Childers_Lab2

Heather

Lab 2 due 1/24 at 11:59 PM

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_wf.

```
# 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(pumpkins_train)
```

```r
# 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
```

```r
# 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().

```r
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?
The rmse and the mae are both smaller than our model from last week and the r squared value is closer to 1 than our model from last week

Which model performs better on predicting the price of different packages of pumpkins? The polynomial model fits the pricing better than the linear model.

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

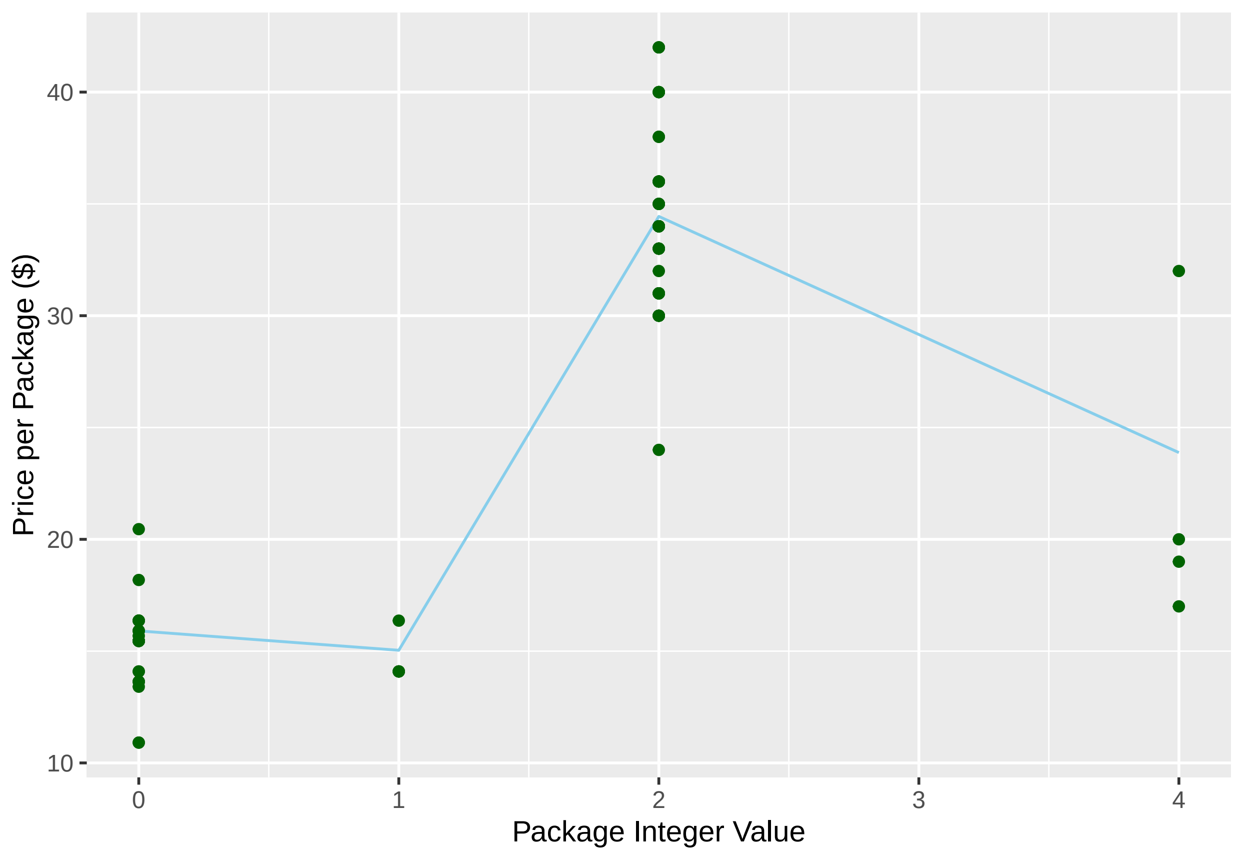
```r
# Bind encoded package column to the results
poly_results <- poly_results %>%
  bind_cols(package_encode %>%
              rename(package_integer = package)) %>%
  relocate(package_integer, .after = package)


# 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.

```r
# Make a scatter plot
ggplot(poly_results, aes(x = package_integer, y = price))+
  geom_line(aes(y = .pred), color = "skyblue")+
  geom_point(color = "darkgreen")+
  labs(x = "Package Integer Value",
       y = "Price per Package ($)")
```

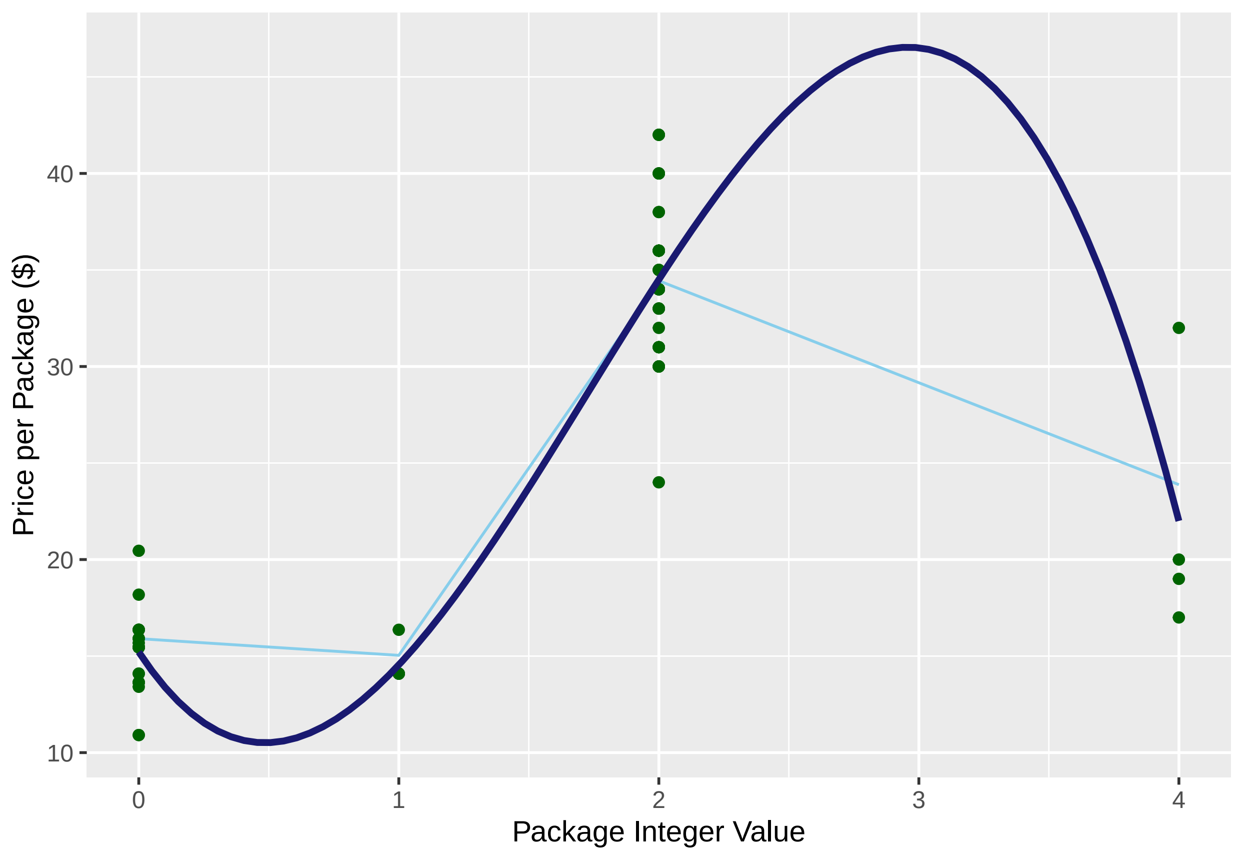


You can see that a curved line fits your data much better.

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 = price))+
  geom_line(aes(y = .pred), color = "skyblue")+
  geom_point(color = "darkgreen")+
  labs(x = "Package Integer Value",
       y = "Price per Package ($)")+
  geom_smooth(method = lm, formula = y ~ poly(x, degree = 3), color = "midnightblue", size = 1.2, se =
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



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. – Variety ~ price**

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

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

```
poly_var_wf <-workflow() %>%
  add_recipe(poly_variety_recipe) %>%
  add_model(poly_spec)

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

poly_var_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)  variety_poly_1  variety_poly_2  variety_poly_3
##          27.97         -153.39          -17.27           14.99
```

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

- Create a recipe

- Build a model specification (linear or polynomial)

- Bundle the recipe and model specification into a workflow

- Create a model by fitting the workflow

```r
lm_pumpkin_recipe <- recipe(price ~ variety, data = pumpkins_train) %>%
  step_integer(all_predictors(), zero_based = TRUE)

variety_encode <- lm_pumpkin_recipe %>%
  prep() %>%
  bake(new_data = pumpkins_test) %>%
  select(variety)

# Make price predictions on test data
poly_var_results <- poly_var_wf_fit %>% predict(new_data = pumpkins_test) %>%
  bind_cols(pumpkins_test %>% select(c(variety, price))) %>%
  relocate(.pred, .after = last_col())

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
```

Evaluate model performance on the test data

The RMSE can be interpreted as the average difference between the actual value and the values predicted by the model. We can see that this model has a RMSE that's about 2/3 of the RMSE from the linear pumpkin model, however this model was not as accurate as the polynomial model we created for pumpkin bushel price. Similarly, when comparing the r squared values between the three models, you can see there was a stronger correlation with the polynomial models ratehr than the linear model.

```
#Lab 1 Metrics
metrics(data = lm_results,
        truth = price,
        estimate = .pred)
```

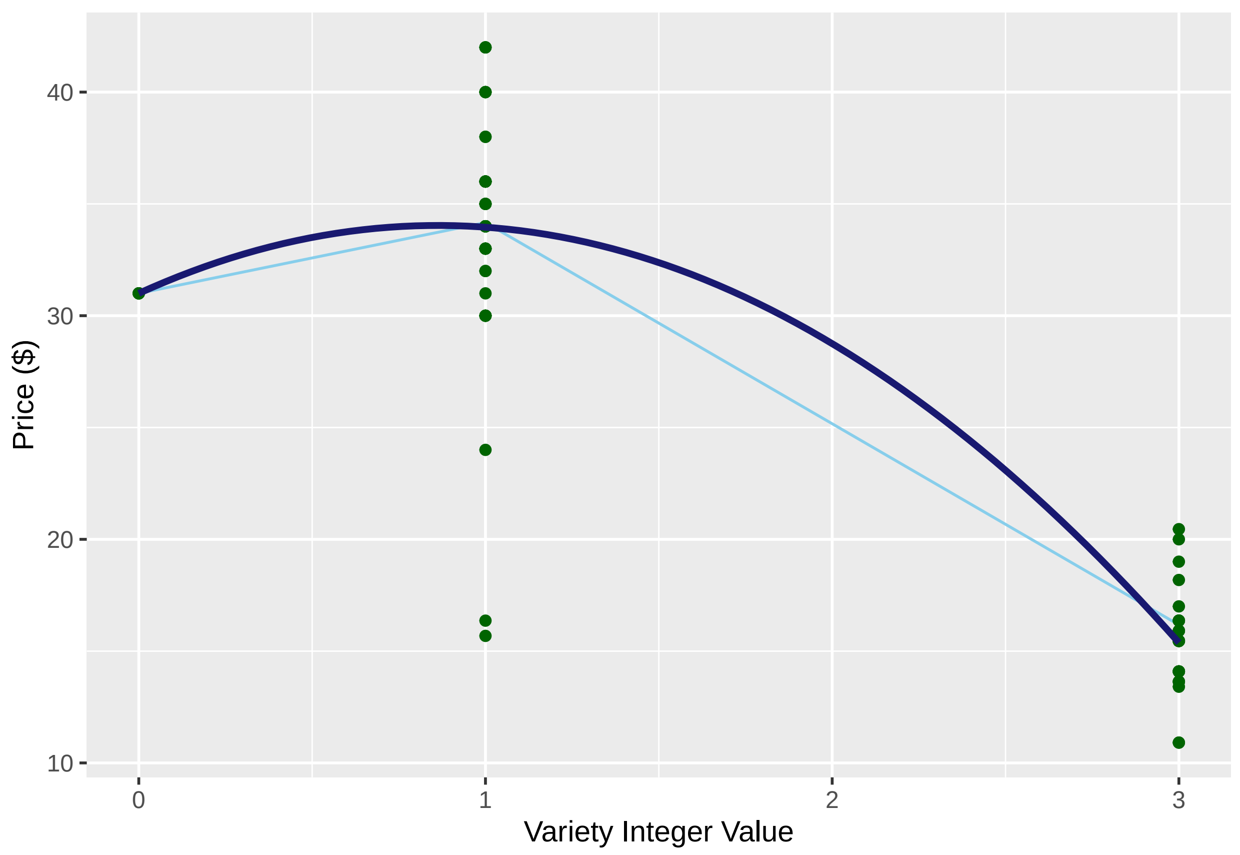
```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       7.23
## 2 rsq     standard       0.495
## 3 mae     standard       5.94
```

```
poly_var_results <- poly_var_results %>%
  bind_cols(variety_encode %>%
              rename(variety_integer = variety)) %>%
  relocate(variety_integer, .after = variety)


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

```
## # A tibble: 10 x 4
##    variety   variety_integer price .pred
##    <chr>               <int> <dbl> <dbl>
##  1 PIE TYPE                3  13.6  16.2
##  2 PIE TYPE                3  16.4  16.2
##  3 PIE TYPE                3  16.4  16.2
##  4 PIE TYPE                3  13.6  16.2
##  5 PIE TYPE                3  15.5  16.2
##  6 PIE TYPE                3  16.4  16.2
##  7 MINIATURE               1  34    34.2
##  8 MINIATURE               1  30    34.2
##  9 MINIATURE               1  30    34.2
## 10 MINIATURE               1  34    34.2
```

- Create a visualization of model performance

```
# Make a smoother scatter plot
ggplot(poly_var_results, aes(x = variety_integer, y = price))+
  geom_line(aes(y = .pred), color = "skyblue")+
  geom_point(color = "darkgreen")+
  labs(x = "Variety Integer Value",
       y = "Price ($)")+
  geom_smooth(method = lm, formula = y ~ poly(x, degree = 2), color = "midnightblue", size = 1.2, se =
```



Price ($)
40
30
20
10
0
1
2
3
Variety Integer Value