Lab 2

Lewis White

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()

```
# 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) #step_poly creates new columns that are basis expansions of v
```

How did that work?

It prepares the data for analysis by specifying the outcome/predictor variables, changing the variables to integers to make analysis quicker, and specifies that we are focused on a fourth degree polynomial for the package variable.

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.

Since there are just 5 unique packages, I believe degree 4 (n-1) is the highest degree we can use.

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")
```

This specifies the type of analysis we want to perform and the package/system were using to perform the analysis.

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?

The new polynomial model performed better across all three metrics of interest. The RMSE for the new model was 3.27 compared to 7.23 in the linear one. The R squared value for our new model is .892, which means the model accounts for 89% of variability in the price. The linear model only accounted for 50% of the variability in price. The MAE was also reduced in the polynomial model.

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

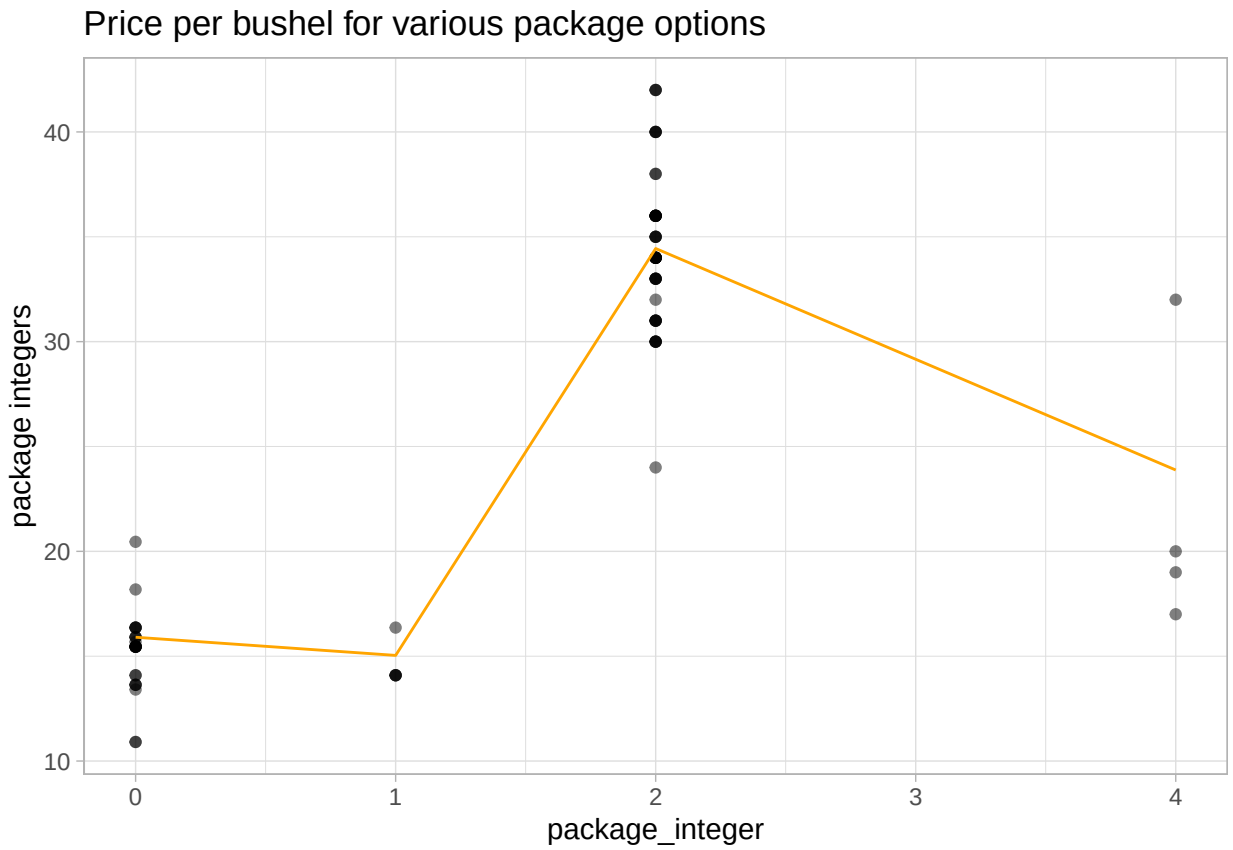
```
# Bind encoded package column to the results
poly_results <- poly_results %>%
  bind_cols(package_encode %>%
              rename(package_integer = package)) %>%
  relocate(package_integer, .after = package)

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

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

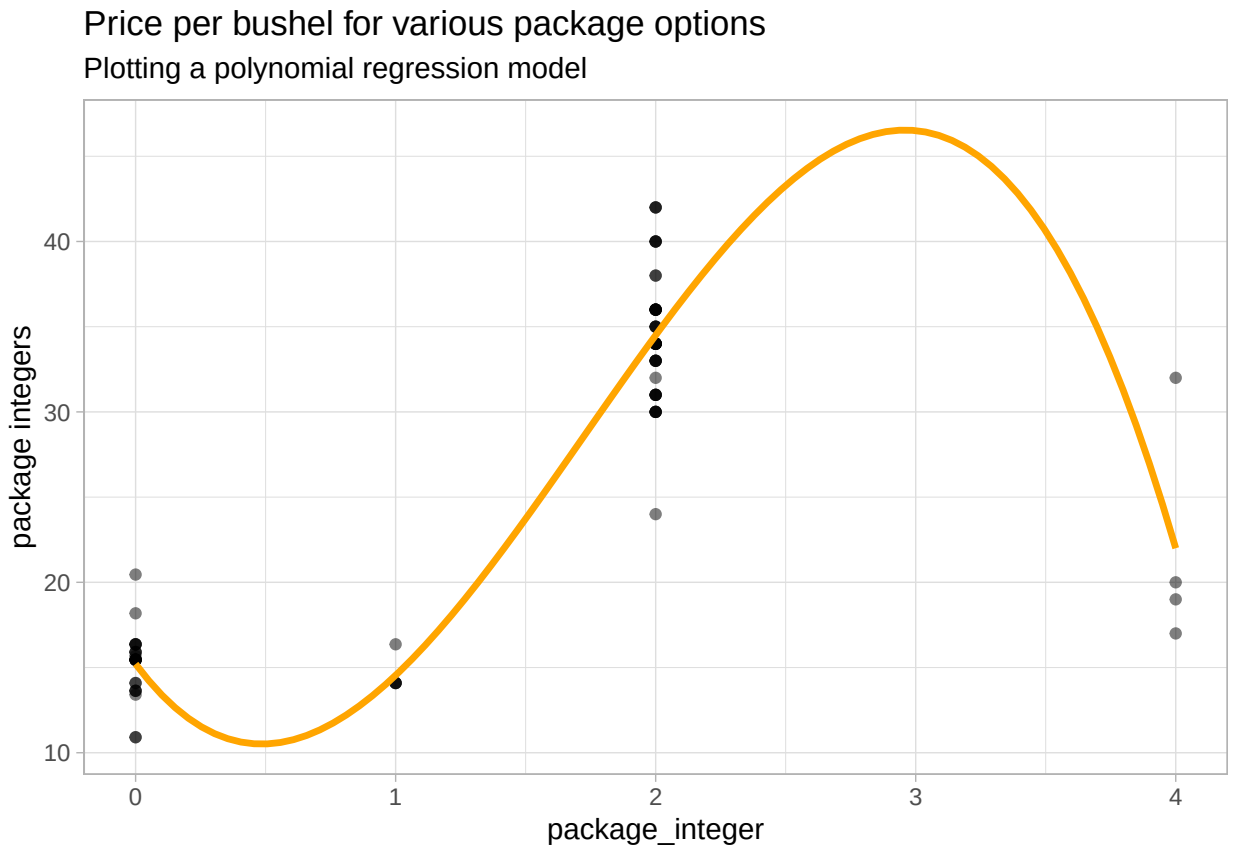
```
# Make a scatter plot
poly_results %>% ggplot(aes(x = package_integer, y = price)) +
  geom_point(alpha = 0.5) +
  geom_line(aes(y = .pred), color = "orange") +
  labs(y = "package integers",
       title = "Price per bushel for various package options")
```



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(alpha = 0.5) +
  geom_smooth(method = "lm", formula = y ~ poly(x, degree = 3), color = "orange", size = 1.2, se = FALSE
  labs(y = "package integers",
       title = "Price per bushel for various package options",
       subtitle = "Plotting a polynomial regression model")
```




```
#the degree needs to be 3 now, because the package with integer 3 did not exist in our test data set.
```

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.

I'm choosing the variety variable.

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

```
cor(baked_pumpkins$variety, baked_pumpkins$price) #-0.863479
```

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

The correlation between the pumpkin variety and price is -0.86, which is pretty strong in the negative direction.

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

Creating and evaluating the model

```
#creating the recipe
variety_recipe <- recipe(price ~ variety, data = pumpkins_train) %>% #specify model
  step_integer(all_predictors(), zero_based = TRUE) %>% #turns variables into integers
  step_poly(all_predictors(), degree = 3) #creates new columns for 3rd degree polynomials


#building the model specifications
poly_variety_spec <- linear_reg() %>% #specify linear regression
  set_engine("lm") %>%
  set_mode("regression")

# Bundle recipe and model spec into a workflow
poly_variety_wf <- workflow() %>%
  add_recipe(variety_recipe) %>%
  add_model(poly_variety_spec)

#fit the workflow
variety_fit <- poly_variety_wf %>%
  fit(data = pumpkins_train)

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

#check out the predicted results
variety_results %>%
  slice_head(n = 15)
```

```
## # A tibble: 15 x 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
## 11 MINIATURE  36    34.2
## 12 MINIATURE  36    34.2
## 13 PIE TYPE   13.4  16.2
## 14 MINIATURE  15.7  34.2
## 15 FAIRYTALE  31    31.0
```

```
#check out how the model performs according to metrics
metrics(data = variety_results, truth = price, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
```

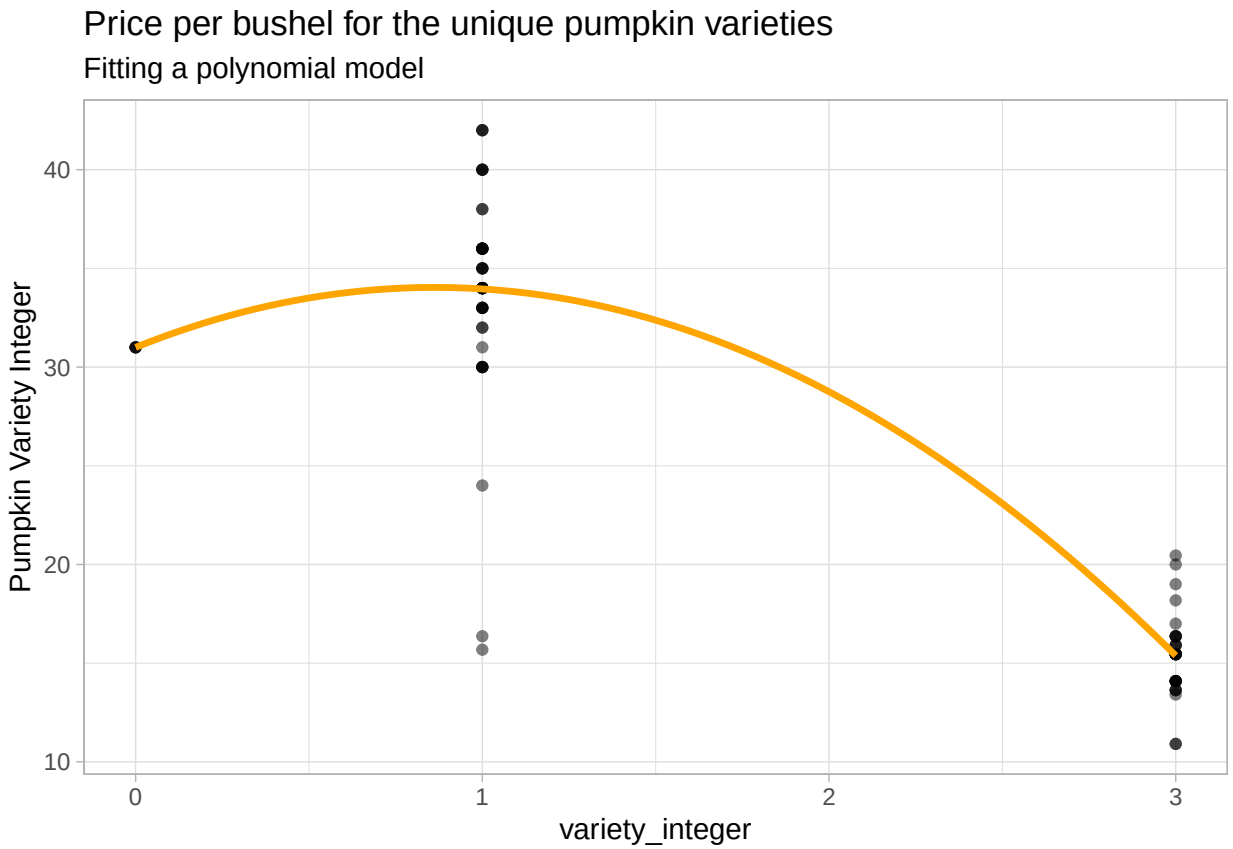
```
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       4.13
## 2 rsq     standard       0.827
## 3 mae     standard       2.52
```

Creating the plot

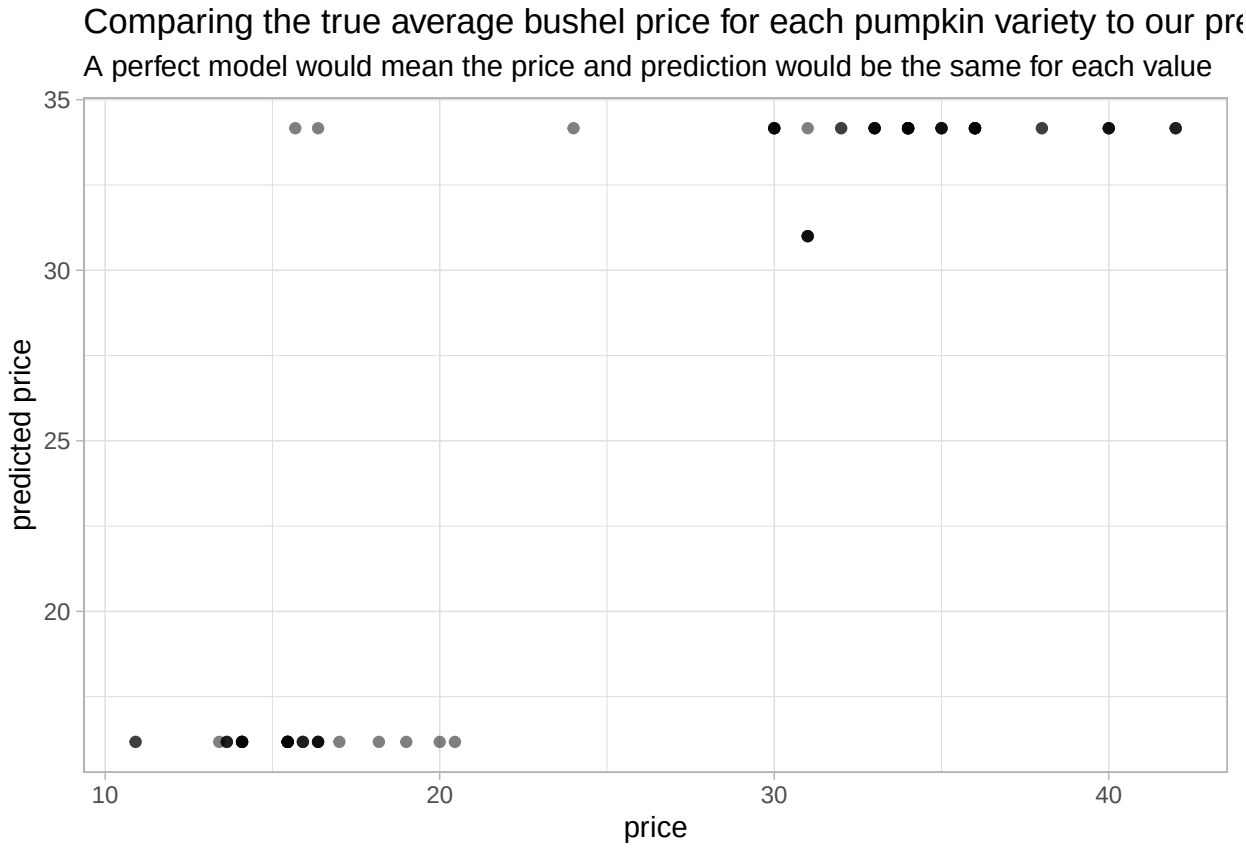
```
#create a simple regression so the variety can have integer values without the polynomials complicating
lm_pumpkins_variety_recipe <- recipe(price ~ variety, data = pumpkins_train) %>%
  step_integer(all_predictors(), zero_based = TRUE)

#encoding the simple regression so we just have the integer values left
variety_encode <- lm_pumpkins_variety_recipe %>%
  prep() %>%
  bake(new_data = pumpkins_test) %>%
  select(variety)

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


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

```
## # A tibble: 5 x 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
```

```
#visualizing the model performance
poly_results_variety %>% ggplot(aes(x = variety_integer, y = price)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", formula = y ~ poly(x, degree = 2), color = "orange", size = 1.2, se = FALSE
  labs(y = "Pumpkin Variety Integer",
       title = "Price per bushel for the unique pumpkin varieties",
       subtitle = "Fitting a polynomial model")
```




```
# another way of visualizing the results. This option compares the true price to the predicted price
ggplot(data = variety_results, aes(x = price, y = .pred)) +
  geom_point(alpha = 0.5) +
  labs(y = "predicted price",
       title = "Comparing the true average bushel price for each pumpkin variety to our predicted value
       subtitle = "A perfect model would mean the price and prediction would be the same for each value
```


Comparing the true average bushel price for each pumpkin variety to our pre
A perfect model would mean the price and prediction would be the same for each value
predicted price
35
30
25
20
10
20
30
40
price