Diamond price prediction

Linear Regression

Abstract:

Have you ever asked yourself, how are diamonds priced? Well, this project talks about the diamonds price prediction based on their cut, color, clarity & other attributes and it also covers the building a simple linear regression model using Python.

Design:

This project is one of the T5 Data Science Boot Camp requirements. Data provided by Kaggle. In this module, we will be laying the foundation for our analysis by processing and exploring a large amount of data on diamond datasets. This dataset has been made available thanks to Kaggle which is the home for many such datasets and competitions.

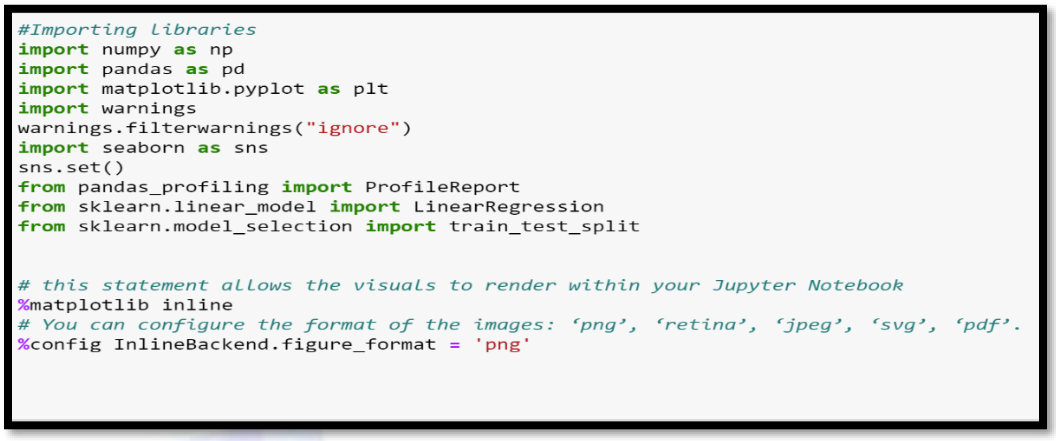
Understanding the dataset:

This dataset considered the classic Diamonds dataset which contains the prices and other attributes of almost 54,000 diamonds and this dataset is hosted on Kaggle. The dataset contains 53940 rows and 10 variables. Before jumping into building the model, let's have a look into the variables & their definitions.

Price: price in US dollars (\$326--\$18,823)
Carat: weight of the diamond (0.2--5.01)
Cut: quality of the cut (Fair, Good, Very Good, Premium, Ideal)
Color: diamond color, from J (worst) to D (best)
Clarity: a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))
X: length in mm (0--10.74)
Y: width in mm (0--58.9)
Z: depth in mm (0--31.8)
Depth: total depth percentage = z / mean (x, y) = 2 * z / (x + y) (43--79)
Table: width of top of diamond relative to widest point (43--95)

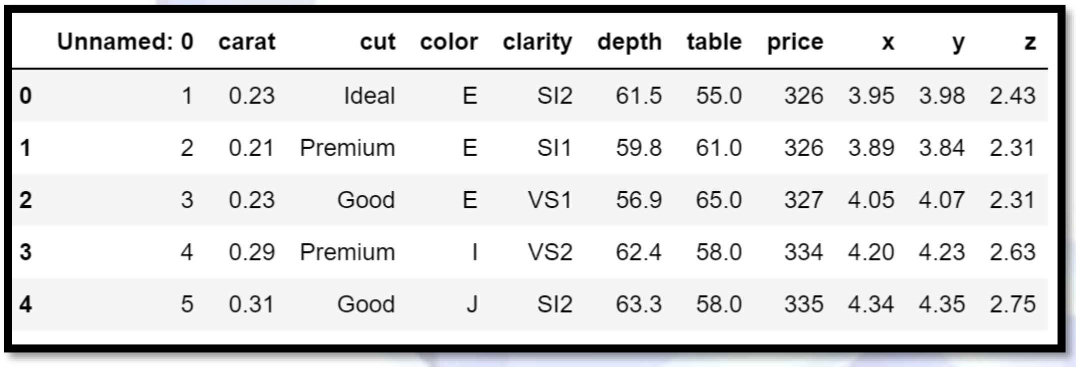
Algorithms:

1. Import Required Packages



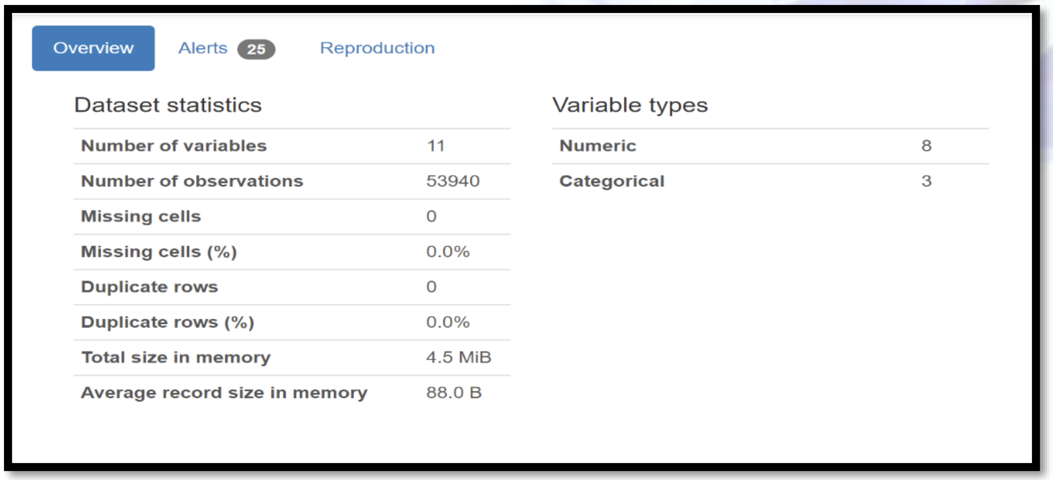
```
#Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
sns.set()
from pandas_profiling import ProfileReport
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split


# this statement allows the visuals to render within your Jupyter Notebook
%matplotlib inline
# You can configure the format of the images: 'png', 'retina', 'jpeg', 'svg', 'pdf'.
%config InlineBackend.figure_format = 'png'
```

2. Load the dataset



	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

3. Perform the exploratory data analysis (EDA)

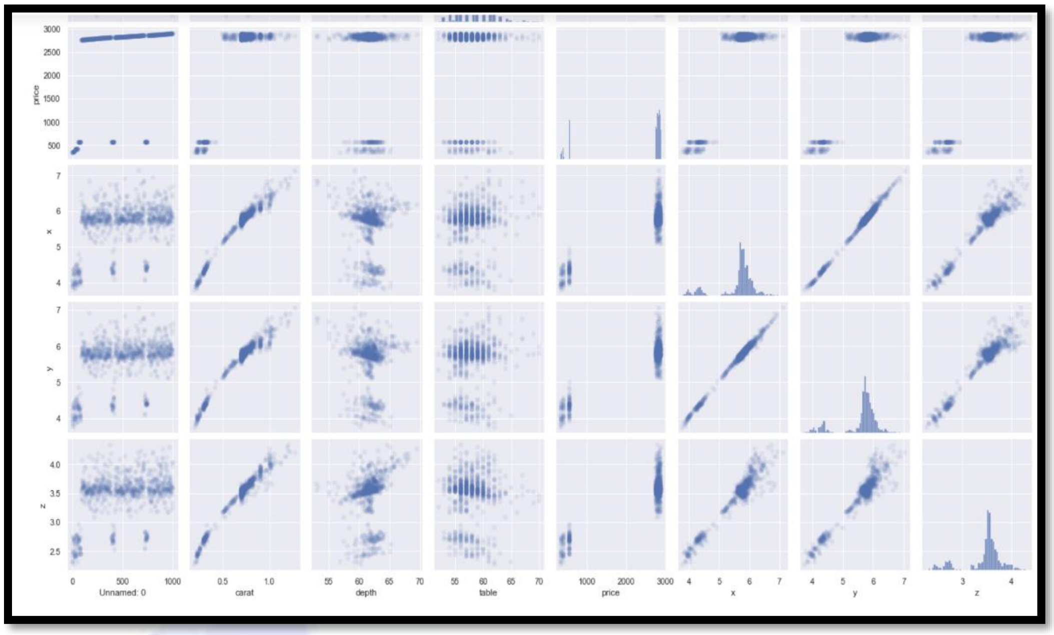


Dataset statistics

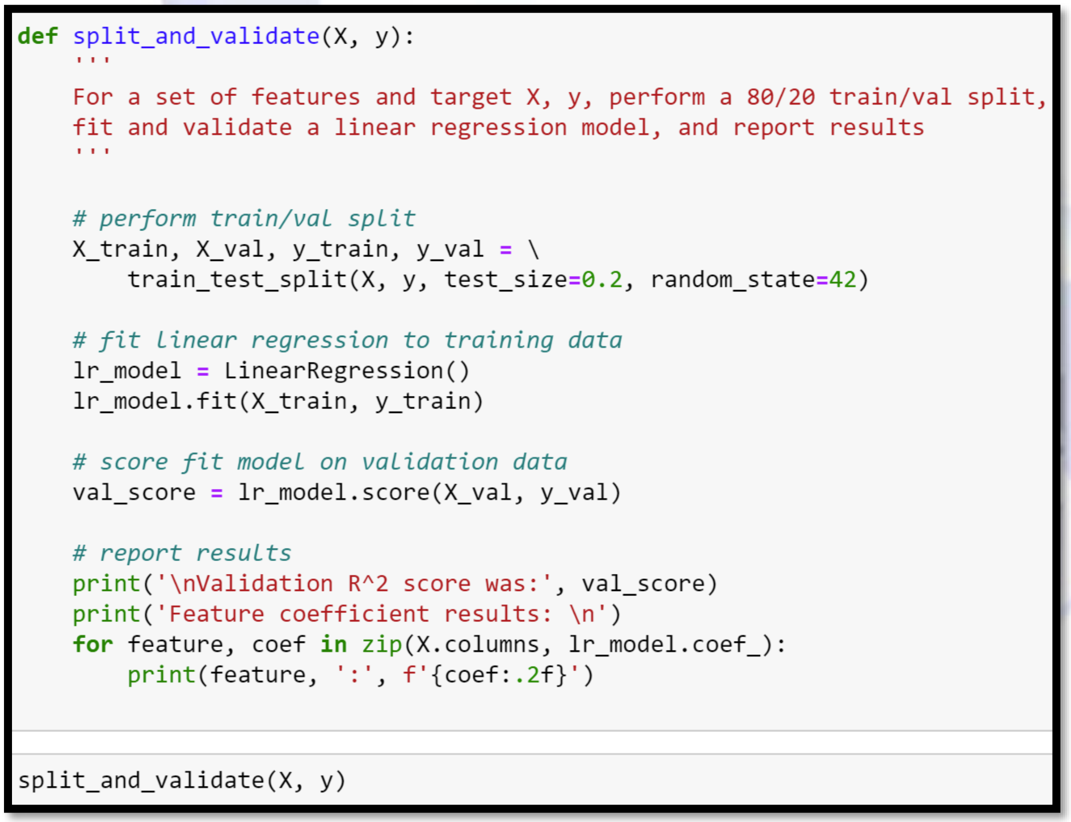
Number of variables	11
Number of observations	53940
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	4.5 MiB
Average record size in memory	88.0 B

Variable types

Numeric	8
Categorical	3



4. Prepare the dataset for training



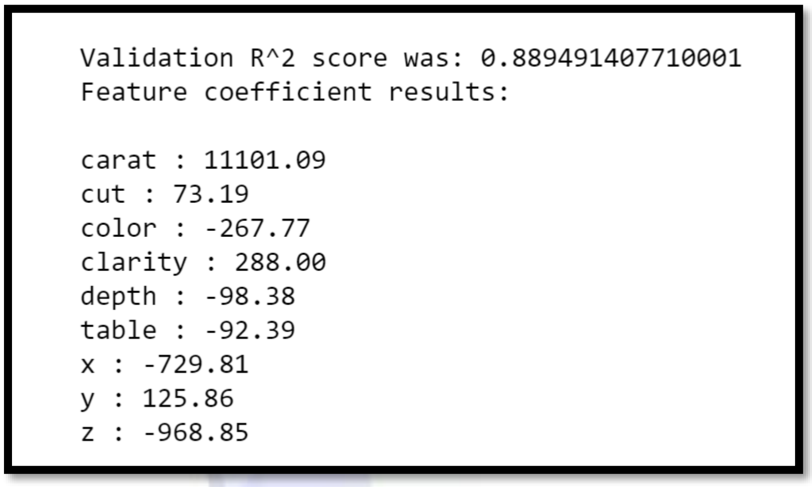
```
def split_and_validate(X, y):
    '''
    For a set of features and target X, y, perform a 80/20 train/val split,
    fit and validate a linear regression model, and report results
    '''

    # perform train/val split
    X_train, X_val, y_train, y_val = \
        train_test_split(X, y, test_size=0.2, random_state=42)

    # fit linear regression to training data
    lr_model = LinearRegression()
    lr_model.fit(X_train, y_train)

    # score fit model on validation data
    val_score = lr_model.score(X_val, y_val)

    # report results
    print('\nValidation R^2 score was:', val_score)
    print('Feature coefficient results: \n')
    for feature, coef in zip(X.columns, lr_model.coef_):
        print(feature, ':', f'{coef:.2f}')
```

```
split_and_validate(X, y)
```

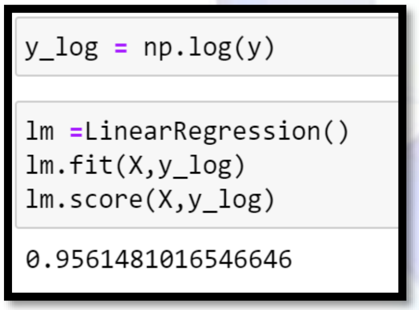
5. Create a linear regression model

6. Train the model to fit the data

7. Make predictions using the trained model



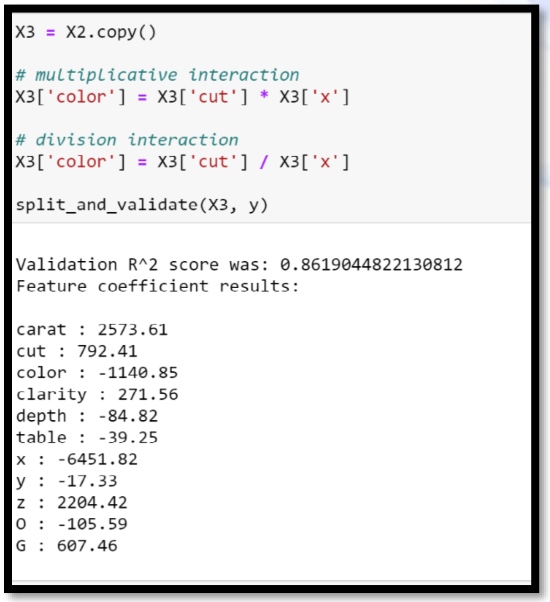
```
Validation R^2 score was: 0.889491407710001
Feature coefficient results:

carat : 11101.09
cut : 73.19
color : -267.77
clarity : 288.00
depth : -98.38
table : -92.39
x : -729.81
y : 125.86
z : -968.85
```



```
y_log = np.log(y)
```

```
lm =LinearRegression()
lm.fit(X,y_log)
lm.score(X,y_log)
```

```
0.9561481016546646
```



```
X3 = X2.copy()

# multiplicative interaction
X3['color'] = X3['cut'] * X3['x']

# division interaction
X3['color'] = X3['cut'] / X3['x']

split_and_validate(X3, y)
```

```
Validation R^2 score was: 0.8619044822130812
Feature coefficient results:

carat : 2573.61
cut : 792.41
color : -1140.85
clarity : 271.56
depth : -84.82
table : -39.25
x : -6451.82
y : -17.33
z : 2204.42
O : -105.59
G : 607.46
```

Tools:

•Pandas for data manipulation

• IQR for discover outliers

• Remove Duplicate or unnecessary data

• Matplotlib for plotting

• Seaborn for visualizations

• Sklearn Linear Regression library

Communication:

• The slides will be provided here, feel free to any pull requests besides details are provided at the readme of the project.