SYMBIOSIS
INSTITUTE OF GEOINFORMATICS
॥वसुधैव कुटुम्बकम्॥

Diamond Pricing - Project Report

Aatreyi Dutt - 24070243059
Python for Data Science
November 2024

Introduction

The quality of a diamond is determined by its 4Cs[1] - cut, clarity, color, and carat. While carat is calculated using the weight of the diamond, the first three Cs are categorical variables. To understand these variables and their relationship to the price of the diamond better, I will be using a diamond pricing dataset from Kaggle to map out the spread of values and their relationships with price. In order to calculate and visualize these results Python's data processing and visualization libraries `pandas` (as `pd`), `numpy` (as `np`), `matplotlib.pyplot` (as `plt`), and `seaborn` (as `sb`).

Data and Preprocessing

I have taken data from an open source dataset on Kaggle[2]. After reading the raw csv file into python as "`data`" using the `pandas` library, I found that the raw dataset contains 53,940 rows and 10 columns, namely:

1. `carat` *the weight of the diamond*
2. `cut` *quality of the cut; from Fair (worst) to Ideal (best)*
3. `color` *transparentness of the diamond; from J (yellowish) to D (clear)*
4. `clarity` *fogginess of the diamond; from I1 (worst) to IF (best)*
5. `depth` *calculated using x, y, z dimensions of diamond*
6. `table` *width of top of diamond compared to the widest part*
7. `price` *price of the diamond in USD*
8. `x` *width of diamond in mm*
9. `y` *breadth of diamond in mm*
10. `z` *height of diamond in mm*

I fetched the first 5 rows of data using `pandas`' `data.head()` callable function:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

And had a look at the column-wise statistics using `data.describe()` to find that there were some zero values in the x, y, z columns - a value that should not be possible for the dimension of a diamond.

	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

As such, I removed all the rows from the dataframe that contained any zero valued column using `data = data[(data != 0).all(axis=1)]` in order to clean the data up before I started processing it. After removing rows with 0 values in it, the data size reduced to 54,920 rows.

Then, as the final preprocessing step, I found the unique values for categorical variables `cut`, `color`, `clarity`, and `color` using the `.unique()` function and manually label encoded them using `.replace()` like so:

Types of diamond cuts

```
data.cut.unique()
```

✓ 0.0s

```
array(['Ideal', 'Premium', 'Good', 'Very Good', 'Fair'], dtype=object)
```

Manual Label Encoding of *cut* attribute

```
data.cut = data.cut.replace(['Ideal', 'Premium', 'Very Good', 'Good', 'Fair'], [5,4,3,2,1])
#                           best                                              worst
```

✓ 0.0s

Now that the data has been cleaned and the columns have been label encoded, the data ready for processing, and now looks like this:

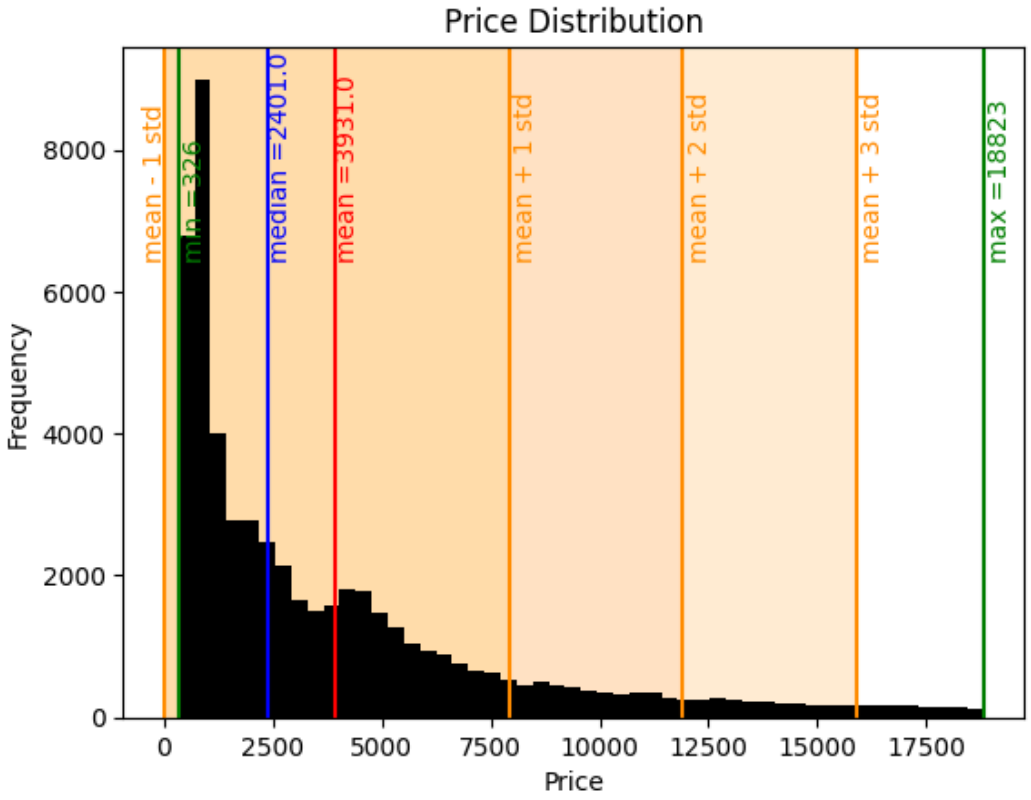
	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	5	6	2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	4	6	3	59.8	61.0	326	3.89	3.84	2.31
2	0.23	2	6	5	56.9	65.0	327	4.05	4.07	2.31
3	0.29	4	2	4	62.4	58.0	334	4.20	4.23	2.63
4	0.31	2	1	2	63.3	58.0	335	4.34	4.35	2.75

Feature Analysis

1. Price

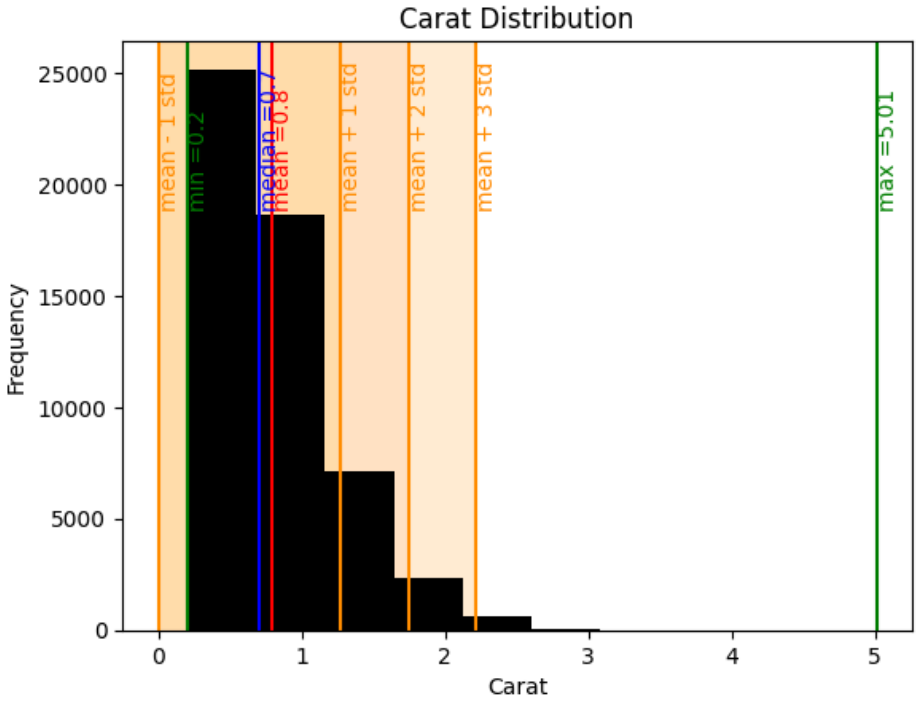
The prices of diamonds in this dataset range from $326 to $18,823, with the average price at $3,931. These statistics were fetched using pandas' `.describe()` function for dataframe columns. The same have then been plotted using matplotlib's `.hist()` function. Some insights from this graph are:

- The mean is greater than the median due to few diamonds with high prices.
- 69% of the prices lie between $0 and $7,918 (mean - σ to mean + σ, where σ is standard deviation).
- The standard deviation is high, implying a high variation in prices as confirmed by the wide orange regions in the graph.



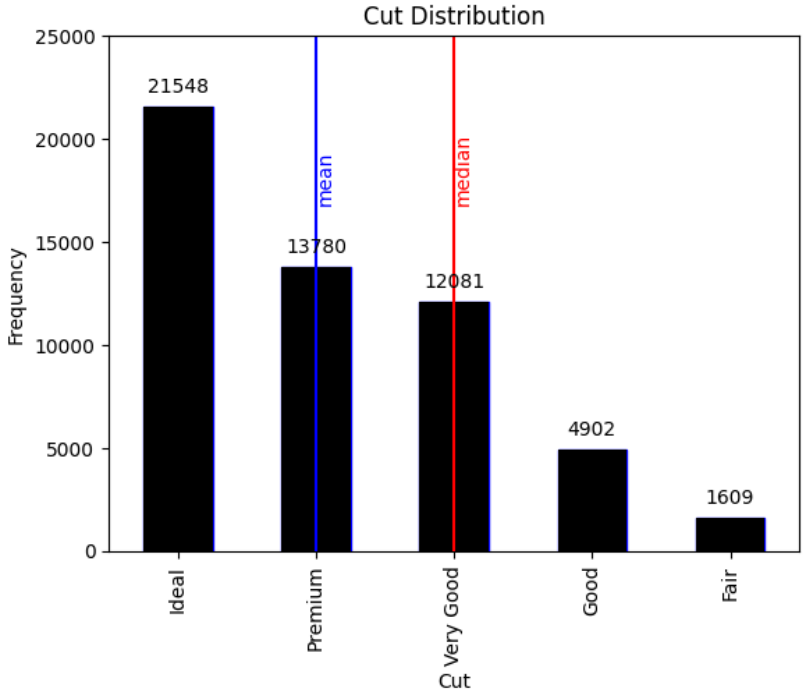
2. Carat

The carat of the diamonds range from 0.2 to 5.01, with the mean at 0.8ct and median at 0.7ct. The standard deviation is 0.47ct which means the carat values of diamonds in this set are close together with less variation, supported by the cluster of values for diamonds below ~2.6ct with rare, high carat values as outliers. 1 carat is equal to 200 mg.



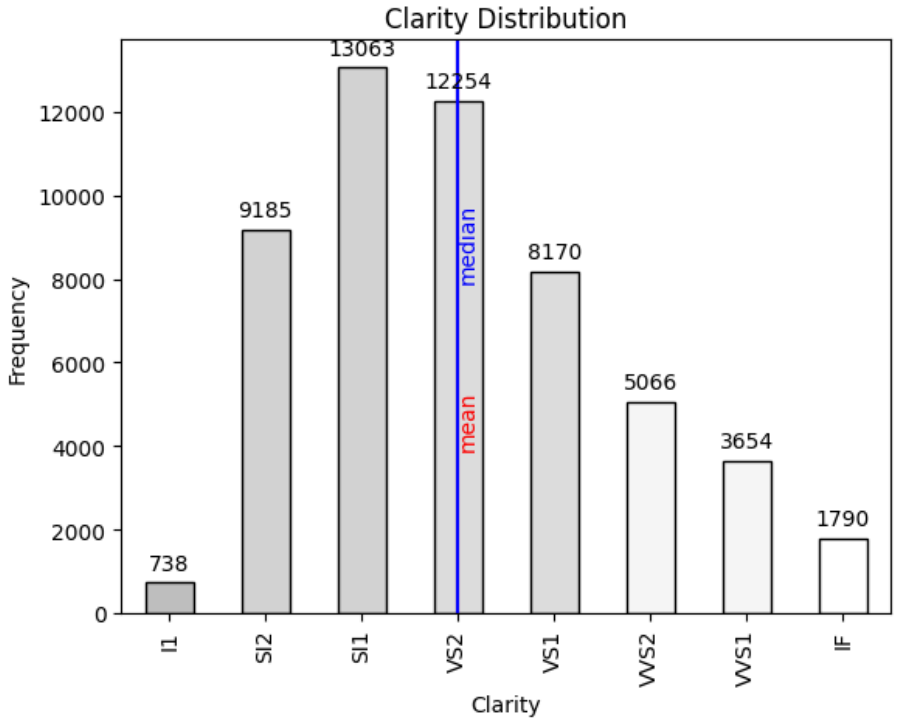
3. Cut

The cut of the diamond ranges from Fair (worst), Good, Very Good, Premium, to Ideal (best). Since this is a categorical variable, the usual statistics do not apply. As such, the bar graph only displays the modal cut ‘Ideal’, the average cut ‘Premium’, and median cut ‘Very Good’. The count values for the categories are also displayed using `.value_counts()`.



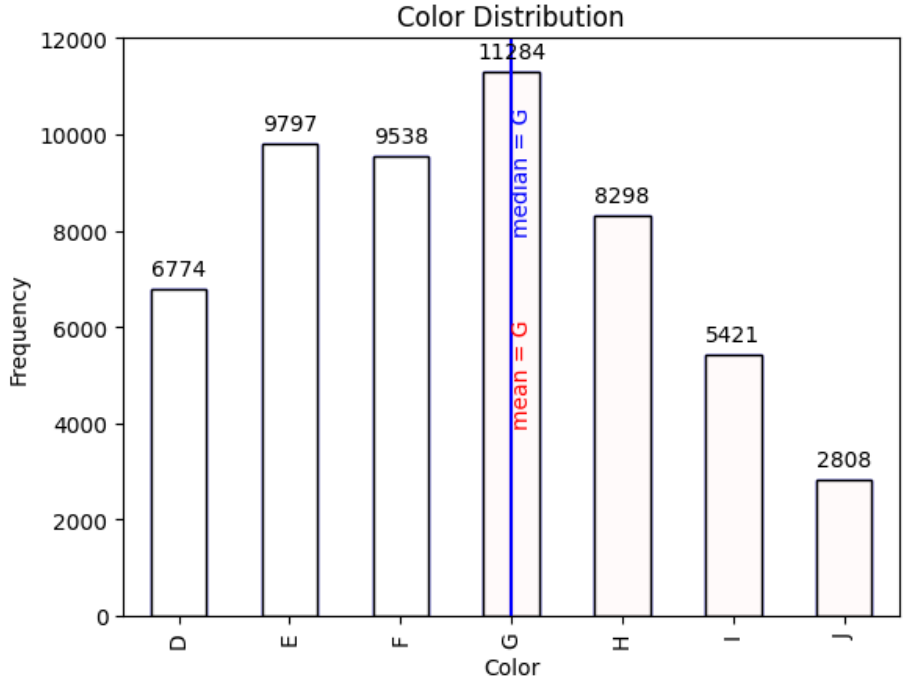
4. Clarity

The clarity of the diamonds vary from I1(very flawed), SI1 and SI2 (flawed), VS1 and VS2 (slightly flawed), VVS1 and VVS2 (few flaws), to IF (flawless). The median and average are VS2 while the modal clarity is SI2. The intensity of bar color indicates the degree of flaws.



5. Color

The color of a diamond varies from D, E, F (clear, white) to G, H, I, J (very slightly yellow). The bar chart below shows the distribution of colors of diamonds in the dataset with mode, median, mean being “G” color. The colors of the bars themselves reflect the subtle color differences between the categories.

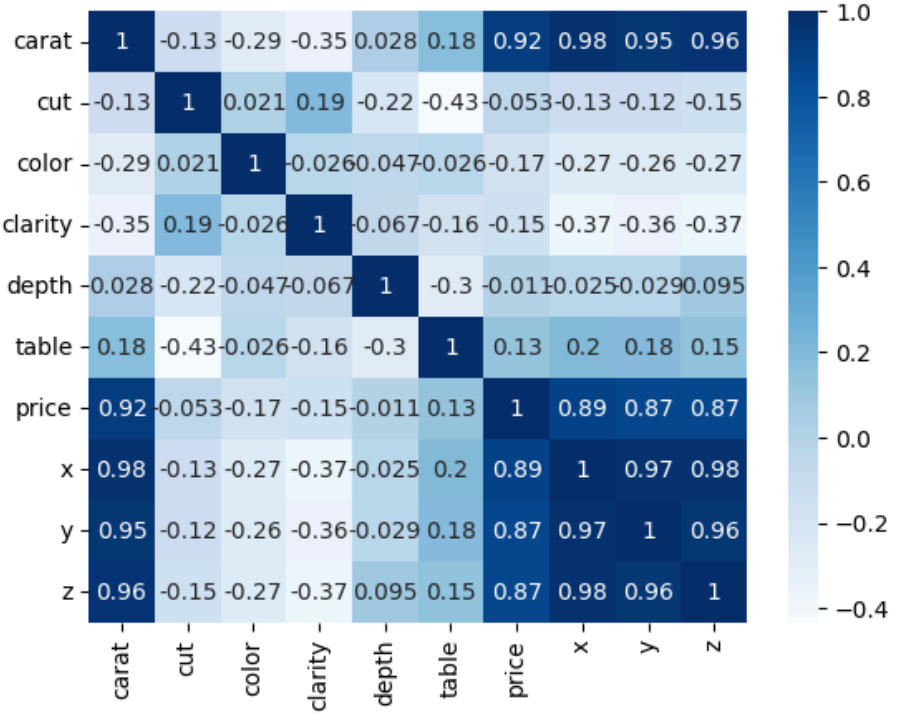


Correlations

1. Between all attributes

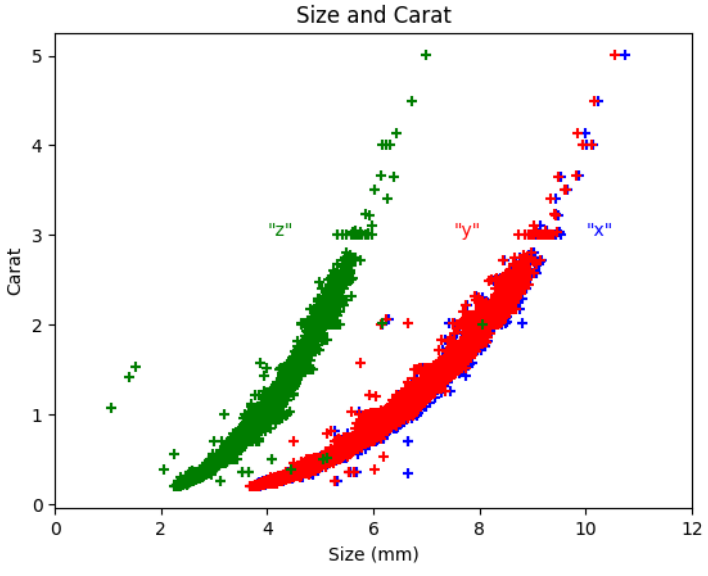
Since the categorical variables are now label encoded, a correlation chart can be made. The Pearson correlations were fetched using seaborn's `.corr()` and the chart below was made using `.heatmap()`. Here, we can gain some insights about the attribute relations:

- Carat and price are highly correlated as expected, with $r = 0.92$.
- The size of the diamond (determined by x, y, z dimensions) are highly correlated to carat, which is a measure of mass of the diamond, with r ranging from 0.87 to 0.89.
- Unexpectedly, there is little to no correlation between price and individual features cut, color, clarity, depth, and table since r is between -0.17 and 0.13.



2. Carat vs Size

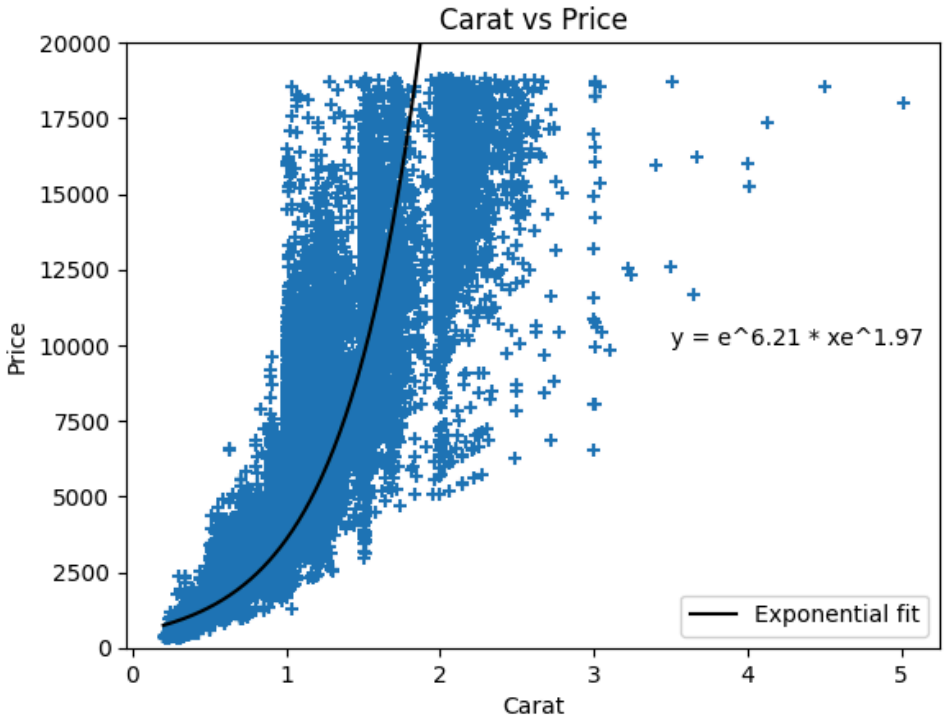
Upon plotting `carat` against `x`, `y`, `z` dimensions of the diamond, we can see that the size is indeed positively correlated to the carat of the diamond. `x` and `y` overlap as they are both width dimensions while `z` is a height dimension.



3. Price vs Carat

Upon plotting the scatter plot of carat vs price, it is visible that the correlation is non-linear. Thus I used numpy's `.log()` function to convert price to ln(`price`) and fit the best line using `p=np.polyfit()`. The function returned an array containing the constants to form the equation of this line, which I plotted separately as a line graph using an artificially finely spaced carat variable ranging from 0.2 to 5.01 with 500 points using `np.linspace()`.

The final equation of the best line is $y = e^{6.21} \times xe^{1.97}$.



Further Work

With this understanding of the spread of values and the correlations between attributes in the dataset, future work with this data can focus on using feature engineering to improve predictions for diamond prices based on their attributes. By considering the extent of correlation one can also optimize weightages of features in the prediction model. Using the insights from the attributes' distributions and their relationships to price, it will be possible to design and test, train and validate a machine learning model on this dataset.

References

[1] GIA 4Cs. "Why the 4Cs? | GIA 4Cs." *GIA 4Cs*, 23 Oct. 2024, 4cs.gia.edu/en-us/4cs-of-diamond-quality. Visited 4 Nov. 2024.

[2] "Diamonds." *Kaggle*, 25 May 2017, www.kaggle.com/datasets/shivam2503/diamonds. Visited 15 Oct. 2024.