Using recipes in tidymodels

J. Williams

3/10/2022

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

```
ames %>%
  select(Neighborhood, Gr_Liv_Area, Year_Built, Bldg_Type)
```

```
## # A tibble: 2,930 x 4
##    Neighborhood Gr_Liv_Area Year_Built Bldg_Type
##    <fct>              <int>      <int> <fct>
##  1 North_Ames          1656       1960 OneFam
##  2 North_Ames           896       1961 OneFam
##  3 North_Ames          1329       1958 OneFam
##  4 North_Ames          2110       1968 OneFam
##  5 Gilbert             1629       1997 OneFam
##  6 Gilbert             1604       1998 OneFam
##  7 Stone_Brook         1338       2001 TwnhsE
##  8 Stone_Brook         1280       1992 TwnhsE
##  9 Stone_Brook         1616       1995 TwnhsE
## 10 Gilbert             1804       1999 OneFam
## # ... with 2,920 more rows
```

```
str(ames$Neighborhood)
```

```
##  Factor w/ 29 levels "North_Ames","College_Creek",..: 1 1 1 1 7 7 17 17 17 7 ...
```

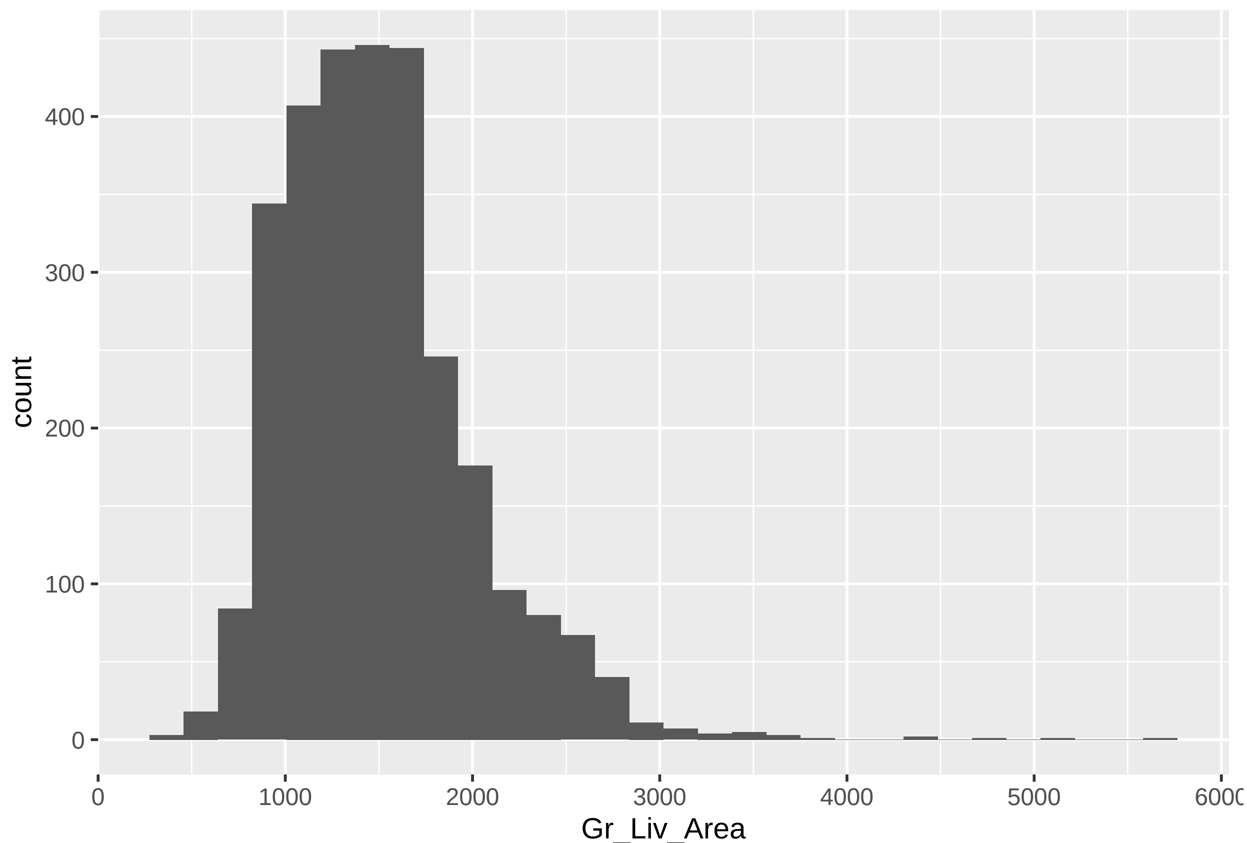
```
str(ames$Bldg_Type)
```

```
##  Factor w/ 5 levels "OneFam","TwoFmCon",..: 1 1 1 1 1 1 5 5 5 1 ...
```

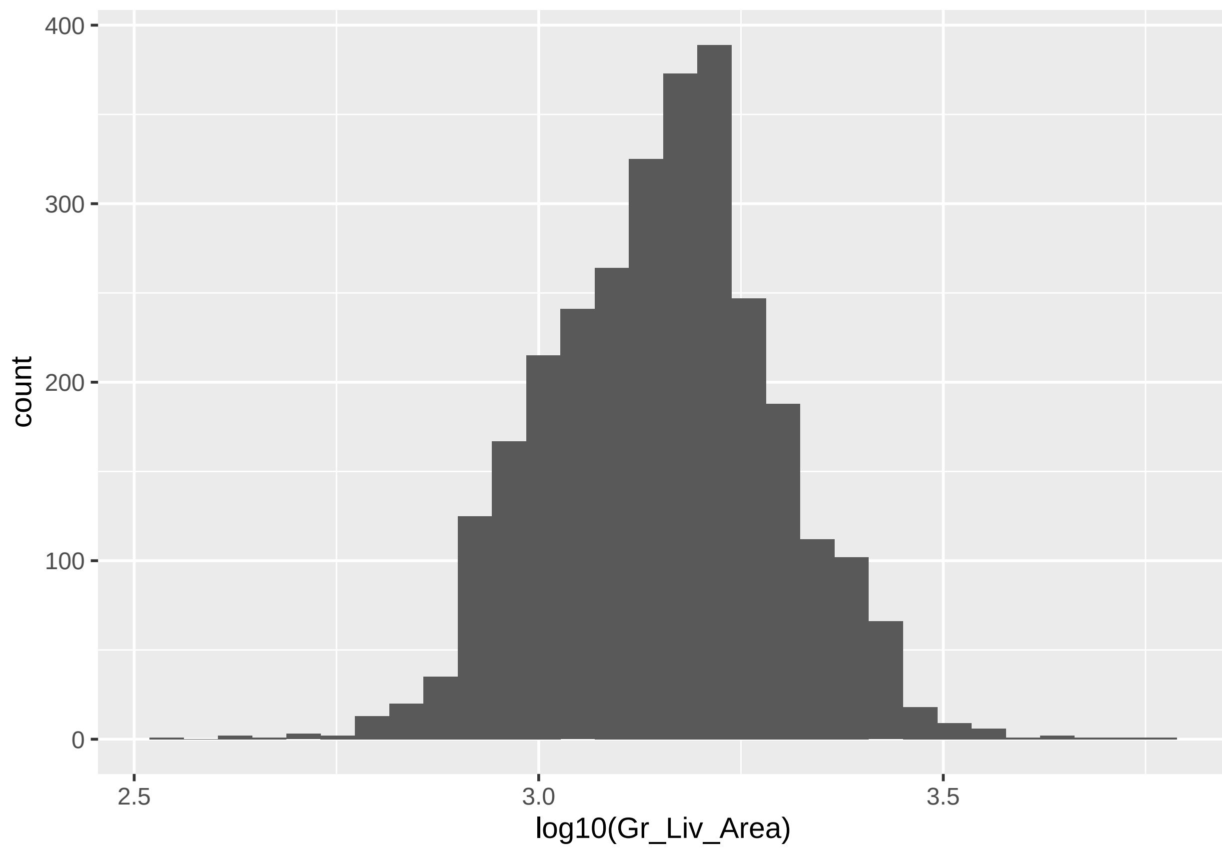
```
# Neighborhood is a factor describing house's neighborhood, with 29 levels
# Gr_Liv_Area is an integer representing the square footage of the house
# Year_Built is an integer representing the year the house was built
# Bldg_Type is a factor describing the type of house, with 5 levels
```

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

```
ggplot(ames, aes(Gr_Liv_Area)) +
  geom_histogram()
```




```
ggplot(ames, aes(log10(Gr_Liv_Area))) +
  geom_histogram()
```




```
# Using the log scale, the data looks much more symmetrical
```

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>
```

```
# R^2 == 0.610
# This means that the model we have created can explain 61% of the variance
```

b. Interpret the coefficients corresponding to:

- The living area of the house.
- The type of building.

```
# There is a positive association with living area and (log10) sale price of the house. A 1-unit increa

# There is a positive association with "TwnhsE" type homes and sale price, but a negative association wi
```

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

```
model.predict = predict(lm.fit, ames.test)
```

```r
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
```

```r
# MSE == 0.0112031
```

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?

```r
ames.recipe0 <-
  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())

lm.workflow0 <- workflow() %>%
  add_recipe(ames.recipe0) %>%
  add_model(lm.model)

lm.fit0 <- fit(lm.workflow0, ames.test)
glance(lm.fit0)
```

```
## # 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     0.780         0.777 0.0852      256. 2.05e-184     8   616. -1213. -1169.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```r
# R^2 == 0.780

new.ames.test0 <- lm.fit0 %>%
  augment(new_data = ames.test)

rmse_vec(new.ames.test0$Sale_Price, new.ames.test0$.pred) ^2
```

```
## [1] 0.007145553
```

```r
# MSE == 0.007145553
```

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?

```r
ames.recipe1 <-
  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())

lm.workflow1 <- workflow() %>%
  add_recipe(ames.recipe1) %>%
  add_model(lm.model)

lm.fit1 <- fit(lm.workflow1, ames.test)
```

```
glance(lm.fit1)
```

```
## # 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     0.838         0.828 0.0748      86.5 9.95e-195    33   705. -1340. -1187.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
# R^2 == 0.838

new.ames.test1 <- lm.fit1 %>%
  augment(new_data = ames.test)

rmse_vec(new.ames.test1$Sale_Price, new.ames.test1$.pred) ^2
```

```
## [1] 0.005274777
```

```
# MSE == 0.0052747777
```

7.

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

```
check <- ames %>%
  group_by(Neighborhood) %>%
  summarize(n())

# 4 neighborhoods have less than 20
```

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.

```
ames.recipe2 <-
  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())

lm.workflow2 <- workflow() %>%
  add_recipe(ames.recipe2) %>%
  add_model(lm.model)

lm.fit2 <- fit(lm.workflow2, ames.test)
```

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?

```
glance(lm.fit2)
```

```
## # 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     0.835         0.826 0.0753      97.0 1.72e-196    29   700. -1338. -1202.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```r
view <- tidy(lm.fit2)

# There is a small positive association with any of the "other" neighborhoods on sale price.

new.ames.test2 <- lm.fit2 %>%
  augment(new_data = ames.test)

rmse_vec(new.ames.test2$Sale_Price, new.ames.test2$.pred) ^2
```

```
## [1] 0.005372731
```

```r
# MSE = 0.005372731
```

8.

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

```
# Our preliminary plan is to use the average color in the top left and top right quadrants.
```

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?

```
### NOTE: for some reason, while trying to knit the file, read_mnist() isn't working
# properly, and the interpreter says that the function doesn't exist. However,
# everything works just fine when actually running the code outside of the "knit."
# I have included the code below, but I'm not running it so that I can create
# a PDF to submit.

mnist <- read_mnist("~/Mscs 341 S22/Class/Data")

mnist7 <- list()
mnist4 <- list()

i <- 1
for (x in 1:60000) {
  if (mnist$train$labels[x] == 4) {
    mnist4[[i]] <- mnist$train$images[x,]
    i <- i + 1
  }
}

i <- 1
for (x in 1:60000) {
  if (mnist$test$labels[x] == 7) {
    mnist7[[i]] <- mnist$train$images[x,]
    i <- i + 1
  }
}

plotImage <- function(dat, size=28) {
  imag <- matrix(dat,nrow=size)[,28:1]
  image(imag, col=grey.colors(256), xlab="", ylab="")
}

index <- (mnist$train$labels == 4) | (mnist$train$labels == 7)
images.matrix <- mnist$train$images[index,]
```

```
prop_47 <- function(image, size = 28){
  a.image <- images.matrix[image,]
  a.image <- matrix(a.image, nrow = 28)[,28:1]

  a.quad1 <- a.image[1:14, 1:14]
  a.quad2 <- a.image[1:14, 15:28]

  a.x1 <- as_tibble(a.quad1, .name_repair = c("unique")) %>%
    pivot_longer(1:14, names_to = "column", values_to = "pixel") %>%
    mutate(pixel = ifelse(pixel >= 100, 1, 0)) %>%
    summarise(prop = sum(pixel / 784))

  a.x2 <- as_tibble(a.quad2, .name_repair = c("unique")) %>%
    pivot_longer(1:14, names_to = "column", values_to = "pixel") %>%
    mutate(pixel = ifelse(pixel >= 100, 1, 0)) %>%
    summarise(prop = sum(pixel / 784))

  print(a.x1)
  print(a.x2)
}

# a four
prop_47(1)
#0.0281 and 0.0268

# another four
prop_47(2)
#0.0434 and 0.0230

# a seven
prop_47(3)
#0.0332 and 0.0357

# another seven
prop_47(7)
#0.0204 and 0.0281
```

Our group had this homework assignment's deadline extended to 10 March at 6:00pm.