Predicting (Diamond) Prices

AUTHOR
Zev van Zanten

Setup

Introduction

For the project, I sought to do something that would let me integrate data visualization, data science models, and dashboards.

R's diamonds dataset - tucked away in the ggplot2 package - contains the sale information of 53,940 different diamonds. Each sale contains information on a diamond's price (in USD), size (in carats), length, width, height, cut (a qualitative and cardinal measure), color, and clarity.

First I explored the data to understand what it looked like and to figure out what was most valuable to do with it. I focused my exploratory analysis on patterns in price, color, cut, carats, and clarity as well as regressions between price and these specific variables.

The end result of this exploratory analysis was that there were a number of interesting patterns in the attributes of diamonds sold on the market. However, the most valuable (and interesting) thing was that a simple regression built with just four variables had a remarkable R^2 of around 91%. This meant I could use the data to provide phenomenal predictive power. And each of the regression models I tested - which allowed us to envision different relationships and account for things like covariance - had a slightly different perspective.

Seeing this, it felt right to focus on creating an interactive visual display using a dashboard to give people information on the price of a specific diamond (as well as some other statistics for diamonds of this type).

```
#Loading data
data("diamonds")

diamonds_named <- diamonds
```

```r
#Getting basic look at data
print(diamonds)
```

```
# A tibble: 53,940 × 10
   carat cut       color clarity depth table price     x     y
z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl>
<dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98
2.43
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84
2.31
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07
2.31
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23
2.63
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35
2.75
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96
2.48
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98
2.47
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11
2.53
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78
2.49
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05
2.39
# ℹ 53,930 more rows
```

```r
#Summary stats for diamonds
summary(diamonds)
```

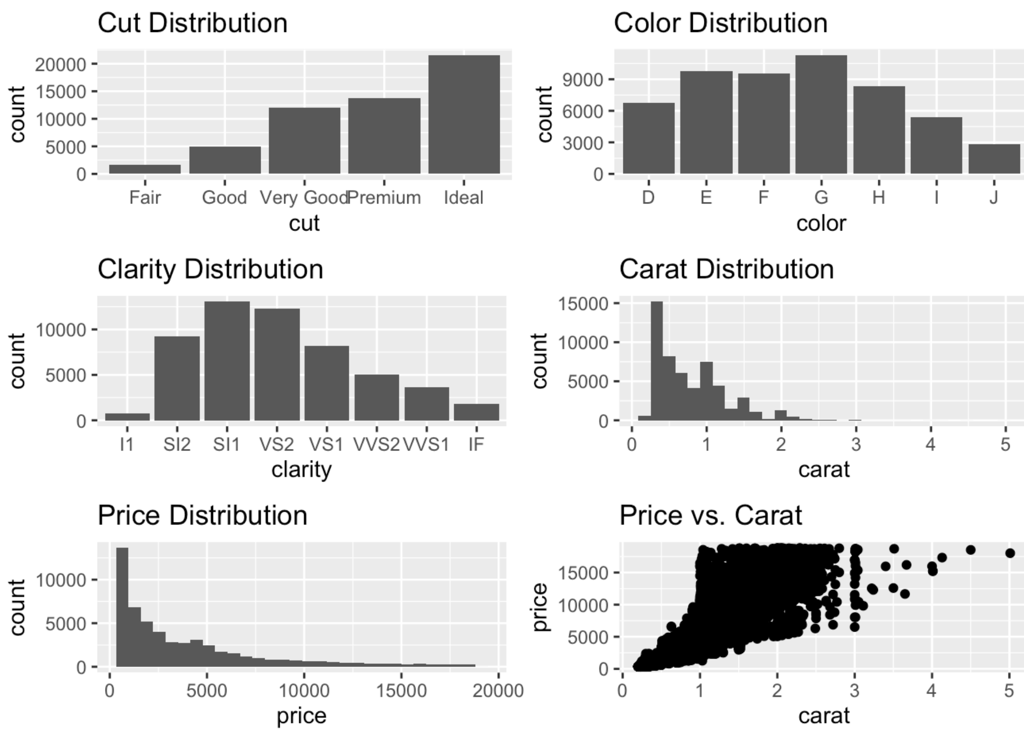
```
     carat               cut        color        clarity
depth
 Min.   :0.2000   Fair     : 1610   D: 6775   SI1    :13065
Min.   :43.00
 1st Qu.:0.4000   Good     : 4906   E: 9797   VS2    :12258
1st Qu.:61.00
 Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194
Median :61.80
 Mean   :0.7979   Premium  :13791   G:11292   VS1    : 8171
Mean   :61.75
```

```
 3rd Qu.:1.0400   Ideal    :21551   H: 8304   VVS2   : 5066
3rd Qu.:62.50
 Max.   :5.0100                     I: 5422   VVS1   : 3655
Max.   :79.00
                                    J: 2808   (Other): 2531
     table           price             x               y
 Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   :
0.000
 1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.:
4.720
 Median :57.00   Median : 2401   Median : 5.700   Median :
5.710
 Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   :
5.735
 3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.:
6.540
 Max.   :95.00   Max.   :18823   Max.   :10.740   Max.
:58.900

       z
 Min.   : 0.000
 1st Qu.: 2.910
 Median : 3.530
 Mean   : 3.539
 3rd Qu.: 4.040
 Max.   :31.800
```

```r
#Visualizations of distributions
# build each plot and give it a title
p1 <- ggplot(diamonds, aes(cut)) + geom_bar()           + ggtit
p2 <- ggplot(diamonds, aes(color)) + geom_bar()         + ggtit
p3 <- ggplot(diamonds, aes(clarity)) + geom_bar()       + ggtit
p4 <- ggplot(diamonds, aes(carat)) + geom_histogram()   + ggtit
p5 <- ggplot(diamonds, aes(price)) + geom_histogram()   + ggtit
p6 <- ggplot(diamonds, aes(carat, price)) + geom_point() + ggti

# stitch them into 2 columns, 3 rows
( p1 + p2 ) /
( p3 + p4 ) /
( p5 + p6 )
```

```
`stat_bin()` using `bins = 30`. Pick better value with
```

```
`binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```




```
#Creating regressions
#Linear regression model
lm_model <- lm(price ~ cut + color + clarity + carat, data = dia

#Defining x and y so the next three regressions can work
x <- model.matrix(price ~ cut + color + clarity + carat, data =
y <- diamonds_named$price


#Classic ridge model to penalize less valuable models using the
ridge_model <- cv.glmnet(x, y, alpha = 0)

#Classic lasso model to take an alternate approach to building a
lasso_model <- cv.glmnet(x, y, alpha = 1)

#Classic Elastic Regression to leverage the best of the lasso an
elastic_net_model <- cv.glmnet(x, y, alpha = 0.5)

#Classic polynomial model to see if that better fits the relatio
diamonds_named$carat2 <- diamonds_named$carat^2
```

```
poly_model <- lm(price ~ cut + color + clarity + carat + carat2

# Defining a function to calculate R^2 for three of my regressio
r2 <- function(actual, predicted) {
  1 - sum((actual - predicted)^2) / sum((actual - mean(actual))
}

# Computing R² for each model

lm_r2 <- summary(lm_model)$r.squared
poly_r2 <- summary(poly_model)$r.squared
ridge_r2 <- r2(y, predict(ridge_model, newx = x, s = "lambda.mi
lasso_r2 <- r2(y, predict(lasso_model, newx = x, s = "lambda.mi
enet_r2 <- r2(y, predict(elastic_net_model, newx = x, s = "lamb

# Combining all R² values into a list
r2_results <- list(
  Linear      = lm_r2,
  Polynomial  = poly_r2,
  Ridge       = ridge_r2,
  Lasso       = lasso_r2,
  ElasticNet  = enet_r2
)

# Printing the R^2 results
print(r2_results)
```

```
$Linear
[1] 0.9159406

$Polynomial
[1] 0.9169638

$Ridge
[1] 0.9027132

$Lasso
[1] 0.9158972

$ElasticNet
[1] 0.9159011
```

Building the Dashboard

After settling on a dashboard for predicting the price of diamonds, I thought about what this dashboard should look like. My central idea was that individuals should be able to input data to get a price as close to their situation as possible (meaning they should be able input carat, cut, color, and clarity.) I also wanted to help them understand the general data they were looking at, as I knew that just getting a dollar value would not be the most helpful. As a result, I decided that rather than just getting an outputted number, the users should get the R^2 value of the model they picked, the scatterplot itself, see where their point is within said scatterplot, and see the regression line.

I decided to include each of the five regressions as they were relatively similar in predictive power and could satisfy different needs/concerns of the user. The straightforward linear model gave them an idea of what their diamond was worth, while the Ridge, Lasso, and Elastic Net allowed them to avoid potential overfitting (if they were afraid of them.) And a polynomial model let them see how a different approach to the data changed the price.

To build the dashboard, I wound up taking an iterative approach that focused on building up the individual components and then stitching them together. First, I selected and constructed the different regressions and data visualizations as they were the basis of the entire dashboard. This was primarily regressions, though I also needed a scatterplot, labeled points, and a line.

After this, I identified the different filters and functions I would need to handle all the desired outputs and built them one by one. First, I built something that filtered all of the data according to specifications. Then I built a different function that would construct lines and other features from key inputs and another function that would extract R^2.

With all the sub components built and ready, I stitched them together in a ShinyR database, using reactive functions and buttons to trigger most of them and finding opportunities to reduce code length and complexity whenever possible. I also stylized the overall dashboard using cards, HTML code and other odds and ends.

```
# Loading the diamonds dataset
```

```
data("diamonds")
diamonds_named <- diamonds

# Preparing the matrix to fit the glmnet models
x <- model.matrix(price ~ cut + color + clarity + carat, data =
y <- diamonds_named$price

#Creating the linear model
lm_model <- lm(price ~ cut + color + clarity + carat, data = di
lm_r2 <- summary(lm_model)$r.squared

#Creating the Polynomial regression model (degree 2 on carat)
diamonds_named$carat2 <- diamonds_named$carat^2
poly_model <- lm(price ~ cut + color + clarity + carat + carat2
poly_r2 <- summary(poly_model)$r.squared

#Creating a Ridge regression model
ridge_model <- cv.glmnet(x, y, alpha = 0)
ridge_preds <- predict(ridge_model, newx = x, s = "lambda.min")
ridge_r2 <- 1 - sum((y - ridge_preds)^2) / sum((y - mean(y))^2)

#Creating a Lasso regression model
lasso_model <- cv.glmnet(x, y, alpha = 1)
lasso_preds <- predict(lasso_model, newx = x, s = "lambda.min")
lasso_r2 <- 1 - sum((y - lasso_preds)^2) / sum((y - mean(y))^2)

#Creating an Elastic Net regression model
elastic_net_model <- cv.glmnet(x, y, alpha = 0.5)
elastic_preds <- predict(elastic_net_model, newx = x, s = "lamb
elastic_r2 <- 1 - sum((y - elastic_preds)^2) / sum((y - mean(y)


# stash them in a named list for easy UI display
r2_values <- list(
  "Linear Regression"     = lm_r2,
  "Polynomial Regression" = poly_r2,
  "Ridge Regression"      = ridge_r2,
  "Lasso Regression"      = lasso_r2,
  "Elastic Net"           = elastic_r2
)

#Shiny UI helper function to make it easier to create all of the

card_layout <- function(name, text, img) {
  #Specifying dimensions and other key features
```

```
  card(
   layout_column_wrap(
    width = 1/2,
    responsive = TRUE,
    card_body(
      tags$p(tags$strong(name), text)
    ),
    img(src = img, style = "width:100%; height:auto;")
  ),
  class = "mb-4"
 )
}



#Creating the UI

ui <- fluidPage(
  #Title panel
  titlePanel("What's My Diamond Worth?", windowTitle = "Diamond
  tags$head(
    tags$style(HTML("
      body {
        font-family: 'Arial', sans-serif;
        background-color: #f4f4f9;
      }
      .container {
        max-width: 1200px;
        margin: 0 auto;
      }
      .panel {
        background-color: white;
        padding: 20px;
        box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
        border-radius: 8px;
      }
      .btn-primary {
        background-color: #007BFF;
        color: white;
        border: none;
      }
      .btn-primary:hover {
        background-color: #0056b3;
      }
      h1 {
```

```
      font-size: 30px;
      font-weight: 700;
      color: #2c3e50;
    }
    h3 {
      font-size: 22px;
      color: #34495e;
    }
    .text-muted {
      color: #7f8c8d;
    }
    .output {
      margin-top: 20px;
      font-size: 18px;
      color: #2c3e50;
    }
    .predictionPlot {
      width: 100%;
      max-height: 500px;
      margin-top: 20px;
    }
    .card {
      padding: 15px;
      border: 1px solid #e1e1e1;
      border-radius: 8px;
      background-color: #ffffff;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
      margin-bottom: 20px;
    }
    .card-header {
      font-weight: bold;
      background-color: #f8f8f8;
      padding: 10px;
    }
  "))
),

#Creating a layout for the sidebar, complete with choices and
sidebarLayout(
  sidebarPanel(
    class = "panel",
    # user picks diamond specs
    div(class = "card",
        div(class = "card-header", "Diamond Characteristics")
        selectInput("cut_predict", "Cut:", choices = levels(d
```

```
        selectInput("color_predict", "Color:", choices = leve
        selectInput("clarity_predict", "Clarity:", choices = 
        numericInput("carat_predict", "Carat:", value = 0.5, r
      ),
      # user picks model
      div(class = "card",
        div(class = "card-header", "Select Model Type"),
        selectInput("model_type", "Model Type:",
                    choices = names(r2_values),
                    selected = "Linear Regression")
      ),

      # only want confidence interval for linear/polynomial

      #Adding a conditional panel that lets users select a conf

      conditionalPanel(
        condition = "input.model_type == 'Linear Regression' ||
        sliderInput(
          "conf_level", "Confidence level:",
          min = 0.5, max = 0.99, value = 0.95, step = 0.01
        )
      ),

      #Enter buttons
      actionButton("predictButton", "Predict Price",
                   style = "background:transparent; border:2px s
      actionButton("infoButton", "More Info About Variables",
                   style = "background:transparent; border:2px s
    ),

    #The main panel for displaying stuff
    mainPanel(
      class = "container",
      h1("Diamond Price Prediction App"),
      textOutput("predictedPrice"), #Prediction
      textOutput("r2Text"), #R2 score
      div(class = "predictionPlot", plotOutput("predictionPlot"
    )
  )
)

#Creating the service side
server <- function(input, output) {
  # grab inputs when button clicked
```

```
predict_inputs <- eventReactive(input$predictButton, {
  list(
    cut     = input$cut_predict,
    color   = input$color_predict,
    clarity = input$clarity_predict,
    carat   = input$carat_predict,
    model   = input$model_type
  )
})
# calculate the predicted price based on chosen model
predicted_price <- eventReactive(input$predictButton, {
  inp <- predict_inputs()
  # build a tiny df for predict()
  df <- data.frame(
    cut     = factor(inp$cut,    levels = levels(diamonds_name
    color   = factor(inp$color,  levels = levels(diamonds_name
    clarity = factor(inp$clarity,levels = levels(diamonds_name
    carat   = inp$carat
  )
  df$carat2 <- df$carat^2 # only poly needs this
  x_new     <- model.matrix(~ cut + color + clarity + carat, 

  #Allowing switching between models
  switch(inp$model,
    "Linear Regression"     = predict(lm_model, newdata = df)
    "Polynomial Regression" = predict(poly_model, newdata = d
    "Ridge Regression"      = as.numeric(predict(ridge_model,
    "Lasso Regression"      = as.numeric(predict(lasso_model,
    "Elastic Net"           = as.numeric(predict(elastic_net_
  )
})

# build a grid for plotting line + intervals

fit_df <- eventReactive(input$predictButton, {
  inp <- predict_inputs()
  grid <- seq(0.2, 5, length.out = 100)
  df <- data.frame(
    cut     = factor(inp$cut, levels = levels(diamonds_named$
    color   = factor(inp$color, levels = levels(diamonds_named
    clarity = factor(inp$clarity, levels = levels(diamonds_nar
    carat   = grid
  )
  df$carat2 <- df$carat^2
  xg <- model.matrix(~ cut + color + clarity + carat, data = 
```

```
  if (inp$model %in% c("Linear Regression", "Polynomial Regres
    # prediction intervals if for linear regression or polynon
    fit_obj <- if (inp$model == "Linear Regression") lm_model
    ci <- predict(fit_obj, newdata = df, interval = "predictio
    data.frame(carat = grid, fit = ci[,"fit"], lwr = ci[,"lwr"
  } else {
    preds <- switch(inp$model,
      "Ridge Regression" = as.numeric(predict(ridge_model, nev
      "Lasso Regression" = as.numeric(predict(lasso_model, nev
      "Elastic Net"      = as.numeric(predict(elastic_net_mode
    )
    data.frame(carat = grid, fit = preds, lwr = preds, upr = p
  }
})


#Outputting a predicted price

output$predictedPrice <- renderText({
  price <- predicted_price()
  paste0("The predicted price is: $", round(price, 2))
})

#Outputting a predicted r^2

output$r2Text <- renderText({
  inp <- predict_inputs()
  r2 <- r2_values[[inp$model]]
  paste0("R² for ", inp$model, ": ", round(r2, 4))
})


#Rendering a plot for the outputted regression line and plot

output$predictionPlot <- renderPlot({
  df_line <- fit_df()
  inp <- predict_inputs()
  pred_val <- predicted_price()
  #Creating the different parts of the regression line output
  ggplot() +
    geom_point(data = diamonds_named, aes(x = carat, y = price
    geom_ribbon(data = df_line, aes(x = carat, ymin = lwr, yma
    geom_line(data = df_line, aes(x = carat, y = fit), size =
    annotate("point", x = inp$carat, y = pred_val, color = "re
    labs(title = paste("Predicted Price Using", inp$model), x
```

```
      theme_minimal()
  })


  #Creating a more info button for interested users.

  observeEvent(input$infoButton, {
    showModal(modalDialog(
      title = "Variable Dictionary", size = "l", easyClose = TRU
      card_layout("Cut", "refers to how well the diamond has bee
        "https://external-content.duckduckgo.com/iu/?u=https%3A%
      card_layout("Color", "refers to the absence of color in th
        "https://external-content.duckduckgo.com/iu/?u=https%3A%
      card_layout("Clarity", "measures the presence of internal
        "https://external-content.duckduckgo.com/iu/?u=https%3A%
      card_layout("Carat", "measures the weight of the diamond.
        "https://external-content.duckduckgo.com/iu/?u=https%3A%
      footer = modalButton("Close")
    ))
  })
}

#Putting it all together.
shinyApp(ui = ui, server = server)
```

Shiny applications not supported in static R Markdown documents

Methods / Implementation

When building the dashboard, I chose to rely on Shiny as it was familiar to me. Regression models, too, were something primarily chosen because they were familiar enough to work with yet remained novel. The various bells and whistles were to make the dashboard more engaging and educational.

I also decided to use the built in linear regression models from R and the other 4 models from the glmnet package.

As soon as the code starts to run, it creates a model matrix x with all the predictors, and a response vectore y for the estimated price of a diamond. Then it fit sthe regression models for an Ordinary least squares (linear), Polynomial (adds a $carat^2$ term), Ridge (L2-penalized), Lasso (L1-penalized), and Elastic Net (mix of L1 & L2) model types. Next it gets the r^2 or psuedo r^2 values. Then it stores all of the models r^2 values before creating and using a helper function for the card layout that would display all the helper information and building the ui with a side bar layout containing all the needed inputs including a slider for a confidence interval that only appeared if you were using OLS or Polynomial models. Then when “Predict Price” is clicked, packages the user’s specs into a tiny data frame, chooses the right model, and returns a single price prediction. When this happens the code also creates a fitting grid by generating a sequence of carat values (0.2–5.0), using the selected model to predict over that grid (with prediction intervals for lm’s), and then building a data frame (df_line) for plotting the fit and ribbon. Lastly the site renders the the plot output with data point and line of best fit.

I decided to put the entire dashboard inside of a single chunk (rather an R file or some other form of storage) so that it was easily accessible. To reproduce the results/to see the work, you can run either the chunk I used to test various things or the dashboard itself.

Discussion & Conclusion

I envision this project having a number of helpful use cases and as a way to bring more clarity to the diamond market. For occasional shoppers who don't usually engage in the diamond market but may do so occasionally (one or two times in their lifetime), this model helps them in negotiations. With a better understanding of what a diamond should be worth, they can counter the informational advantages that would previously put them at a disadvantage in the market. For example, something seeking to sell a diamond they inherited or a person looking to propose to their partner can get a better understanding of what they should ask for or what they should offer respectively. Even for individuals with more experience transacting in the diamond market (i.e., diamond merchants and jewelers), having a predictive model on hand can help them make better offers and make wiser choices of what prices to buy and sell for. Overall, my predictive model should be a powerful source of information that corrects potential flaws in the market.

My analysis (and the model itself) shows that the vast majority of variety in diamond prices (about 90% across the five models) can be attributed to four simple factors: carat (size), color, cut, and clarity. This is incredibly valuable for anyone interested in understanding what affects the prices of diamonds and could even be extrapolated to help understand the pricing of jewelry as a whole (or anything that contains diamonds) by helping them understand some of the factors influencing pricing. It also has value as something to help us understand what influences pricing.

In the end, I chose to make this because I was interested in the intersection of data science, data communication, and economics/business and saw this as the perfect way to work at the intersection of those interests while using a dataset that was accessible to everyone. It taught me a lot about the best practices in data communication (I spent hours experimenting with different dashboards) and data analysis (I spent a long time looking at and trying out different visualizations, and regressions). It also taught me more about best practices in writing and documenting code and in product (and project) management.