Diamond Data Analysis

Python Data Analysis Project

Introduction :

A **diamond** is a precious gemstone formed from carbon atoms arranged in a unique crystalline structure known as a diamond cubic lattice. Its exceptional physical properties make it one of the most sought-after gemstones for jewellery and industrial applications.

Industry Scope :

The **diamond industry** encompasses a wide range of activities, including mining, cutting, polishing, trading, and retailing of diamonds. Its scope spans multiple sectors, from luxury goods to industrial applications. The diamond industry is vast and evolving, combining tradition with innovation.

Purpose of the Analysis :

The purpose of analysing diamond-related data is to extract meaningful insights that can inform decision-making, improve business strategies, and enhance understanding of the diamond. Understanding of Diamonds evaluated based on their **physical, aesthetic, and market-based features**, which help determine their quality and value. These features are often classified using the **4Cs** framework, Caret, Cut, Color and Clarity.

DataSet OverView :

The dataset comprises 1000 rows and 11 columns, each providing information about individual Diamond titles, such as their Caret, color of diamond, and Price. Below is an overview of the columns in the dataset:

- **Rows (Observations)**: 1000
- **Columns (Features)**: 11
- **Column Names**: The dataset initially had generic column names (`Unnamed: X`). After examining the content, these columns correspond to:
  - **S.N.**: Serial number (observation index).
  - **Carat**: Weight of the diamond.
  - **Cut**: Quality of the diamond's cut (e.g., Ideal, Premium).
  - **Color**: Grading of color, from D (colourless) to Z (yellowish).
  - **Clarity**: Clarity grading, based on inclusions and blemishes (e.g., SI2, VS1).
  - **Depth**: Height of the diamond (table to culet) as a percentage of its width.
  - **Table**: Width of the top flat surface as a percentage of the diamond's total width.
  - **Price**: Price of the diamond (likely in USD).
  - **X, Y, Z**: Dimensions of the diamond in mm (length, width, depth).

1. Import and Setup :

[22]:
```python
#Libraries for data manipulation
import pandas as pd
import numpy as np

#for visualization
import seaborn as sbn
import matplotlib.pyplot as plt
#import missingno as msno

#setup the frame visualize
sns.set(style="whitegrid")
plt.rcParams['figure.figsize']=(10,6)
```

2. Data Loading :

[36]:

```
#Loading data set
df=pd.read_csv('C:/Users/vinay/OneDrive/Documents/diamonddataset.csv')

#diplay some first and some rows to understand data arc
df.head()
```

[36]:

	S.N.	Caret	Cut	Color	Clarity	Depth	Table	Price	X	Y	Z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326.0	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326.0	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327.0	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334.0	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335.0	4.34	4.35	2.75

[38]:

```
df.tail()
```

[38]:

	S.N.	Caret	Cut	Color	Clarity	Depth	Table	Price	X	Y	Z
994	995	0.76	Premium	E	SI1	61.1	58.0	2897.0	5.91	5.85	3.59
995	996	0.54	Ideal	D	VVS2	61.4	52.0	2897.0	5.30	5.34	3.26
996	997	0.72	Ideal	E	SI1	62.5	55.0	2897.0	5.69	5.74	3.57
997	998	0.72	Good	F	VS1	59.4	61.0	2897.0	5.82	5.89	3.48
998	999	0.74	Premium	D	VS2	61.8	58.0	2897.0	5.81	5.77	3.58

3. Exploratory Data Analysis :

3.1 Basic Information of datasets:

[40]:

```
#display the data types of each column...
df.dtypes
```

[40]:

```
S.N.          int64
Caret       float64
Cut          object
Color        object
Clarity      object
Depth       float64
Table       float64
Price       float64
X           float64
Y           float64
Z           float64
dtype: object
```

[49]:

```
df.shape
```

[49]:

```
(999, 11)
```

[42]:

```
df.info()
```

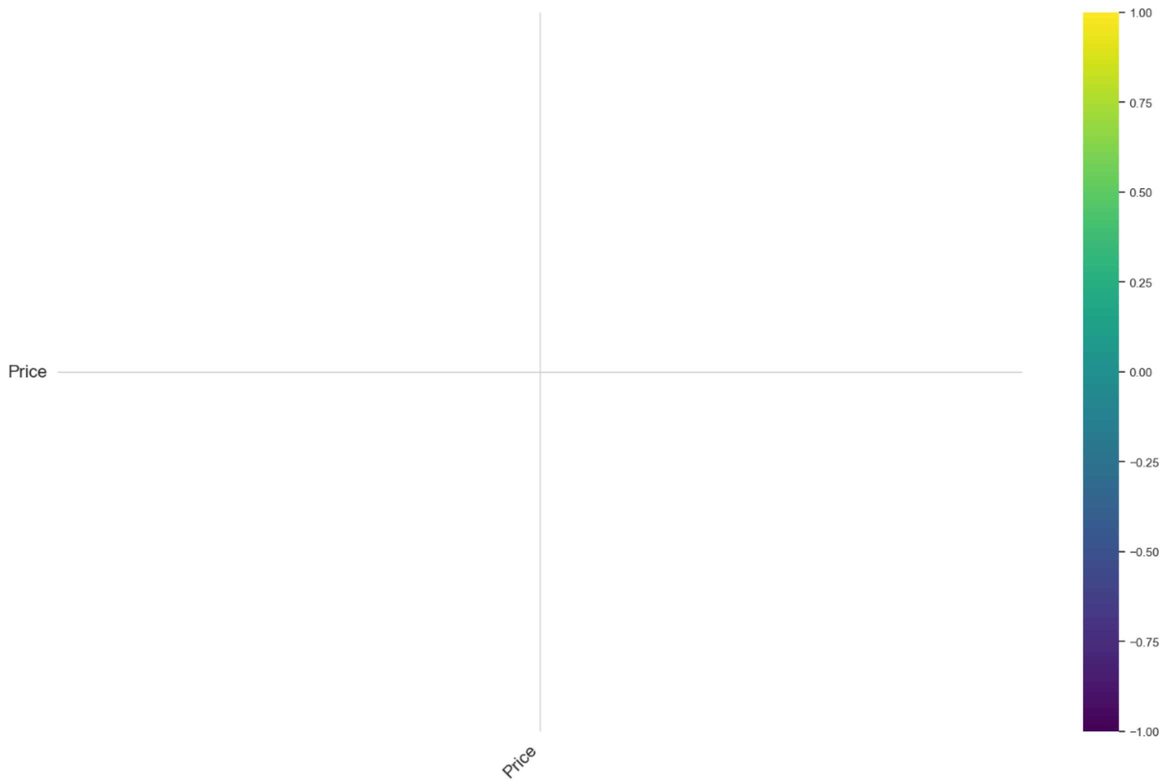
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   S.N.     999 non-null    int64  
 1   Caret    999 non-null    float64
 2   Cut      999 non-null    object 
 3   Color    999 non-null    object 
 4   Clarity  999 non-null    object 
 5   Depth    999 non-null    float64
 6   Table    999 non-null    float64
 7   Price    998 non-null    float64
 8   X        999 non-null    float64
 9   Y        999 non-null    float64
 10  Z        999 non-null    float64
dtypes: float64(7), int64(1), object(3)
memory usage: 86.0+ KB
```

3.2 The Missing Check values :

[29]:

```
#visualize missing data using heatmap
msno.heatmap(df, cmap='viridis')
```

[29]: <Axes: >



3.3 Duplicate Rows :

[31]:

```
#count and remove duplicate rows :
df.duplicated().sum()
df=df.drop_duplicates()
```

3.4 Building Synthetic Data Set :

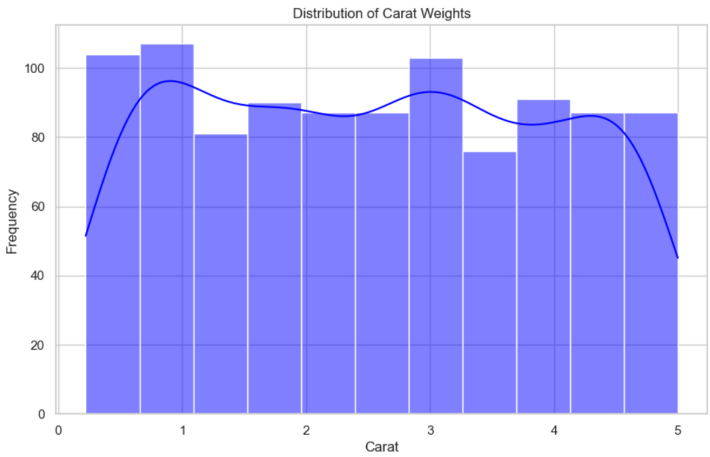
```
[14]: # Generating a synthetic dataset
np.random.seed(42)
data = {
    "carat": np.round(np.random.uniform(0.2, 5.0, 1000), 2),
    "cut": np.random.choice(["Fair", "Good", "Very Good", "Premium", "Ideal"], 1000),
    "color": np.random.choice(["D", "E", "F", "G", "H", "I", "J"], 1000),
    "clarity": np.random.choice(["IF", "VVS1", "VVS2", "VS1", "VS2", "SI1", "SI2"], 1000),
    "price": np.round(np.random.uniform(300, 20000, 1000), 2),
}
df = pd.DataFrame(data)

# Preview the dataset
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   carat    1000 non-null   float64
 1   cut      1000 non-null   object 
 2   color    1000 non-null   object 
 3   clarity  1000 non-null   object 
 4   price    1000 non-null   float64
dtypes: float64(2), object(3)
memory usage: 39.2+ KB
None
   carat        cut color clarity     price
0   2.00    Premium     J    VVS2   7607.54
1   4.76  Very Good     E     SI1  18068.22
2   3.71      Ideal     F    VVS1  16191.86
3   3.07       Fair     I      IF  19701.71
4   0.95      Ideal     D      IF  15158.69
```

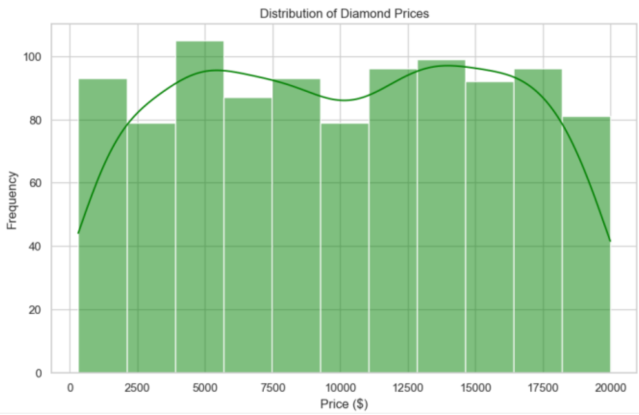
4. Analysis Queries :

- **Numeric :**

```
[15]: # Carat Distribution
sns.histplot(df['carat'], kde=True, color='blue')
plt.title('Distribution of Carat Weights')
plt.xlabel('Carat')
plt.ylabel('Frequency')
plt.show()
```

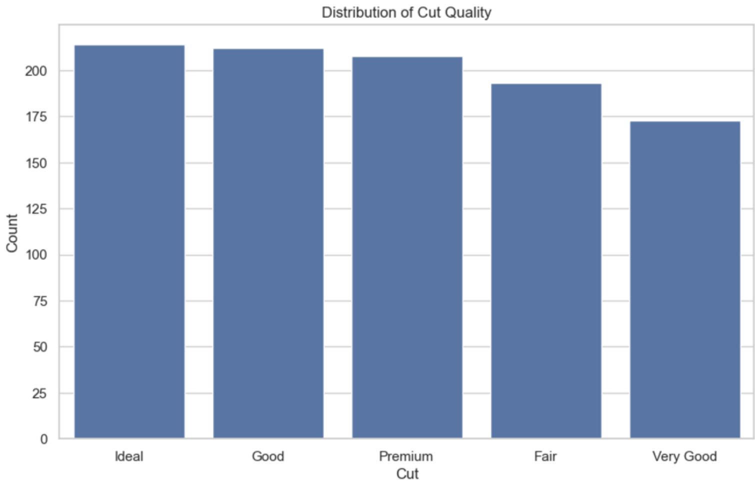



```
[16]: # Price Distribution
sns.histplot(df['price'], kde=True, color='green')
plt.title('Distribution of Diamond Prices')
plt.xlabel('Price ($)')
plt.ylabel('Frequency')
plt.show()
```

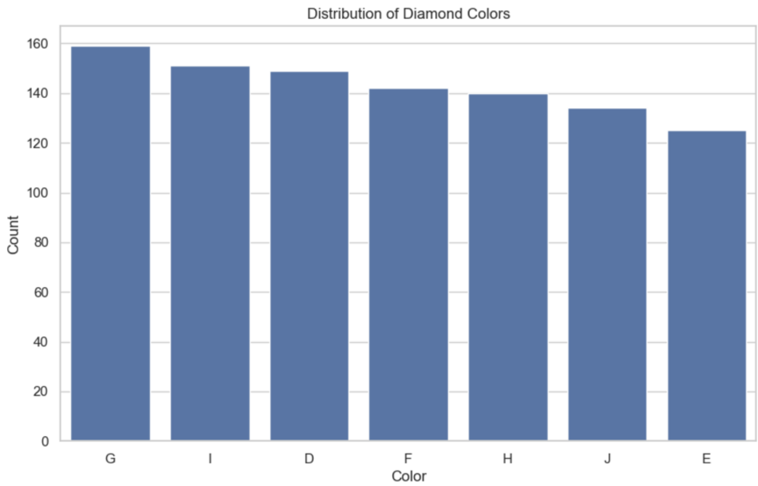


- **Categorical :**

```
[18]: #Cut Distribution
sns.countplot(x='cut', data=df, order=df['cut'].value_counts().index)
plt.title('Distribution of Cut Quality')
plt.xlabel('Cut')
plt.ylabel('Count')
plt.show()
```



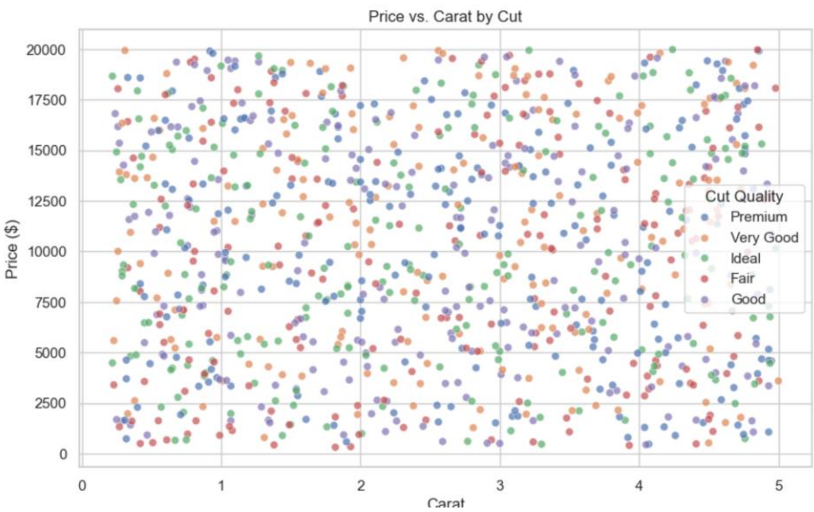

```
[19]: # Color Distribution
sns.countplot(x='color', data=df, order=df['color'].value_counts().index)
plt.title('Distribution of Diamond Colors')
plt.xlabel('Color')
plt.ylabel('Count')
plt.show()
```



4.1 Bivariate Analysis :

*. Price vs Carat :

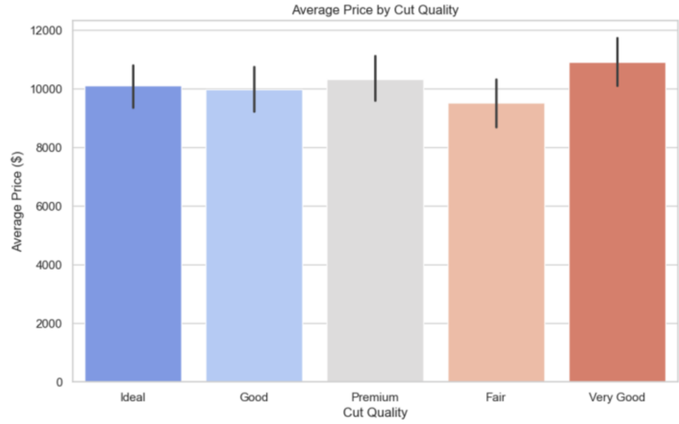
```
sns.scatterplot(x='carat', y='price', data=df, hue='cut', alpha=0.7)
plt.title('Price vs. Carat by Cut')
plt.xlabel('Carat')
plt.ylabel('Price ($)')
plt.legend(title='Cut Quality')
plt.show()
```



*. Average Price by Cut :

```
[21]: sns.barplot(x='cut', y='price', data=df, order=df['cut'].value_counts().index, palette='coolwarm')
plt.title('Average Price by Cut Quality')
plt.xlabel('Cut Quality')
plt.ylabel('Average Price ($)')
plt.show()
```

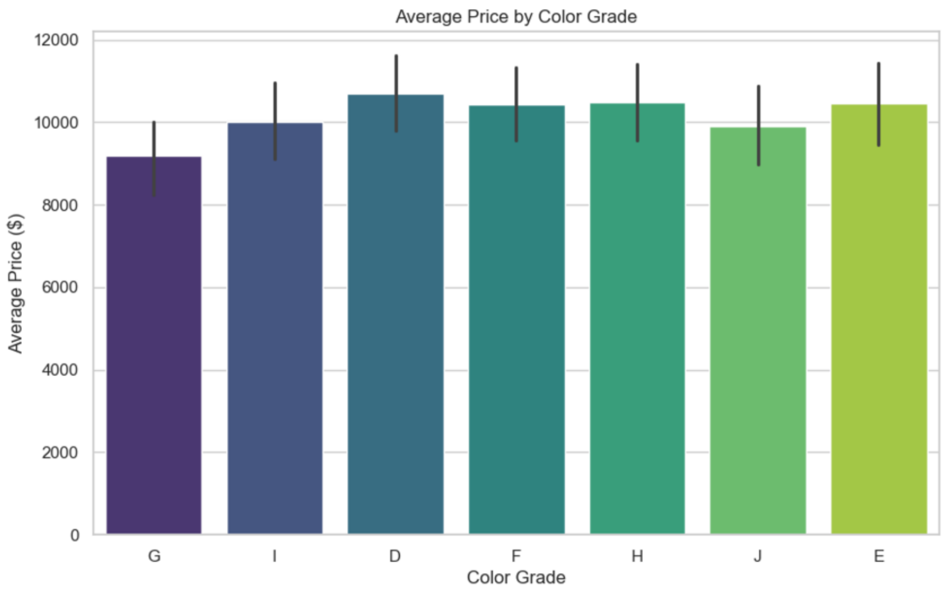
```
C:\Users\vinay\AppData\Local\Temp\ipykernel_8780\1464452416.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='cut', y='price', data=df, order=df['cut'].value_counts().index, palette='coolwarm')
```



*. Average Price by Color :

```
[22]: sns.barplot(x='color', y='price', data=df, order=df['color'].value_counts().index, palette='viridis')
plt.title('Average Price by Color Grade')
plt.xlabel('Color Grade')
plt.ylabel('Average Price ($)')
plt.show()
```

```
C:\Users\vinay\AppData\Local\Temp\ipykernel_8780\293574538.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='color', y='price', data=df, order=df['color'].value_counts().index, palette='viridis')
```



5. Insights :

*. Carat vs Price:

Larger carat diamonds tend to have higher prices.

*. Cut Quality:

Premium and Ideal cuts tend to have higher average prices.

*. Color Grades:

Diamonds with better color grades (D, E) are more expensive on average.

Summary

The diamond dataset highlights that carat, cut, color, and clarity are the key determinants of a diamond's price, with carat showing the strongest positive correlation. High-quality cuts like Ideal and Premium, as well as better color grades (D, E, F) and clarity grades (FL, IF, VVS1), command significantly higher prices. The dataset predominantly consists of mid-range diamonds, such as (SI1) and (VS2) in clarity and G or H in color, balancing quality and affordability. Outliers in carat and price represent rare, luxury-grade diamonds, underscoring niche market dynamics.

This dataset is well-suited for building predictive pricing models and understanding customer preferences.

Vinay Upadhyay | LinkedIn