Curtin_Lab2

Charlie Curtin

01-18-24

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)

poly_wf
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
```

```
##
## * step_integer()
## * step_poly()
##
## -- Model -----------------------------------------------------------------------
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

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().

```
# evaluate model performance
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?

- By comparing the root mean-squared error and the R^2 value of our two different models, we can see that the polynomial model predicts pumpkin price better than the linear model. Our linear model has an RMSE of \$7.23, while our polynomial model has an RMSE of \$3.28, meaning our polynomial model has a smaller average difference between observed and predicted prices. The linear model has an R^2 value of .49, while our polynomial model has an R^2 value of .89, showing that our polynomial model has a better fit.

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

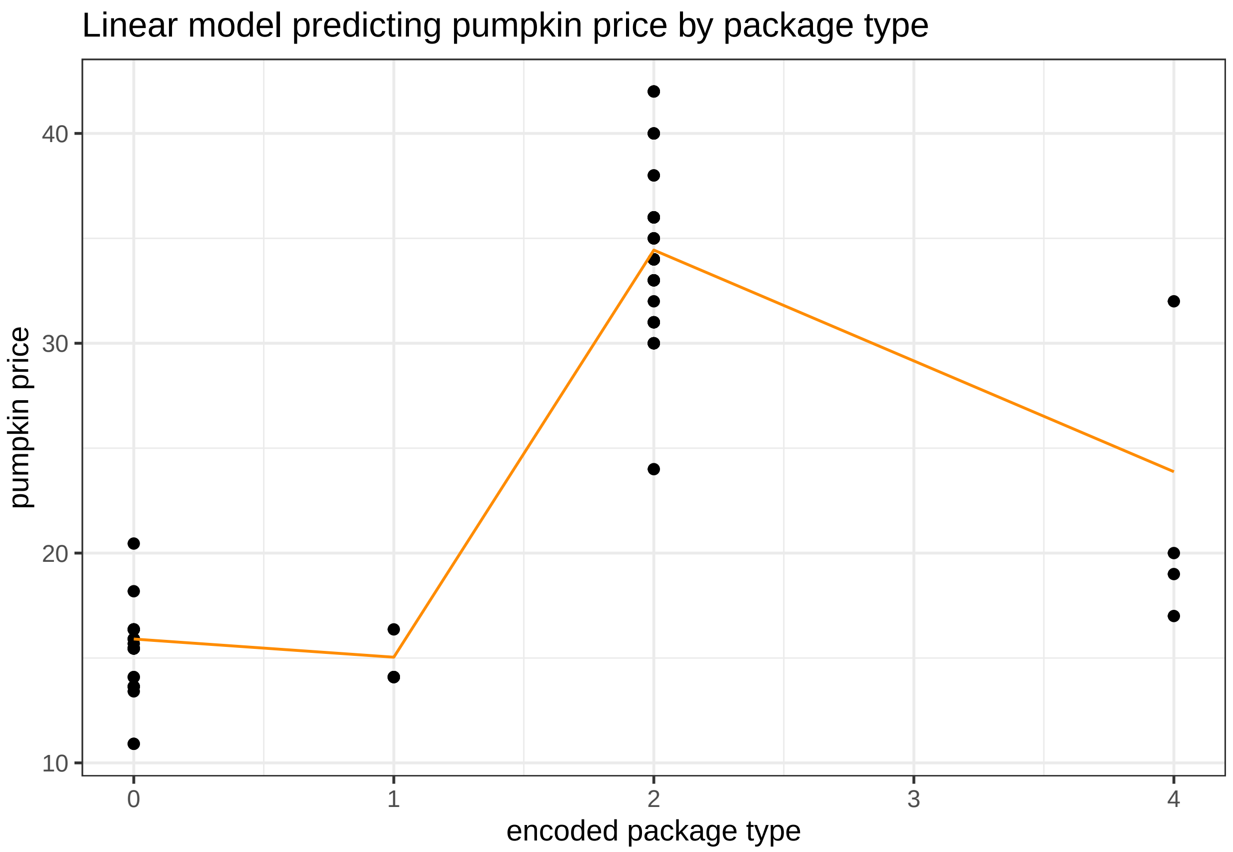
```
# Bind encoded package column to the results
poly_results <- poly_results %>%
  bind_cols(package_encode %>%
              rename(package_integer = package)) %>%
  relocate(package_integer, .after = package)

# 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.

```r
# Make a scatter plot
poly_results %>% ggplot() +
  geom_point(aes(x = package_integer, y = price)) +
  geom_line(aes(x = package_integer, y = .pred),
            color = "darkorange") +
  labs(x = "encoded package type",
       y = "pumpkin price",
       title = "Linear model predicting pumpkin price by package type") +
  theme_bw()
```

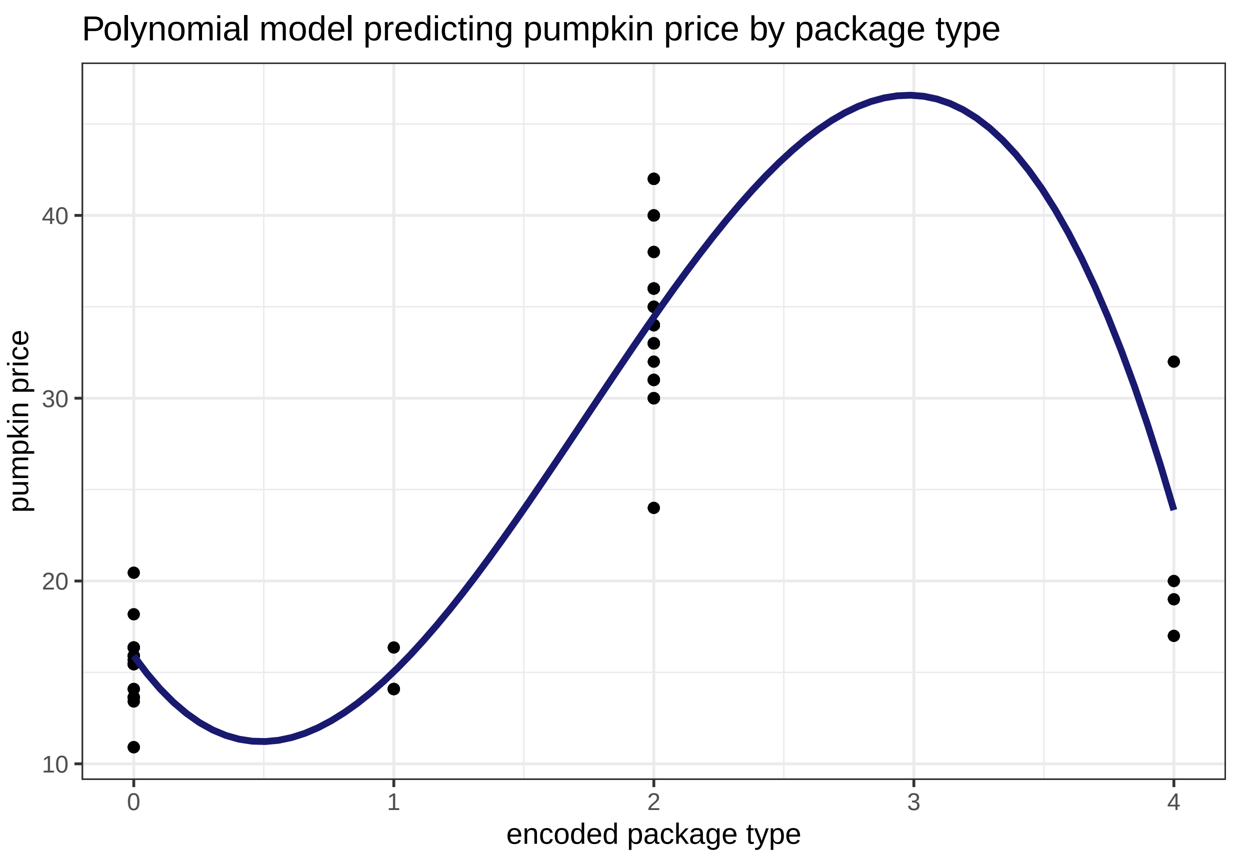


You can see that a curved line fits your data much better.

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)

```r
# Make a smoother scatter plot
poly_results %>% ggplot() +
  geom_point(aes(x = package_integer, y = price)) +
  geom_smooth(aes(x = package_integer, y = .pred),
              method = "lm",
              formula = y ~ poly(x, degree = 3),
              color = "midnightblue",
              size = 1.2,
              se = FALSE) +
  labs(x = "encoded package type",
       y = "pumpkin price",
       title = "Polynomial model predicting pumpkin price by package type") +
  theme_bw()
```

```
## 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. 7. Determine its correlation with the outcome variable (price). (Remember we calculated a correlation matrix last week) 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 - Evaluate model performance on the test data - Create a visualization of model performance

Predicting sales price based on month of the year sold

```r
# Find the correlation between the package and the price
print(cor(new_pumpkins$month, new_pumpkins$price))
```

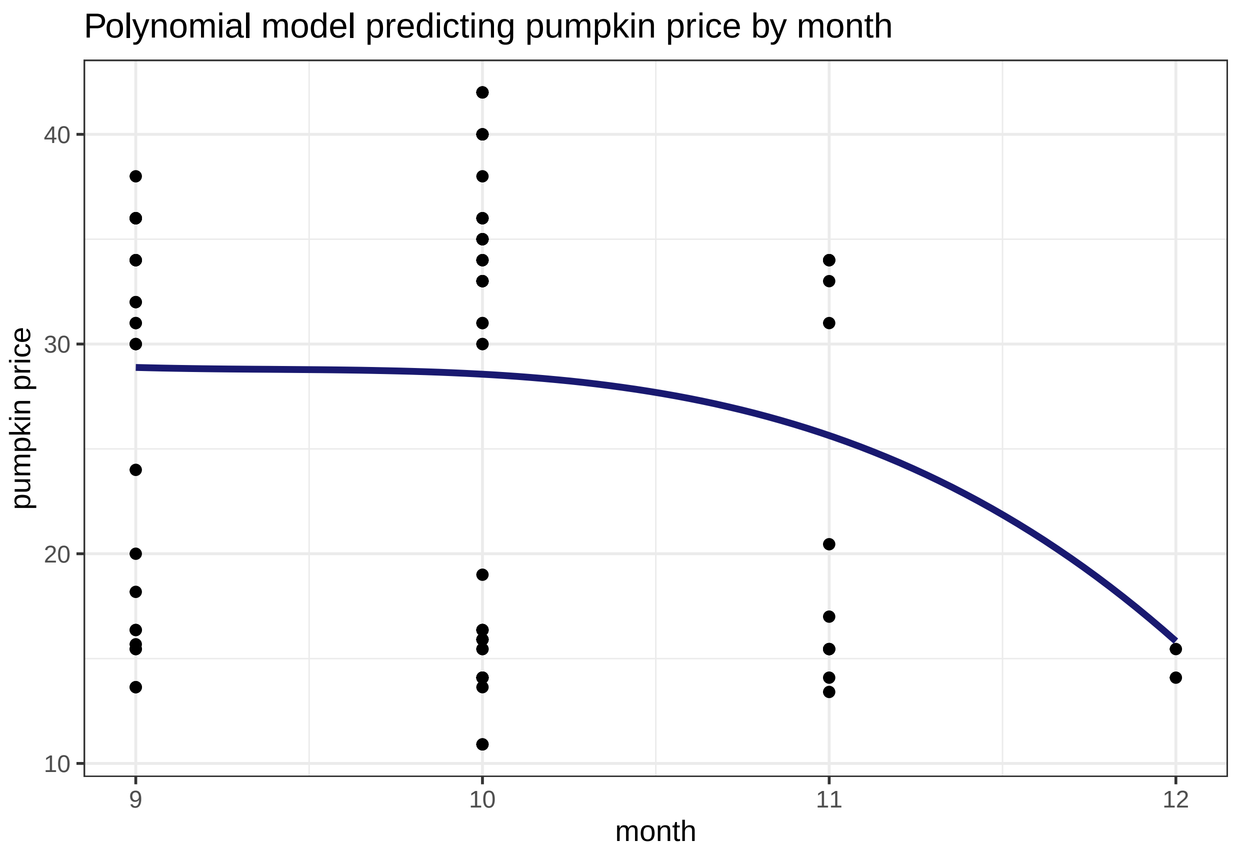
```
## [1] -0.1487829
```

```r
set.seed(123)

# Split the data into training and test sets
pumpkins_split <- new_pumpkins %>%
  initial_split(prop = 0.8)

# Extract training and test data
pumpkins_train <- training(pumpkins_split)
pumpkins_test <- testing(pumpkins_split)

# Create a recipe for preprocessing the data, specifying a polynomial
pump_month_recipe <- recipe(price ~ month, data = pumpkins_train) %>%
  step_poly(all_predictors(), degree = 4)

# Create a model
poly_month_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

# Bundle recipe and model spec into a workflow
poly_month_wf <- workflow() %>%
  add_recipe(pump_month_recipe) %>%
  add_model(poly_month_spec)

# Create a model
poly_month_wf_fit <- poly_month_wf %>%
  fit(data = pumpkins_train)

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

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

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      9.64
## 2 rsq     standard      0.0587
## 3 mae     standard      8.57
```

- According to the root-mean squared error, on average, our predictions of price are off by $9.64.

```r
# visualize model results
poly_month_results %>% ggplot() +
  geom_point(aes(x = month, y = price)) +
  geom_smooth(aes(x = month, y = .pred),
              method = "lm",
              formula = y ~ poly(x, degree = 3),
              color = "midnightblue",
              size = 1.2,
              se = FALSE) +
  labs(x = "month",
       y = "pumpkin price",
       title = "Polynomial model predicting pumpkin price by month") +
  theme_bw()
```



- Our model doesn't perform very well, appearing to underfit our data, leading to the high RMSE.

Lab 2 due 1/24 at 11:59 PM