Lab 2

Elke Windschitll

2023-01-18

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, so open and run your Lab 1 .Rmd as a first step so those objects are available in your Environment (unless you created an R Project last time, in which case, kudos to you!).

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 package to a numerical form, and then add a polynomial effect with step_poly()

Note: Sourcing from lab1

```
# 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? Choose another value for degree if you need to. 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 find the highest value for degree that is consistent with our data.

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 × 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 × 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?

The model from this week seems to be performing better than the model last week. We can see this from a lower rmse value (3.27 vs. 7.23 last week) and an rsq value (0.892 vs. 0.495 last week) closer to one, meaning that the correlation is higher here, and therefore better.

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

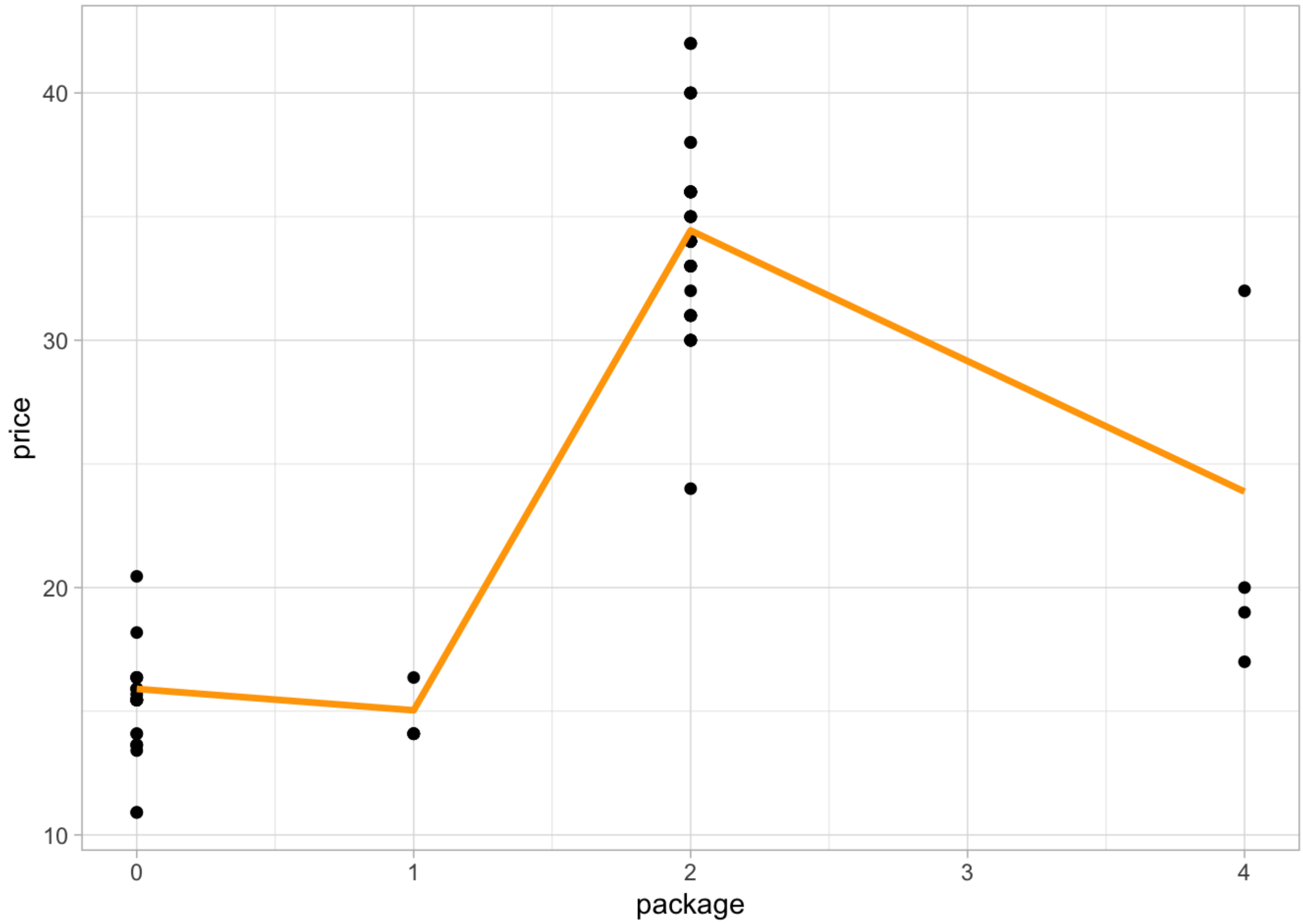
```
# Bind encoded package column to the results
poly_results <- poly_results %>%
  bind_cols(package_encode %>%
              rename(package_integer = package)) %>%
  relocate(package_integer, .after = package)
# Print new results data frame
poly_results %>%
  slice_head(n = 5)
```

```
## # A tibble: 5 × 4
##   package              package_integer price .pred
##   <chr>                          <dbl> <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.

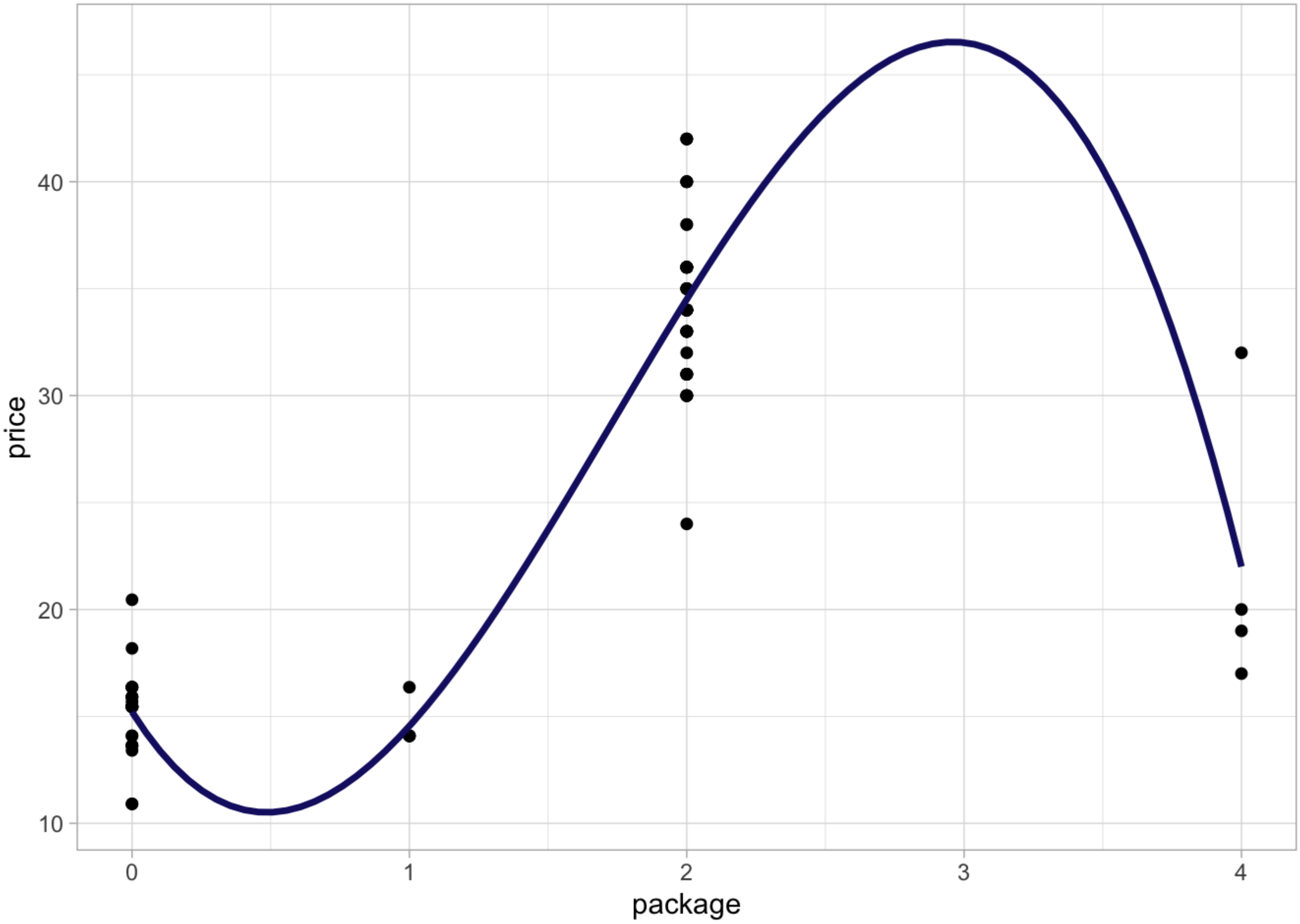
```
# Make a scatter plot
poly_results %>% ggplot(aes(x = package_integer, y = price)) +
  geom_point(size = 1.6) +
  geom_line(aes(y = .pred), color = "orange", size = 1.2) +
  xlab("package")
```



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
poly_results %>% ggplot(aes(x = package_integer, y = price)) +
  geom_point(size = 1.6) +
  geom_smooth(method = lm, formula = y ~ poly(x, degree = 3), color = "midnightblue", si
ze = 1.2, se = FALSE) +
  xlab("package")
```



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

```
cor(baked_pumpkins$variety, baked_pumpkins$price)
```

```
## [1] -0.863479
```

The correlation between price and variety is -0.86.

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

Lab 2 due 1/24 at 11:59 PM

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

```
# Encode variety column
variety_encode <- variety_pumpkins_recipe_encode_step %>%
  prep() %>%
  bake(new_data = pumpkins_test) %>%
  select(variety)
```

```
# Specify a recipe
variety_pumpkins_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 variety_spec
variety_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

# Bundle recipe and model spec into a workflow
variety_wf <- workflow() %>%
  add_recipe(variety_pumpkins_recipe) %>%
  add_model(variety_spec)

# Create a model
variety_wf_fit <- variety_wf %>%
  fit(data = pumpkins_train)

# Print learned model coefficients
variety_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
```

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

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

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

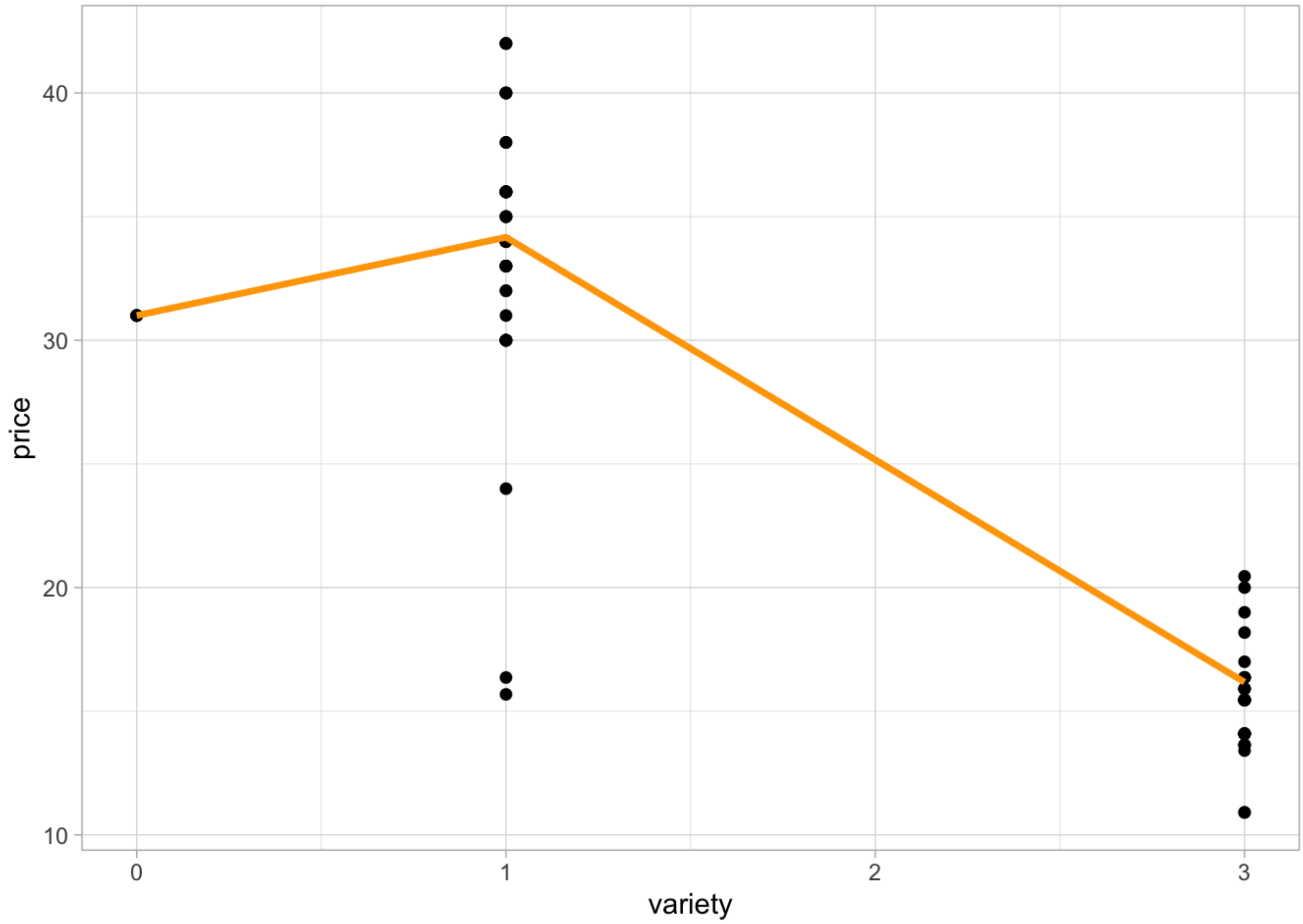
```
# Evaluate
metrics(data = variety_results, truth = price, estimate = .pred)
```

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

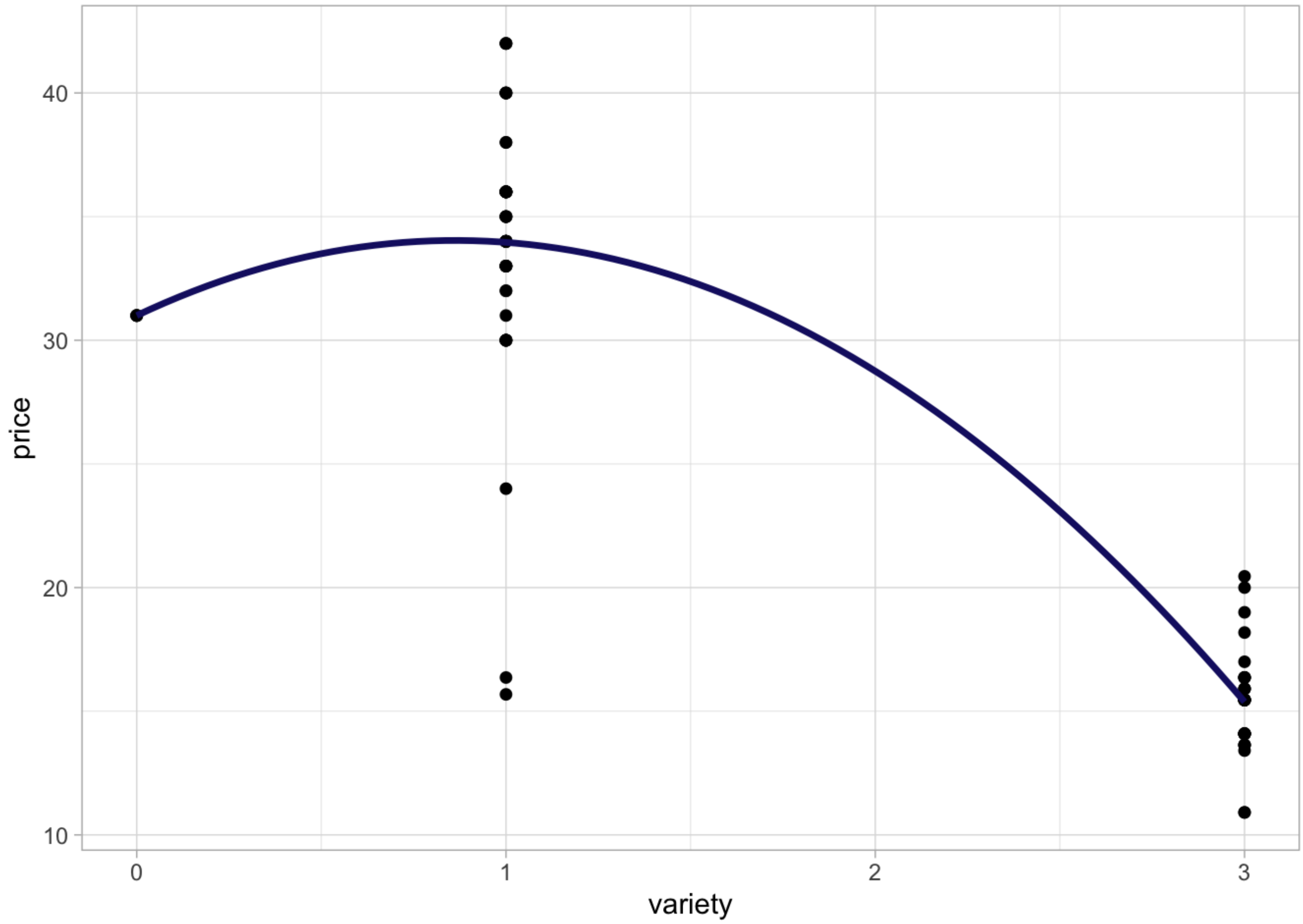
```
# Bind encoded variety column to the results
variety_results <- variety_results %>%
  bind_cols(variety_encode %>%
              rename(variety_integer = variety)) %>%
  relocate(variety_integer, .after = variety)

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

```
## # A tibble: 5 × 4
##   variety  variety_integer price .pred
##   <chr>              <dbl> <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
```

```
# Make a scatter plot
variety_results %>% ggplot(aes(x = variety_integer, y = price)) +
  geom_point(size = 1.6) +
  geom_line(aes(y = .pred), color = "orange", size = 1.2) +
  xlab("variety")
```




```
# Make a smoother scatter plot
variety_results %>% ggplot(aes(x = variety_integer, y = price)) +
  geom_point(size = 1.6) +
  geom_smooth(method = lm, formula = y ~ poly(x, degree = 2), color = "midnightblue", si
ze = 1.2, se = FALSE) +
  xlab("variety")
```



Based on my metrics, it appears packages is a “better” indicator of price than variety. The rmse is lower in the package model than the variety model, and the rsq is higher in the package model than the variety model.