Ingersoll_Lab2

Sofia Ingersoll

2024-01-24

Today we will be continuing the pumpkin case study from last week. We will be using the data that you cleaned and split last time (pumpkins_train) and will be comparing our results today to those you have already obtained. Open and run your Lab 1.Rmd as a first step so those objects are available in your Environment.

Once you have done that, we'll start today's lab by specifying a recipe for a polynomial model. First we specify a recipe that identifies our variables and data, converts the package variable to a numerical form, and then adds a polynomial effect with step_poly()

```
# Specify a recipe
poly_pumpkins_recipe <-
  recipe(price ~ package, data = pumpkins_train) %>%
  step_integer(all_predictors(), zero_based = TRUE) %>%
  step_poly(all_predictors(), degree = 4)
```

How did that work? Later we will learn about model tuning that will let us do things like find the optimal value for degree. For now, we'd like to have a flexible model, so we'll use a relatively large value.

Polynomial regression is still linear regression, so our model specification looks similar to before.

```
# Create a model specification called poly_spec
poly_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")
```

Question 1: Now take the recipe and model specification that just created and bundle them into a workflow called poly_df.

```
# Bundle recipe and model spec into a workflow
poly_wf <- workflow() %>%
  add_recipe(poly_pumpkins_recipe) %>%
  add_model(poly_spec)
```

Question 2: fit a model to the pumpkins_train data using your workflow and assign it to poly_wf_fit

```
# Create a model
poly_wf_fit <- poly_wf %>%
  fit(data = pumpkins_train)
```

```
# Print learned model coefficients
poly_wf_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_integer()
## * step_poly()
##
## -- Model ----------------------------------------------------------------------
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
##    (Intercept)  package_poly_1  package_poly_2  package_poly_3  package_poly_4
##        27.9706        103.8566       -110.9068        -62.6442          0.2677
```

```
# Make price predictions on test data
poly_results <- poly_wf_fit %>% predict(new_data = pumpkins_test) %>%
  bind_cols(pumpkins_test %>% select(c(package, price))) %>%
  relocate(.pred, .after = last_col())

# Print the results
poly_results %>%
  slice_head(n = 10)
```

```
## # A tibble: 10 x 3
##    package              price .pred
##    <chr>                <dbl> <dbl>
##  1 1 1/9 bushel cartons  13.6  15.9
##  2 1 1/9 bushel cartons  16.4  15.9
##  3 1 1/9 bushel cartons  16.4  15.9
##  4 1 1/9 bushel cartons  13.6  15.9
##  5 1 1/9 bushel cartons  15.5  15.9
##  6 1 1/9 bushel cartons  16.4  15.9
##  7 1/2 bushel cartons    34    34.4
##  8 1/2 bushel cartons    30    34.4
##  9 1/2 bushel cartons    30    34.4
## 10 1/2 bushel cartons    34    34.4
```

Now let's evaluate how the model performed on the test_set using yardstick::metrics().

```
metrics(data = poly_results, truth = price, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
```

```
## 1 rmse     standard       3.27
## 2 rsq      standard       0.892
## 3 mae      standard       2.35
```

Question 3: How do the performance metrics differ between the linear model from last week and the polynomial model we fit today? Which model performs better on predicting the price of different packages of pumpkins?

Let's visualize our model results. First prep the results by binding the encoded package variable to them.

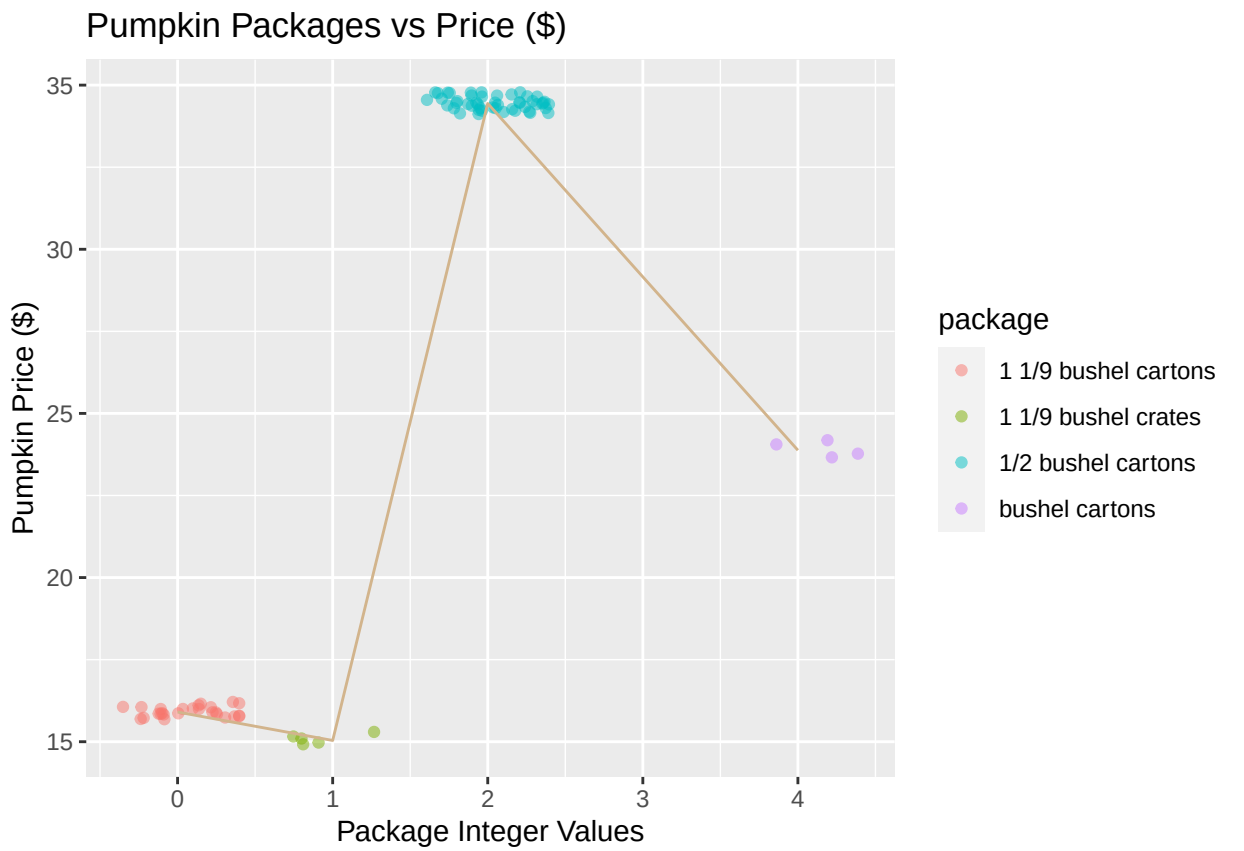
```
# Bind encoded package column to the results
poly_results <- poly_results %>%
  bind_cols(package_encode %>%
              rename(package_integer = package)) %>%
  relocate(package_integer, .after = package)


# Print new results data frame
poly_results %>%
  slice_head(n = 5)
```

```
## # A tibble: 5 x 4
##   package              package_integer price .pred
##   <chr>                          <int> <dbl> <dbl>
## 1 1 1/9 bushel cartons               0  13.6  15.9
## 2 1 1/9 bushel cartons               0  16.4  15.9
## 3 1 1/9 bushel cartons               0  16.4  15.9
## 4 1 1/9 bushel cartons               0  13.6  15.9
## 5 1 1/9 bushel cartons               0  15.5  15.9
```

OK, now let's take a look!

Question 4: Create a scatter plot that takes the poly_results and plots package vs. price. Then draw a line showing our model's predicted values (.pred). Hint: you'll need separate geoms for the data points and the prediction line.

```
# Make a scatter plot
  ggplot(poly_results, aes(x = package_integer,
                           y = .pred)) +
  geom_jitter(aes(color = package),
              alpha = .5) +
  geom_line(aes(y = .pred),
            color = 'tan',
            width = 3) +
  labs(title = "Pumpkin Packages vs Price ($)",
       x = 'Package Integer Values',
       y = 'Pumpkin Price ($)')
```



You can see that a curved line fits your data much better.

Question 5: Now make a smoother line by using geom_smooth instead of geom_line and passing it a polynomial formula like this: geom_smooth(method = lm, formula = y ~ poly(x, degree = 3), color = "midnightblue", size = 1.2, se = FALSE)

```
# Make a smoother scatter plot
  ggplot(poly_results, aes(x = package_integer,
                           y = .pred)) +
  geom_jitter(aes(color = package),
              alpha = .5) +
  geom_line(aes(y = .pred),
            color = 'tan',
            width = 4) +
  geom_smooth(method = lm, formula = y ~ poly(x, degree = 3), color = 'midnightblue', size = 1.2, se = F
  labs(title = "Pumpkin Packages vs Price ($)",
       x = 'Package Integer Values',
       y = 'Pumpkin Price ($)')
```

OK, now it's your turn to go through the process one more time.

Additional assignment components : 6. Choose a new predictor variable (anything not involving package type) in this dataset.

```
# Specify a recipe
poly_var_recipe <-
  recipe(price ~ variety,
         data = pumpkins_train) %>%
  step_integer(all_predictors(),
               zero_based = TRUE)

# encode it for a polynomial model
variety_baked <- poly_var_recipe %>%
  prep() %>%
  bake(new_data = pumpkins_test) %>%
  select(variety)
```

7. Determine its correlation with the outcome variable (price). (Remember we calculated a correlation matrix last week)

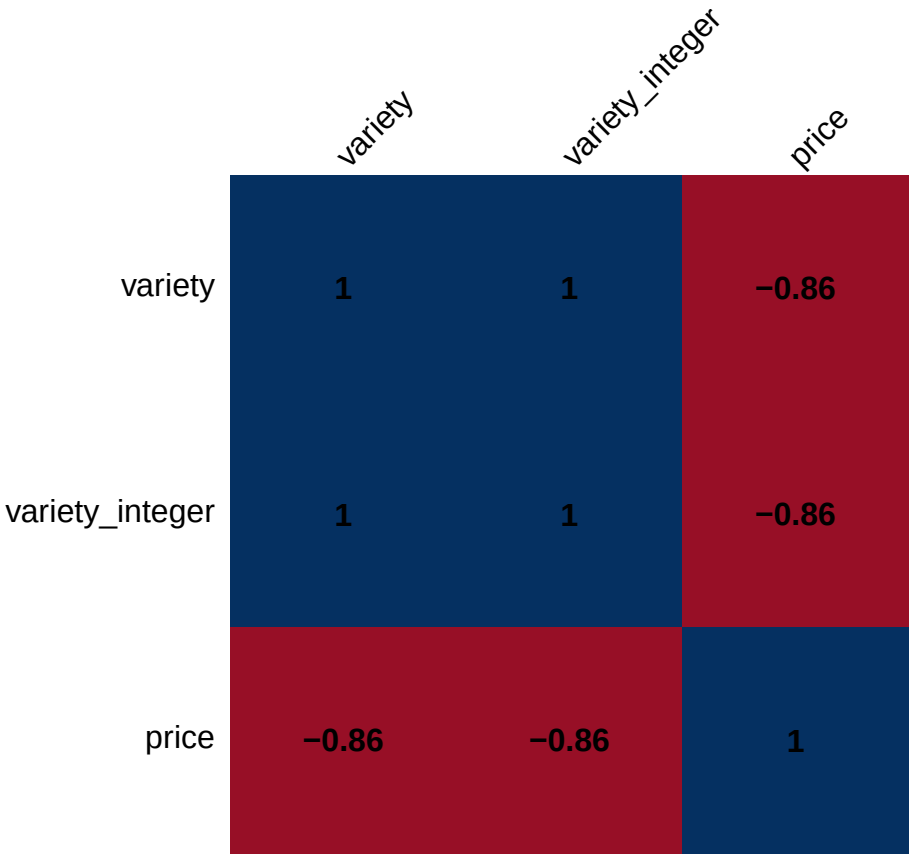
There is a negative correlation of `-0.861` between the variety type and price of pumpkins.

```
# Prep & bake the recipe to determine correlation
poly_baked <- prep(poly_var_recipe) %>%
  bake(new_data = NULL)
```

```r
# Bind encoded package column to the results
poly_baked <- poly_baked %>%
  mutate(variety_integer = as.integer(variety)) %>%
  relocate(variety_integer, .after = variety)

# determine variety correlation with price
cor(poly_baked$variety_integer, poly_baked$price)
```

```
## [1] -0.8611427
```

```r
# Obtain correlation matrix
corr_mat <- cor(poly_baked)

# Make a correlation plot between the variables
corrplot(corr_mat, method = "shade", shade.col = NA, tl.col = "black", tl.srt = 45, addCoef.col = "blac
```



8. Create and test a model for your new predictor:

- Create a recipe
- Build a model specification (linear or polynomial)

```
# Create a recipe
glm_poly_recipe <- recipe(price ~ variety,
                          data = pumpkins_train) %>%
  step_integer(all_predictors(),
               zero_based = TRUE) %>%
  step_poly(all_predictors(),
            degree = 3)

# Create a model specification called poly_spec
poly_mod_spec <- linear_reg() %>%
  set_engine("glm") %>%
  set_mode("regression")
```

- Bundle the recipe and model specification into a workflow
- Create a model by fitting the workflow

```
# Bundle recipe and model spec into a workflow
poly_var_wf <- workflow() %>%
  add_recipe(glm_poly_recipe) %>%
  add_model(poly_mod_spec)

# Create a model
poly_var_wf_fit <- poly_var_wf %>%
  fit(data = pumpkins_train)
```

- Evaluate model performance on the test data

A large value of `4.13` was tabulated for the estimated RMSE, indicating a greater variance in the data distribution compared to the model fit. The values output by this polynomial model are smaller than the previous linear model. Therefore, we can deduce that we are on track for improving our model. However, we still have a reasonable ways to go before our model will be running more accurately.

```
# Let's create price predictions using test data
poly_var_results <- poly_var_wf_fit %>%
  predict(new_data = pumpkins_test) %>%
  bind_cols(pumpkins_test %>%
              select(c(variety, price))) #%>%
#relocate(variety_integer, .after = variety)

poly_var_results <- poly_var_results %>%
  cbind(poly_baked$variety_integer)


# Let's take a peek at the results
poly_var_results %>%
  slice_head(n = 10)
```

```
##       .pred   variety    price poly_baked$variety_integer
## 1  16.17451  PIE TYPE 13.63636                           1
## 2  16.17451  PIE TYPE 16.36364                           1
## 3  16.17451  PIE TYPE 16.36364                           3
## 4  16.17451  PIE TYPE 13.63636                           1
```

```
## 5  16.17451  PIE TYPE 15.45455                          1
## 6  16.17451  PIE TYPE 16.36364                          1
## 7  34.16427 MINIATURE 34.00000                          1
## 8  34.16427 MINIATURE 30.00000                          3
## 9  34.16427 MINIATURE 30.00000                          1
## 10 34.16427 MINIATURE 34.00000                          1
```

```
# Evaluate performance of polynomial regression
metrics(data = poly_var_results, truth = price, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       4.13
## 2 rsq     standard       0.827
## 3 mae     standard       2.52
```

- Create a visualization of model performance

The visualization below demonstrates how the initial fitting method was overfitting for the variance in the model. Whereas a polynomial regression provided a smoother curve with less precision and more overall accuracy of data distribution when fitting.

```
# Creating a visualization
  ggplot(poly_var_results,
         aes(poly_baked$variety_integer,
                 y = price)) +
  geom_jitter(aes(color = variety),
              alpha = .5) +
  geom_line(aes(y = .pred),
            color = 'tan',
            width = 4) +
  geom_smooth(method = lm, formula = y ~ poly(x, degree = 3), color = 'midnightblue', size = 1.2, se = F
  labs(title = "Pumpkin Varieties vs Price ($)",
       x = 'Variety Integer Values',
       y = 'Pumpkin Price ($)')
```

Lab 2 due 1/24 at 11:59 PM