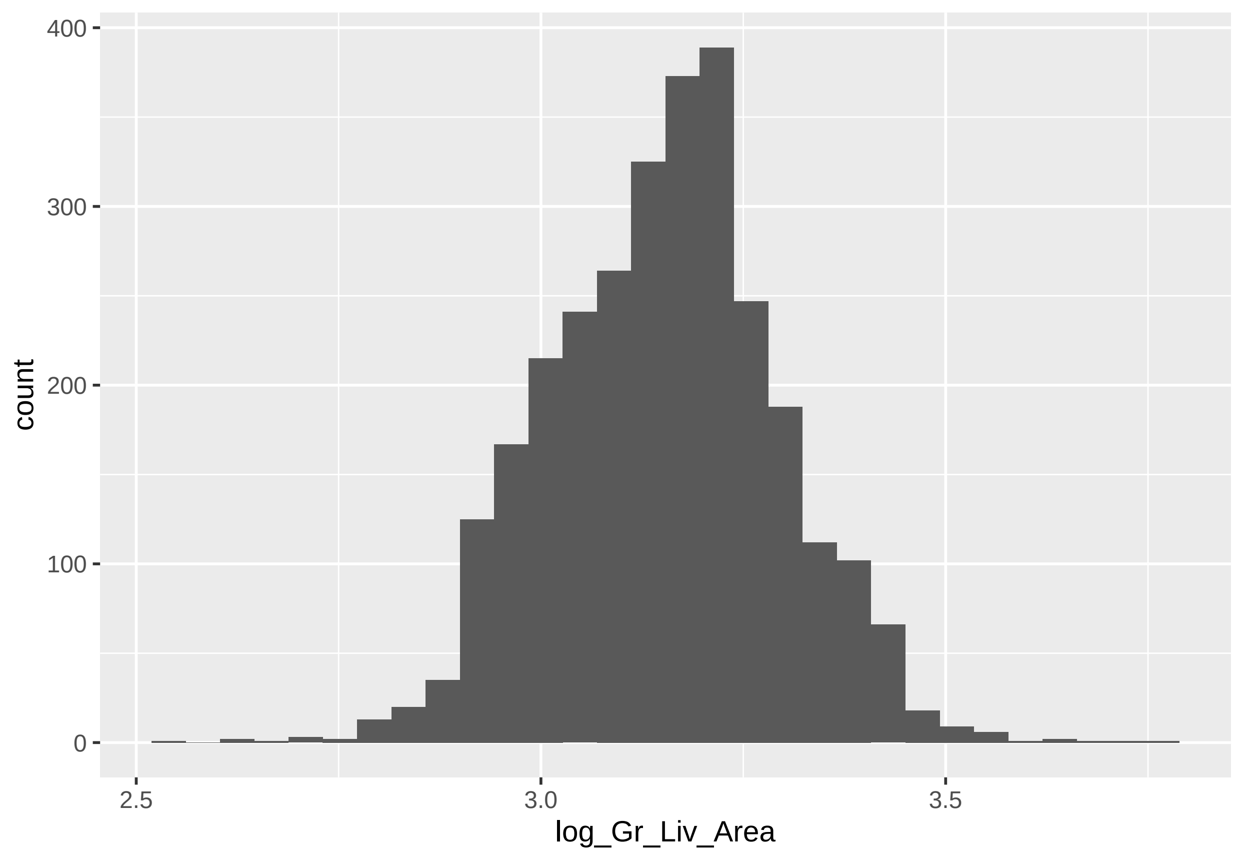
Neighborhood is a categorical factor with 29 different levels. Gr_Liv_Area is a quantitative variable. Year_built is also a quantitative variable. Lastly, Bldg_type is a categorical variable with 5 levels.

2. Do a histogram of `Gr_Liv_Area`. How does this histogram looks using a log scale?

```
ames%>%
  ggplot(aes(x = Gr_Liv_Area))+
    geom_histogram()
```



```
ames%>%
  mutate(log_Gr_Liv_Area = log10(Gr_Liv_Area))%>%
  ggplot(aes(x = log_Gr_Liv_Area))+
    geom_histogram()
```



From the two histograms above, we see that when we use the log scale the distribution is a little more even, closer to a normal distribution. Thus, it better satisfies some of our assumptions for linearity. Without the log scale, the distribution is a little right skewed.

Creating a recipe

A recipe is a collection of steps for preprocessing a dataset. Our initial recipe will include the following steps:

- We would like to make it explicit that we are modeling the `Sale_Price` (response variable) based on `Latitude` and `Longitude`, `Gr_Liv_area`, and `Bldg_type` (explanatory variables)
- We would like to use a log scale for `Gr_Liv_Area`
- We would like to transform all of our categorical variables into indicator variables.

```
ames.recipe <-
  recipe(Sale_Price ~ Longitude + Latitude + Gr_Liv_Area +
            Bldg_Type, data=ames.train) %>%
  step_log(Gr_Liv_Area, base=10) %>%
  step_dummy(all_nominal_predictors())
ames.recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          4
##
## Operations:
##
## Log transformation on Gr_Liv_Area
## Dummy variables from all_nominal_predictors()
```

Once we created the recipe we can use in conjunction with a linear model, add it to a workflow and fit our workflow using our training dataset

```
lm.model <- linear_reg() %>%
  set_engine("lm")

lm.wflow <- workflow() %>%
  add_recipe(ames.recipe) %>%
  add_model(lm.model)

lm.fit <- fit(lm.wflow, ames.train)
```

3.

a. What is the R^2 of the linear model you created? How do you interpret this value?

```
tidy(lm.fit)
```

```
## # A tibble: 8 x 5
##   term                estimate std.error statistic  p.value
##   <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)        -169.      10.3        -16.4  3.11e-57
## 2 Longitude            -1.22     0.0899     -13.6  2.01e-40
## 3 Latitude              1.37     0.128       10.7  5.32e-26
## 4 Gr_Liv_Area           0.846    0.0171      49.6  0
## 5 Bldg_Type_TwoFmCon   -0.133    0.0158      -8.39 8.47e-17
## 6 Bldg_Type_Duplex     -0.115    0.0123      -9.35 2.00e-20
## 7 Bldg_Type_Twnhs      -0.0414   0.0123      -3.36 7.85e- 4
## 8 Bldg_Type_TwnhsE      0.0598   0.00845      7.07 2.04e-12
```

```
glance(lm.fit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic p.value    df logLik    AIC    BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1     0.610         0.608 0.110      521.       0     7  1848. -3678. -3626.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Our new R^2 is 0.61, meaning that 61% of our data's variability is explained by this new model. This is a significant improvement over what we had last time, when our R^2 was ~0.17.

b. Interpret the coefficients corresponding to:

- The living area of the house.

Our logged living area coefficient is 0.846, meaning that we expect a one unit increase in logged living area to lead to a 0.846 unit increase in logged sale price when all other predictors are held equal.

- The type of building.

To interpret type of building, we compare to the baseline type, which is single family home. When holding all other predictors constant, we see that: Two family condos are 0.133 logged units cheaper than single family homes, Duplex homes are 0.115 logged units cheaper than single family homes, Normal townhouses are 0.0414 logged units cheaper than single family homes, and End unit townhouses are 0.0598 logged units more expensive than single family homes.

4. Evaluate your model using the testing dataset. What is the MSE on this dataset?

```
new.ames.test <- lm.fit%>%
  augment(new_data = ames.test)

rmse_vec(new.ames.test$Sale_Price, new.ames.test$.pred)^2
```

```
## [1] 0.0112031
```

We see an MSE of 0.0112, which is a pretty small MSE, indicating our model is doing a decent job. However, it is important to remember we're working on a log scale so our error values will be lower.

5. Add `Year_Built` as an input variable in your existing recipe. What is the R^2 of your model? What is the MSE on the testing dataset?

```
#Recipe Creation
ames.recipe <-
  recipe(Sale_Price ~ Longitude + Latitude + Gr_Liv_Area +
           Bldg_Type + Year_Built, data=ames.train) %>%
  step_log(Gr_Liv_Area, base=10) %>%
  step_dummy(all_nominal_predictors())

#Model Creation
lm.model <- linear_reg() %>%
  set_engine("lm")

lm.wflow <- workflow() %>%
  add_recipe(ames.recipe) %>%
  add_model(lm.model)

lm.fit <- fit(lm.wflow, ames.train)

#Coefficients and Other Model Summary
tidy(lm.fit)
```

```
## # A tibble: 9 x 5
##   term               estimate std.error statistic   p.value
##   <chr>                 <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)        -19.6     9.68         -2.03  4.30e-  2
## 2 Longitude            0.0284  0.0835        0.340 7.34e-  1
## 3 Latitude             0.477   0.110         4.35  1.43e-  5
## 4 Gr_Liv_Area          0.734   0.0145       50.6   0
## 5 Year_Built           0.00261 0.0000793    32.9   1.58e-195
## 6 Bldg_Type_TwoFmCon  -0.0484  0.0133       -3.63  2.93e-  4
## 7 Bldg_Type_Duplex    -0.117   0.0102      -11.4   1.79e- 29
## 8 Bldg_Type_Twnhs     -0.101   0.0103       -9.75  4.83e- 22
## 9 Bldg_Type_TwnhsE    -0.0101  0.00730      -1.39  1.65e-  1
```

```
glance(lm.fit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared  sigma statistic p.value    df logLik    AIC    BIC
##       <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1     0.733         0.732 0.0911      803.       0     8  2295. -4569. -4512.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
#Prediction
new.ames.test <- new.ames.test%>%
  select(-starts_with(".pred"))

new.ames.test <- lm.fit%>%
  augment(new_data = ames.test)

rmse_vec(new.ames.test$Sale_Price, new.ames.test$.pred)^2
```

```
## [1] 0.007296907
```

We now have an R^2 of 0.733, which is a pretty solid improvement over our previous model. We see a new MSE of 0.007297, which is a sizeable drop from the previous model, when we were at 0.0112.

6. Add `Neighborhood` as an input variable recipe to your model from 5. What is the R^2 of your model? What is the MSE on the testing dataset?

```
#Recipe Creation
ames.recipe <-
  recipe(Sale_Price ~ Longitude + Latitude + Gr_Liv_Area +
           Bldg_Type + Year_Built + Neighborhood, data=ames.train) %>%
  step_log(Gr_Liv_Area, base=10) %>%
  step_dummy(all_nominal_predictors())

#Model Creation
lm.model <- linear_reg() %>%
  set_engine("lm")

lm.wflow <- workflow() %>%
  add_recipe(ames.recipe) %>%
  add_model(lm.model)

lm.fit <- fit(lm.wflow, ames.train)

#Coefficients and Other Model Summary
tidy(lm.fit)
```

```
## # A tibble: 37 x 5
##    term                        estimate std.error statistic   p.value
##    <chr>                          <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)                -63.6     33.2         -1.91  5.60e-  2
##  2 Longitude                   -0.113    0.330       -0.342 7.33e-  1
##  3 Latitude                     1.24     0.491        2.53  1.16e-  2
##  4 Gr_Liv_Area                  0.627    0.0144      43.4   1.72e-301
##  5 Year_Built                   0.00205  0.000121    16.9   8.37e- 61
##  6 Bldg_Type_TwoFmCon          -0.0317   0.0117      -2.70  7.08e-  3
##  7 Bldg_Type_Duplex            -0.0922   0.00916    -10.1   2.28e- 23
##  8 Bldg_Type_Twnhs             -0.100    0.0122      -8.21  3.60e- 16
##  9 Bldg_Type_TwnhsE            -0.0389   0.00774     -5.03  5.29e-  7
## 10 Neighborhood_College_Creek   0.0338   0.0284       1.19  2.34e-  1
## # ... with 27 more rows
```

```
glance(lm.fit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared  sigma statistic p.value    df logLik    AIC    BIC
##       <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1     0.801         0.798 0.0792      265.       0    35  2636. -5199. -4986.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
#Prediction
new.ames.test <- new.ames.test%>%
  select(-starts_with(".pred"))

new.ames.test <- lm.fit%>%
  augment(new_data = ames.test)

rmse_vec(new.ames.test$Sale_Price, new.ames.test$.pred)^2
```

```
## [1] 0.005853077
```

Our new R^2 is 0.801, which is another decent increase from our last model. We see a new MSE of 0.00585, which is another decent improvement.

7.

a. Summarize and sort the number of observations in each neighborhood. How many many neighborhoods have less than 20 observations?

```
ames.train%>%
  count(Neighborhood, sort = TRUE)%>%
  print(n = 30)
```

```
## # A tibble: 28 x 2
##    Neighborhood                                n
##    <fct>                                   <int>
##  1 North_Ames                                347
##  2 College_Creek                             222
##  3 Old_Town                                  186
##  4 Edwards                                   159
##  5 Somerset                                  149
##  6 Northridge_Heights                        134
##  7 Gilbert                                   131
##  8 Sawyer                                    123
##  9 Northwest_Ames                            115
## 10 Sawyer_West                               104
## 11 Mitchell                                   91
## 12 Brookside                                  81
## 13 Crawford                                   81
## 14 Iowa_DOT_and_Rail_Road                     72
## 15 Northridge                                 57
## 16 Timberland                                 50
## 17 Stone_Brook                                44
## 18 South_and_West_of_Iowa_State_University    37
## 19 Clear_Creek                                29
## 20 Meadow_Village                             28
## 21 Briardale                                  24
## 22 Bloomington_Heights                        24
## 23 Veenker                                    19
## 24 Northpark_Villa                            19
## 25 Blueste                                     9
## 26 Greens                                      6
## 27 Green_Hills                                 2
## 28 Landmark                                          1
```

```
ames.train%>%
  count(Neighborhood)%>%
  filter(n < 20)
```

```
## # A tibble: 6 x 2
##   Neighborhood        n
##   <fct>           <int>
## 1 Veenker            19
## 2 Northpark_Villa    19
## 3 Blueste             9
## 4 Greens              6
## 5 Green_Hills         2
## 6 Landmark            1
```

We see that there are 6 neighborhoods that have less than 20 observations.

b. Consult the documentation for `step_other` and add a step to your recipe where you collapse neighborhoods with less than 1% of your data. Make sure to add this step before the `step_dummy` command.

```
#Recipe Creation
ames.recipe <-
  recipe(Sale_Price ~ Longitude + Latitude + Gr_Liv_Area +
           Bldg_Type + Year_Built + Neighborhood, data=ames.train) %>%
  step_log(Gr_Liv_Area, base=10) %>%
  step_other(Neighborhood, threshold = 0.01)%>%
  step_dummy(all_nominal_predictors())
```

c. Rerun your workflow and your model. How do you interpret the coefficient of the model associated with the collapsed set of neighborhoods? What is the MSE of this new model?

```
#Model Creation
lm.model <- linear_reg() %>%
  set_engine("lm")

lm.wflow <- workflow() %>%
  add_recipe(ames.recipe) %>%
  add_model(lm.model)

lm.fit <- fit(lm.wflow, ames.train)

#Coefficients and Other Model Summary
tidy(lm.fit)%>%print(n = 31)
```

```
## # A tibble: 31 x 5
##    term                                      estimate std.error statistic   p.value
##    <chr>                                        <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)                               -6.62e+1 32.1          -2.06  3.91e-  2
##  2 Longitude                                 -4.54e-1  0.306        -1.49  1.37e-  1
##  3 Latitude                                   5.45e-1  0.395         1.38  1.68e-  1
##  4 Gr_Liv_Area                                6.25e-1  0.0145       43.2   2.87e-299
##  5 Year_Built                                 2.05e-3  0.000121     16.9   1.16e- 60
##  6 Bldg_Type_TwoFmCon                        -3.14e-2  0.0118       -2.66  7.84e-  3
##  7 Bldg_Type_Duplex                          -9.21e-2  0.00920     -10.0   4.10e- 23
##  8 Bldg_Type_Twnhs                           -1.05e-1  0.0118       -8.92  9.55e- 19
##  9 Bldg_Type_TwnhsE                          -3.90e-2  0.00766      -5.09  3.92e-  7
## 10 Neighborhood_College_Creek              -6.11e-3  0.0245      -0.250 8.03e-  1
## 11 Neighborhood_Old_Town                   -2.23e-2  0.00959     -2.33  1.99e-  2
## 12 Neighborhood_Edwards                    -6.08e-2  0.0192      -3.17  1.55e-  3
## 13 Neighborhood_Somerset                    3.72e-2  0.0127       2.94  3.33e-  3
## 14 Neighborhood_Northridge_Heights          1.11e-1  0.0160       6.99  3.61e- 12
## 15 Neighborhood_Gilbert                    -5.15e-2  0.0130      -3.97  7.43e-  5
## 16 Neighborhood_Sawyer                     -2.18e-2  0.0187      -1.16  2.44e-  1
## 17 Neighborhood_Northwest_Ames             -1.32e-2  0.0104      -1.27  2.04e-  1
## 18 Neighborhood_Sawyer_West                -4.06e-2  0.0230      -1.76  7.81e-  2
## 19 Neighborhood_Mitchell                    3.76e-2  0.0224       1.68  9.35e-  2
## 20 Neighborhood_Brookside                  -4.50e-3  0.0115      -0.393 6.95e-  1
## 21 Neighborhood_Crawford                    1.03e-1  0.0167       6.20  6.85e- 10
## 22 Neighborhood_Iowa_DOT_and_Rail_Road     -7.45e-2  0.0137      -5.44  5.99e-  8
## 23 Neighborhood_Timberland                  7.47e-2  0.0247       3.03  2.48e-  3
## 24 Neighborhood_Northridge                  6.61e-2  0.0166       3.99  6.71e-  5
## 25 Neighborhood_Stone_Brook                 1.39e-1  0.0163       8.55  2.21e- 17
## 26 Neighborhood_South_and_West_of_Iowa_S~ -3.67e-2  0.0204      -1.80  7.16e-  2
## 27 Neighborhood_Clear_Creek                 4.58e-2  0.0233       1.97  4.95e-  2
## 28 Neighborhood_Meadow_Village             -5.44e-2  0.0259      -2.10  3.61e-  2
## 29 Neighborhood_Briardale                  -5.38e-2  0.0204      -2.64  8.37e-  3
## 30 Neighborhood_Bloomington_Heights         1.36e-2  0.0216       0.629 5.29e-  1
## 31 Neighborhood_other                       5.94e-2  0.0152       3.92  9.14e-  5
```

```r
glance(lm.fit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared  sigma statistic p.value    df logLik    AIC    BIC
##       <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1     0.798         0.796 0.0796      305.       0    30  2622. -5180. -4995.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```r
#Prediction
new.ames.test <- new.ames.test%>%
  select(-starts_with(".pred"))

new.ames.test <- lm.fit%>%
  augment(new_data = ames.test)

rmse_vec(new.ames.test$Sale_Price, new.ames.test$.pred)^2
```

```
## [1] 0.005911335
```

Our new R^2 is 0.798, and our new MSE is 0.00591. Both of these are slight decreases from our previous model, which makes sense because we've eliminated a couple predictors. However, this step is beneficial because it lessens the number of predictors we have and makes our model slightly simpler.

To interpret the 'other' or collapsed neighborhood coefficient, we can say that if all other factors are held constant, then we'd expect a house in one of these 'other' neighborhoods to be 0.0594 units of logged sale price more expensive than a house in the baseline neighborhood, which is North_Ames.

8.

a. What two features are you planning to use for your first challenge?

1) Counting the number of pixels that are inked and finding what proportion of those are in the top vs bottom half of the image, and then taking the difference in those proportions. We hypothesize that 3s will have a difference close to 0 (since they are normally symetrical over a hypothetical 'x' axis), which 2s will have a greater difference (probably heavier on the bottom).

2) Our second feature is pretty complicated to explain - for each of the 784 pixels, we calculated the average ink value (between 0 to 255). We then looked at how that average compared among 2s and 3s and found which pixels had the largest difference between their averages of 2s and 3s. This told us which pixels were most indicative of whether a value was 2 or 3. We then took each image and calculated a weighted average by looping through every pixel, multiplying its pixel value by the difference value we calculated above, and adding it to a total sum for that image. In this way, if a digit ends with a negative sum, then this would suggest it's more likely to be a 3 (the difference value we calculated was negative), whereas if it has a large sum it'd suggest it's a 2(because the difference value was positive for 2s).

b. Using the MNIST dataset select a couple of instances of the two digits assigned to your group. Calculate the two features on those instances. Are the two features similar across the two different types of digits?

```r
#Loading in the data
library(caret)
library(dslabs)
mnist <- read_mnist("~/Mscs 341 S22/Class/Data")

set.seed(12345)

index_23 <- which(mnist$train$labels %in% c(2,3))
sample_indices <- sample(index_23, size = 20)
y.trial <- mnist$train$labels[sample_indices][1:20]
x.trial <- mnist$train$images[sample_indices,][1:20,]

#Placeholder for splitting upper half and lower half
row_column <- expand_grid(row = 1:28, col = 1:28)
upper_half_idx <- which(row_column$row <= 14)
lower_half_idx <- which(row_column$row >14)


#If x > 200 then ink present, otherwise not
test_x.trial <- x.trial > 200

#Calculating proportion of ink that is in each half for each number
test_x.trial <- cbind(rowSums(test_x.trial[,upper_half_idx]) / rowSums(test_x.trial),
      rowSums(test_x.trial[,lower_half_idx]) / rowSums(test_x.trial),
      rowSums(test_x.trial[,upper_half_idx]),
      rowSums(test_x.trial[,lower_half_idx]))

mnist_23 <- as_tibble(test_x.trial)%>%
  mutate(y = y.trial,
         diff = V1 - V2,
         abs_diff = abs(diff),
         index = sample_indices)%>%
  rename(top_prop = "V1",
         bottom_prop = "V2",
         top_count = "V3",
         bottom_count = "V4")
```

We now have the dataset and can compare this feature across 2s and 3s

```r
mnist_23%>%
  group_by(y)%>%
  summarize(meanDiff = mean(diff))
```

```
## # A tibble: 2 x 2
##       y meanDiff
##   <int>    <dbl>
## 1     2  -0.185
## 2     3  -0.0148
```

We see that 3s are generally more symmetrical because the difference is closer to 0(meaning equal amounts of the ink is on the top and bottom), which makes sense. On the other hand, 2s have a larger difference that is negative, indicating that more of the ink in 2s is on the bottom half, which also makes sense.

—Second Feature—

The following code goes through what is described above in part a for the second feature, and the function at the end calculates the final 'feature value' for 'size' number of images.

```
master.df = data.frame(matrix(nrow = 784,ncol = 2))
indicator.2 = mnist$train$labels==(2)
indicator.2 = as_tibble(indicator.2)
indicator.3 = mnist$train$labels==(3)
indicator.3 = as_tibble(indicator.3)

for (i in 1:784){
tib.2 = tibble(mnist[1][["train"]][["images"]][,i])

colnames(tib.2) = "actual.value"

Mean = cbind(indicator.2,tib.2)%>%
  dplyr::filter(value == TRUE)%>%
  summarize(mean = mean(actual.value))

master.df[1][i,] = as.numeric(Mean)}

for (i in 1:784){
tib.3 = tibble(mnist[1][["train"]][["images"]][,i])

colnames(tib.3) = "actual.value"

Mean = cbind(indicator.3,tib.3)%>%
  dplyr::filter(value == TRUE)%>%
  summarize(mean = mean(actual.value))

master.df[2][i,] = as.numeric(Mean)}

master.df.1 = master.df%>%
  mutate(diff = X1 - X2)%>%
  mutate(pos.ind = ifelse(diff>0,1,0))%>%
  mutate(RN = row_number())%>%
  select(diff)

feature.extracter.train <- function(size){

index_23 <- which(mnist$train$labels %in% c(2,3))
y <- mnist$train$labels[index_23]
x <- mnist$train$images[index_23,]

 i = 1
```

```r
 iterations = size
 value.vector = vector(length = iterations)
 for (i in 1:size){
 binded = cbind(master.df.1,x[i,])
 colnames(binded) = c("diff", "value")
 binded = binded%>%
 mutate(product = diff*value)%>%
   summarize(sum = sum(product))
 value.vector[i] = as.numeric(binded)

   }

 Return = tibble(letter = as.factor(y[1:size]),value.vector)

 Return

}


feature.extracter.train(200)%>%
  group_by(letter)%>%
  summarize(meanValue = mean(value.vector))
```

```
## # A tibble: 2 x 2
##   letter meanValue
##   <fct>      <dbl>
## 1 2        634088.
## 2 3       -614936.
```

We see that this feature demonstrates a start contrast between the digits, with 3s having an average value of -614936 and 2s having an average value of 634088, indicating a pretty successful feature.

Thus, both features are pretty different across digits, indicating they probably do a good job distinguishing between our two digits.