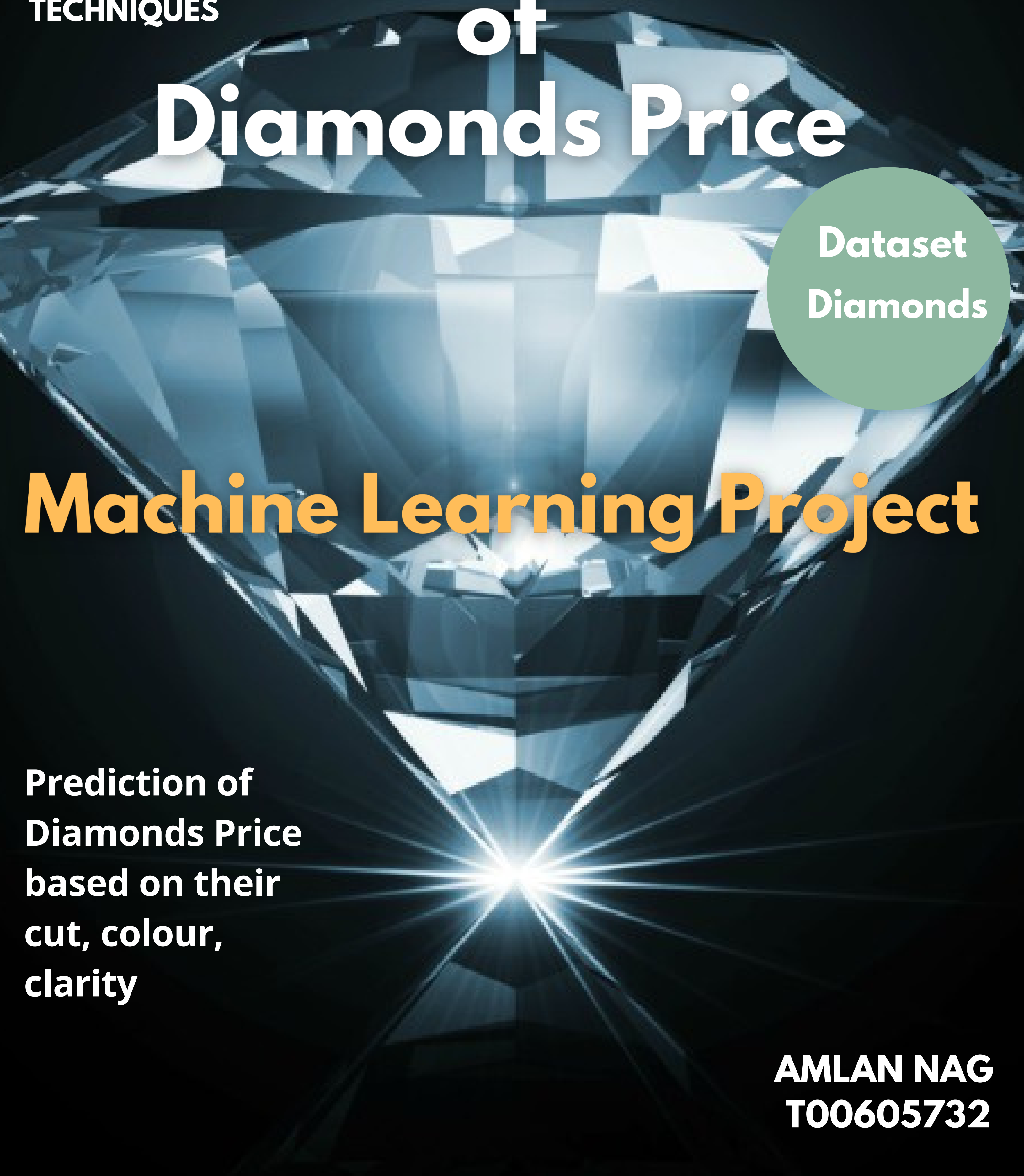
SUPERVISED MACHINE LEARNING TECHNIQUES

Prediction of Diamonds Price

APPPLIED ARTIFICIAL INTELLIGENCE

Dataset Diamonds



Machine Learning Project

Prediction of Diamonds Price based on their cut, colour, clarity

AMLAN NAG
T00605732

Contents

Introduction and Background

This project mainly focused on the price prediction of diamonds based on their cut, color, clarity and price. Diamond is the highest valuable and precious metal in terms of jewellery. In northern America, Diamond has a separate Executive based market. Diamonds are perhaps the most expensive among several products that consumers do not quantitatively or objectively value. With a strong emphasis on interpersonal relations, buying is far from fair. The jewellers will entice any man and woman by selling it as a must for the occasion and a status symbol and referring to this expensive and unaffordable piece as priceless.

On the other hand, a gemologist determines a diamond's real value after examining its different "features" and applying the relative valuation theory of "compare and price." As a precious metal, diamond having the most desirable value in the current world. From my initial research on the dataset - Diamond, I analysed and found appropriate levels which would help me to build an ML agent which could predict the future price based on statistics and features. From there, I researched and finalized that I would work with this dataset. I have dropped down all unnecessary data and train them to implement various linear regression models and classifiers.

For this particular model, a classifier would not be the best choice to determine the accuracy. I have implemented different types of regression models and generated accuracy results for each model, giving us a reasonable rate to predict the future diamond price rate. Understanding the background strategies of diamond price factors and future diamond values helps me determine the procedures I should take to train my agent using Machine learning techniques, which help me predict diamond price as per the requirements as diamond values depend on several features and its outreach market. Throughout the background research process, I have gathered all required techniques that would help to implement the machine learning techniques. I also analyzed the dataset and implemented all regression models to determine the best accuracy and best regression value to predict the diamond price.

Project Details

I chose the dataset of “Diamonds” As described in the introduction, I analyzed the dataset initially. I found that there are levels in the dataset: “ Index #, carat, cut, color, clarity, depth, table, price, x, y, z.” I looked at the Diamonds dataset, which includes approximately 54,000 diamond prices and other attributes like carat, cut. There are 53940 rows and ten variables in this dataset. From this data set, I can train the agent in Machine Learning’s environment to observe the price as per quality and then it could predict.

Here all attributes are followed by :

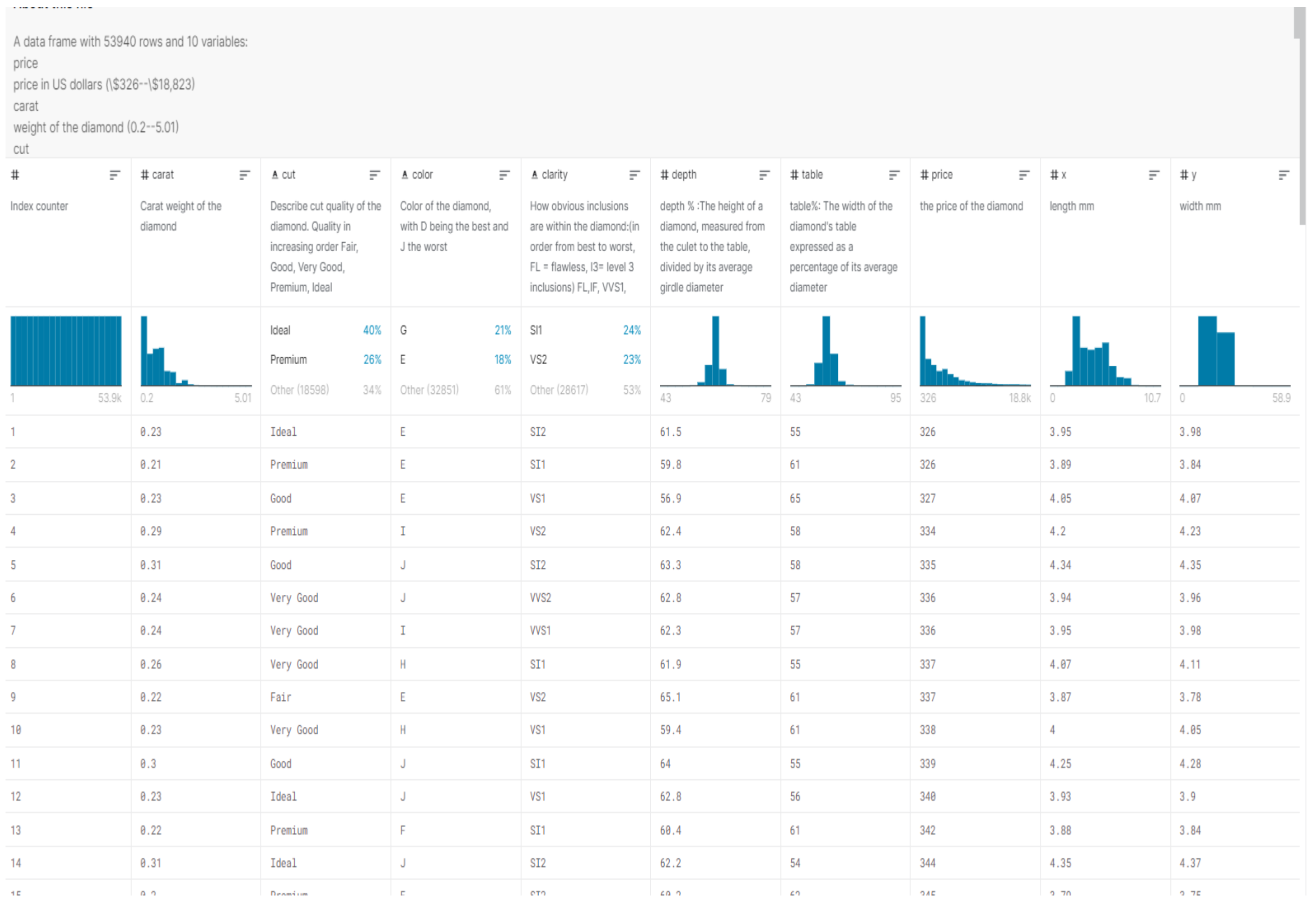
1. Price is in US dollars and Carat weight of the diamond.
2. Cut quality: Fair, Good, Very Good, Premium, Ideal and Color of the diamond, with D being the best and J the worst.
3. Clarity : (In order from best to worst, FL = flawless, I3= level 3 inclusions) IF, VVS1 and Depth %: The height of a diamond, measured from the culet to the table, divided by its average girdle diameter.
4. table%: The width of the diamond's table is expressed as a percentage of its average diameter and x length in mm and y width in mm.

Initially, I started sorting various rows per level and then determined that the index level would not require the project. Because the database has an automated number-wise level index as per data values. So, I dropped the column Unnamed=0; then it’s representing all data with numbered values. Then, I have called functions and methods to display data values and attributes. Then, I have described the data and then generated graphs and plots. Then implement the function data.isnull().sum() returns the number of missing values in the data set. Then after encoding the column values, it seems that for this project, there is no need for the column -depth, table, x,y,z as the prediction will focus on price carat, caret value, and allotment frequency would be enough to implement machine learning. So, I have dropped column depth, table, x,y and z. After that, I changed all data attributes to float.   Then based on that, implement virtualization graphical views of allotment of Diamond price.

This machine learning project that will help predict the diamond value would implement regression and classification. Accuracy and R^2 regression score function that coefficient of determination will help particular importance. Here in this project mainly used Linear Regression Algorithm, the regression algorithm, I have implemented the determinesLinearRegression model, DecisionTreeRegressor, Lasso, RandomForestRegressor, here RandomForestRegressor have been performed most accurately.

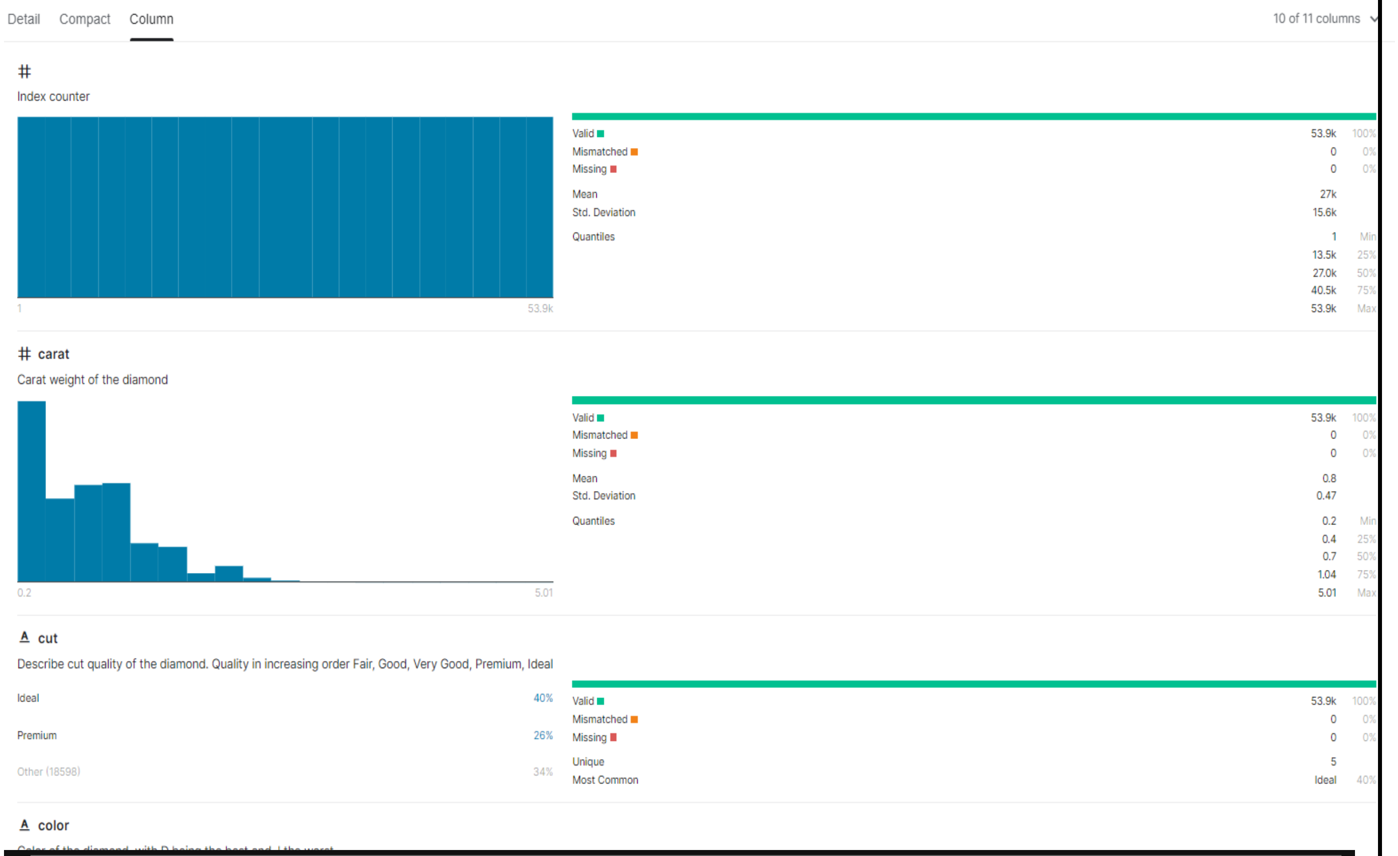
Splitting arrays or matrices into random train and test subsets, I have implemented the data arrays, shuffle values, and listed return values. It would then transform to standard scaler to x_train and x_test and fit the transform to classifier and regression model.

To suit the results, train the model. Linear regression's fundamental aim is to minimize the cost function. Before developing the model, it is a start to transform the categorical data to numerical data. There are two approaches to do this. 1. Integer Encoding or Mark Encoder 2. It is encoding in a single pass. For preparation, we need to translate the data from the Pandas data frame into PyTorch tensors. The first step is to transform it to NumPy arrays. For training and validation, we will need to build PyTorch datasets and data loaders. The first step is to make a TensorDataset by translating the input and target arrays to tensors with the torch.tensor function. And now, we'll divide the dataset into two parts: training and validation. The training set will be used to train the model, while the validation set will be used to test the model and tune the hyperparameters for improved generalization of the trained model. In the end, the user will input their acceptations like value for carat, cut, clarity and color; then, the machine will predict the price.



A data frame with 53940 rows and 10 variables:
price
price in US dollars (\$326--\$18,823)
carat
weight of the diamond (0.2--5.01)
cut

#	# carat	cut	color	clarity	# depth	# table	# price	# x	# y
Index counter	Carat weight of the diamond	Describe cut quality of the diamond. Quality in increasing order Fair, Good, Very Good, Premium, Ideal	Color of the diamond, with D being the best and J the worst	How obvious inclusions are within the diamond:(in order from best to worst, FL = flawless, I3= level 3 inclusions) FL,IF, VVS1,	depth % :The height of a diamond, measured from the culet to the table, divided by its average girdle diameter	table%: The width of the diamond's table expressed as a percentage of its average diameter	the price of the diamond	length mm	width mm
1 – 53.9k	0.2 – 5.01	Ideal 40% Premium 26% Other (18598) 34%	G 21% E 18% Other (32851) 61%	SI1 24% VS2 23% Other (28617) 53%	43 – 79	43 – 95	326 – 18.8k	0 – 10.7	0 – 58.9
1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98
2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84
3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07
4	0.29	Premium	I	VS2	62.4	58	334	4.2	4.23
5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35
6	0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96
7	0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98
8	0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11
9	0.22	Fair	E	VS2	65.1	61	337	3.87	3.78
10	0.23	Very Good	H	VS1	59.4	61	338	4	4.05
11	0.3	Good	J	SI1	64	55	339	4.25	4.28
12	0.23	Ideal	J	VS1	62.8	56	340	3.93	3.9
13	0.22	Premium	F	SI1	60.4	61	342	3.88	3.84
14	0.31	Ideal	J	SI2	62.2	54	344	4.35	4.37



Detail Compact Column 10 of 11 columns

#
Index counter

Valid 🟩	53.9k	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	27k	
Std. Deviation	15.6k	
Quantiles	1	Min
	13.5k	25%
	27.0k	50%
	40.5k	75%
	53.9k	Max

carat
Carat weight of the diamond

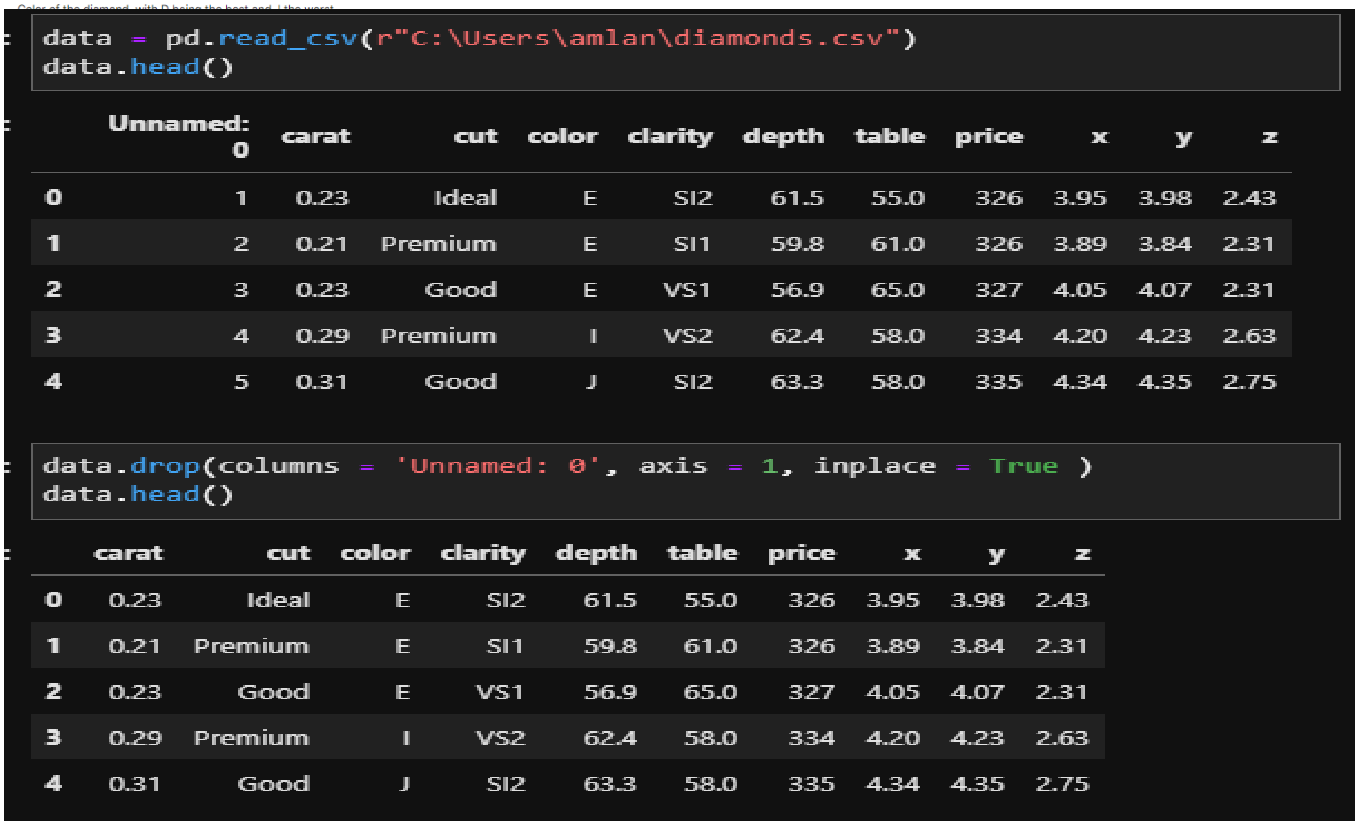
Valid 🟩	53.9k	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	0.8	
Std. Deviation	0.47	
Quantiles	0.2	Min
	0.4	25%
	0.7	50%
	1.04	75%
	5.01	Max

A cut
Describe cut quality of the diamond. Quality in increasing order Fair, Good, Very Good, Premium, Ideal

Ideal	40%
Premium	26%
Other (18598)	34%

Valid 🟩	53.9k	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	5	
Most Common	Ideal	40%

A color
Color of the diamond, with D being the best and J the worst



```
data = pd.read_csv(r"C:\Users\amlan\diamonds.csv")
data.head()
```

```
   Unnamed:  carat       cut  color  clarity  depth  table  price     x     y     z
          0
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
```

```
data.drop(columns = 'Unnamed: 0', axis = 1, inplace = True )
data.head()
```

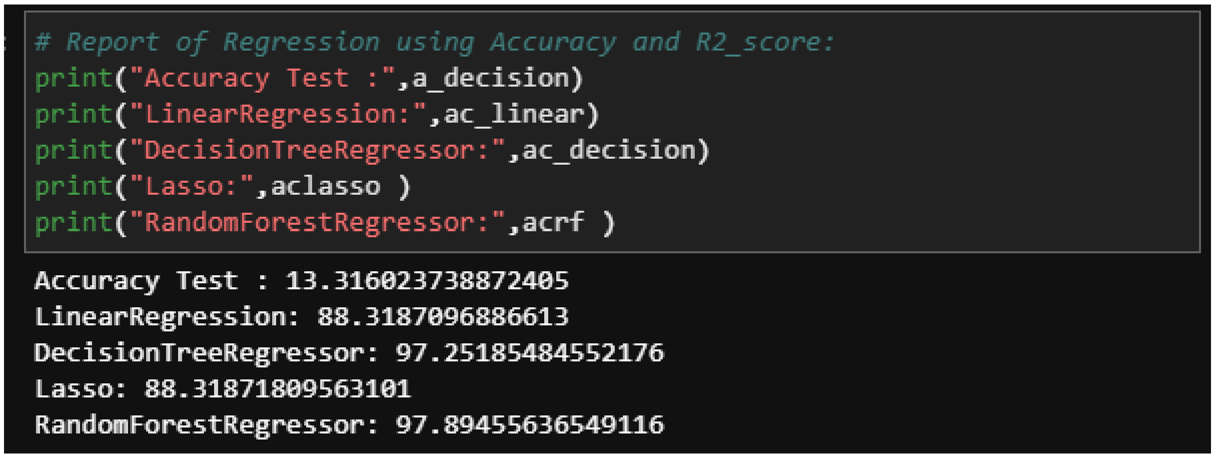
```
   carat       cut  color  clarity  depth  table  price     x     y     z
0   0.23     Ideal      E      SI2   61.5   55.0    326  3.95  3.98  2.43
1   0.21   Premium      E      SI1   59.8   61.0    326  3.89  3.84  2.31
2   0.23      Good      E      VS1   56.9   65.0    327  4.05  4.07  2.31
3   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
```

Method(s)

I have mainly used several regression algorithms for prediction and classification accuracy in this project, implemented decision tree classifiers. Regression techniques are classified as supervised ML. They help predict or explain a specific numerical value based on a compilation of preliminary data, such as forecasting a property’s price based on historical pricing data for comparable properties. Linear regression is the most fundamental technique in which we use the mathematical equation of the line (y = m * x + b) to model a data set. For several data pairs (x, y), we train a linear regression model by measuring the direction and slope of a line that minimizes the cumulative distance between all data points and the bar. In other words, we compute the slope and y-intercept of a line that best approximates the data observations. On the other hand, The most basic classification algorithm is logistic regression under that discussion tree classifier accuracy test, which sounds like a regression process but isn't. Based on one or more variables, accuracy measured the chance of an occurrence happening.

Here mainly after preparing the dataset, randomized values and transform them to standard scaler fit. I decided to start the prediction experiment with a classifier through my project based on Regression. Still, due to check accuracy by the classifier, I implemented the decision tree and assigned the values of X-train and x_test to the classifier and give it a name and a prediction method then implement accuracy test; the reason behind the uses of the techniques is to identify the ratio of prediction for classifier but in this case accuracy of result won't able to satisfy our prediction. Then I jumped to the regression part, and rather than taking precision or accuracy or r2_score; I implemented various regression factors that are most popular to predict metal price like diamonds.

The ML techniques that in several research came out are that diamond price can fluctuate and change as per the national geological demand. For that, depending on one regression model could impact the judgement of values. So, it is required to find out the most efficient and accurate regression model from time to time. In background research and other developer projects, they mainly focused on Linear Regression, which could be workable in predicting the partial value. So, I decided to implement several regression models top identify the best value by taking r2_score. Because it is a constant model that always predicts the expected value of y, disregarding the input features, it would get an R^2 score of 0.0. Also, compactable to require regression models like Lasso, Random forest and decision trees. Due to analytical performance, I have selected a random forest, a meta estimator that uses averaging to improve predictive precision and control over-fitting by fitting several classifying decision trees on different sub-samples. So, the background understanding, and implementation of reflection work out perfectly for the agent; the agent would implement the most accurate model to predict the price.



```
# Report of Regression using Accuracy and R2_score:
print("Accuracy Test :",a_decision)
print("LinearRegression:",ac_linear)
print("DecisionTreeRegressor:",ac_decision)
print("Lasso:",aclasso )
print("RandomForestRegressor:",acrf )
```

```
Accuracy Test : 13.316023738872405
LinearRegression: 88.3187096886613
DecisionTreeRegressor: 97.25185484552176
Lasso: 88.31871809563101
RandomForestRegressor: 97.89455636549116
```

Experiments/Simulations

<u>Steps have been followed to implement the project:</u>

1. From the brainstorming of the project and target prediction and to focus on the virtualization by graphs and the classifier or regression, I first imported all required

packages and required tools packages. In the import command packages like Seaborn, Tensordataset, torchvision,matplotlib, pandas and NumPy for different functionality.

2. Then from the local drive, I called the dataset CSV file of diamonds. Then using head call displayed few values. As the dataset will have auto index numbering, I do not need unnamed index values as it will waste memory space.
3. Then, I dropped the unnamed level index and then shaped sorted data and sorted it out like carat, depth, table price and other levels.
4. As for generating graphs and prepare the data set to analyze by the graphs and values, I followed the research of "Sindiri (2020)," data scientist, in the article "Diamond price prediction based on their cut, colour, clarity, price with PyTorch," he described and analyzed the factors behind the diamonds price and maintain prediction. Also, I followed the steps and commands to generate graphs**.** Data Visualization, data sorting, cleaning, training, and validation part I used my target techniques to implement the project.
5. Then declare as null the sum of each level and represent the data type. Then implement the function data.isnull().sum() returns the number of missing values in the data set. The representation of data info is then called and then re-initiate x,y,z to its null value and temp to its total sum average value and represent them, which ends up data cleaning.
6. After that, data shaping to axis value before generating the graph, now the data set is ready to represent the graph.
7. Now import matplotlib function, and in graph size, I set 20,15 ratios to represent all levels values differently. After that, generate an analyzed graph to understand the valuing factor. Then implement a heatmap and represent cool, warm standard to a compare view.
8. Then comparing set values as in the bar and then analyzing the bar views in catplot.
9. After that, I encoded the dataset with dummies and then described all set code values.

10. After analyzing the dataset then for my target project and training dataset, I do not need levels valued like depth, table,x,y,z, so I dropped all these.
11. Then I reviewed the dataset, and now it’s low storage and low memory management dataset. Then implement and change the dataset to float.
12. After that, I allocated the Diamond weight as per carat and second graph, which indicate the Diamond price. Here, I used xlabel as carat value and ylabel as frequency and then standard subplot 121 and 122 to implement the two different graph sections.
13. After that, I viewed the data set and then implemented a label encoder and transformed data fit of types of diamonds fit them in the ranking of number values. Also, implement cut label and color values to numbers ranking 1 to 8.
14. After new levelling levels with numerical values, then for optimization, I will drop previous levels data where it was valued with terms like color name or code now all set to a numerical value. Also, the price moved to float value, so I am drooping all previous level data.
15. Before generating graphs, initializing the value to null and recall the sum after this step, I followed down and headed up all values for the price. And all sorts of data preparation is complete.
16. The training dataset was then started, implemented the train test split and randomized values and defined train size value and declared a random state. After identifying the train value, test value and data values indicate the total number of types values.
17. After that head and tail are represented from the dataset. It indicates a data index from 0 to 53939; all data is levelled and classified in a split method.
18. After that, I scaled the values to standard scaler values; then, as planned, I started with a decision tree classifier where I called the function and made a prediction level. Then I implemented an accuracy level. Centring and scaling are performed separately on each

element by calculating the necessary statistics on the training set samples. The mean and standard deviation are then saved for later use with transform on other results.

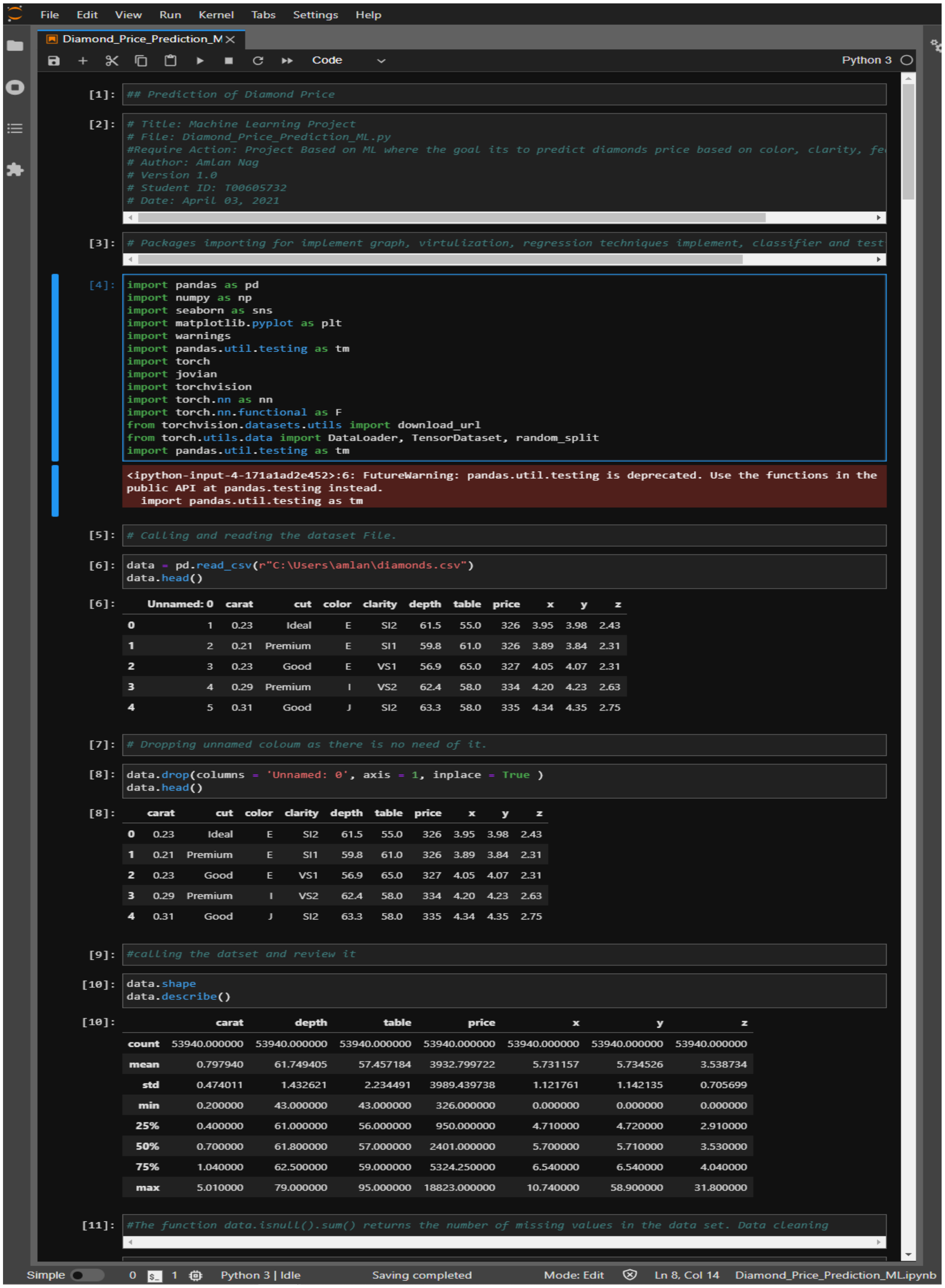
19. It's clear from the accuracy prediction value that this model would support more by a regression-based model. From there, I implemented LinearRegression, DecisionTreeRegressor, Lasso and RandomForestRegressor with the testing method of r2_score.
20. Then I take out all function prints, making it easier to choose the most accurate value model to complete the prediction of the diamond.
21. After that, I defined a function prediction and took the input of carat, cut, clarity, color and then initiated the function's price. In this case, the most accurate function is the random Forest Function. And The implementation of Machine learning reflects successfully and represents the value in USD for your preferred diamond.
22. We will then test the 2nd proficient score regression model to justify our result or do my accuracy test.

Detailed instructions/steps to run the program :

1. Download "Diamond_Price_Prediction_ML.ipynb" Or "Diamond_Price_Prediction_ML.ipynb" file from moodle assignment submission .
2. Download the "Diamonds.csv" file from moodle assignment submission and keep both files in the same directory.
3. Then It might require installing a few packages to run this project. E.g. Like torchvision , seaborn or jovian. Please use the command line prompt or PowerShell to install them using simple commands like- pip install torchvision or pip install 'require packages.'
4. Then Change the root directory of the CSV file. Go to the file where you download; please click properties, copy the file's address, and change the program up to amlan\.

5. Then you are ready to go; please start pressing Shift+ Enter for each line and continue going to the end. It might take time graphs to view, but it will. If any packages error came, please install that package. Clear all output and restart. Otherwise, there will be memory issues.
6. When you reach the def prediction function and click Shift+Enter in the jupyter lab (or Enter for python), the system will ask for a few inputs like carat, cut, clarity, color. To get a realistic price of a diamond, Provide carat input below 1.00 and others in the 1.00-5.00 range.
7. In the validation step, summary, you will see the best-fit algorithm for use that is actually used for the first prediction and the second prediction; I have used the decision tree algorithm.
8. I hope by now, you have received the value price for your desirable diamond price.

Screenshots:



```
[1]: ## Prediction of Diamond Price
```

```
[2]: # Title: Machine Learning Project
     # File: Diamond_Price_Prediction_ML.py
     #Require Action: Project Based on ML where the goal its to predict diamonds price based on color, clarity, fe
     # Author: Amlan Nag
     # Version 1.0
     # Student ID: T00605732
     # Date: April 03, 2021
```

```
[3]: # Packages importing for implement graph, virtulization, regression techniques implement, classifier and test
```

```
[4]: import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     import warnings
     import pandas.util.testing as tm
     import torch
     import jovian
     import torchvision
     import torch.nn as nn
     import torch.nn.functional as F
     from torchvision.datasets.utils import download_url
     from torch.utils.data import DataLoader, TensorDataset, random_split
     import pandas.util.testing as tm
```

```
<ipython-input-4-171a1ad2e452>:6: FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

```
[5]: # Calling and reading the dataset File.
```

```
[6]: data = pd.read_csv(r"C:\Users\amlan\diamonds.csv")
     data.head()
```

[6]:

	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

```
[7]: # Dropping unnamed coloum as there is no need of it.
```

```
[8]: data.drop(columns = 'Unnamed: 0', axis = 1, inplace = True )
     data.head()
```

[8]:

	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

```
[9]: #calling the datset and review it
```

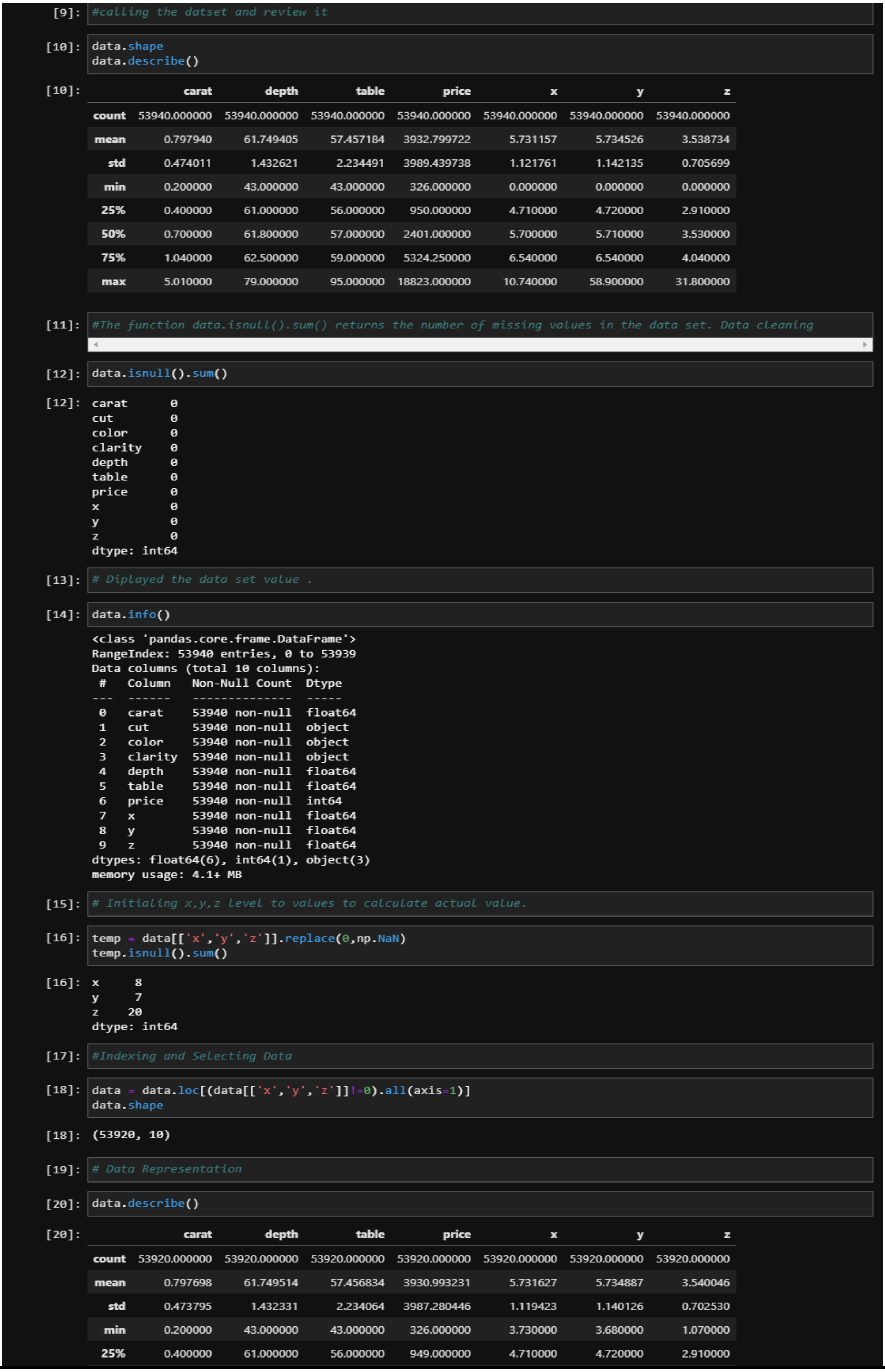
```
[10]: data.shape
      data.describe()
```

[10]:

	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

```
[11]: #The function data.isnull().sum() returns the number of missing values in the data set. Data cleaning
```

```
Simple   0   1   Python 3 | Idle   Saving completed   Mode: Edit   Ln 8, Col 14   Diamond_Price_Prediction_ML.ipynb
```



[9]:
```
#calling the datset and review it
```

[10]:
```
data.shape
data.describe()
```

[10]:

	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

[11]:
```
#The function data.isnull().sum() returns the number of missing values in the data set. Data cleaning
```

[12]:
```
data.isnull().sum()
```

[12]:
```
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
price      0
x          0
y          0
z          0
dtype: int64
```

[13]:
```
# Diplayed the data set value .
```

[14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   carat    53940 non-null  float64
 1   cut      53940 non-null  object
 2   color    53940 non-null  object
 3   clarity  53940 non-null  object
 4   depth    53940 non-null  float64
 5   table    53940 non-null  float64
 6   price    53940 non-null  int64
 7   x        53940 non-null  float64
 8   y        53940 non-null  float64
 9   z        53940 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 4.1+ MB
```

[15]:
```
# Initialing x,y,z level to values to calculate actual value.
```

[16]:
```
temp = data[['x','y','z']].replace(0,np.NaN)
temp.isnull().sum()
```

[16]:
```
x     8
y     7
z    20
dtype: int64
```

[17]:
```
#Indexing and Selecting Data
```

[18]:
```
data = data.loc[(data[['x','y','z']]!=0).all(axis=1)]
data.shape
```

[18]:
```
(53920, 10)
```

[19]:
```
# Data Representation
```

[20]:
```
data.describe()
```

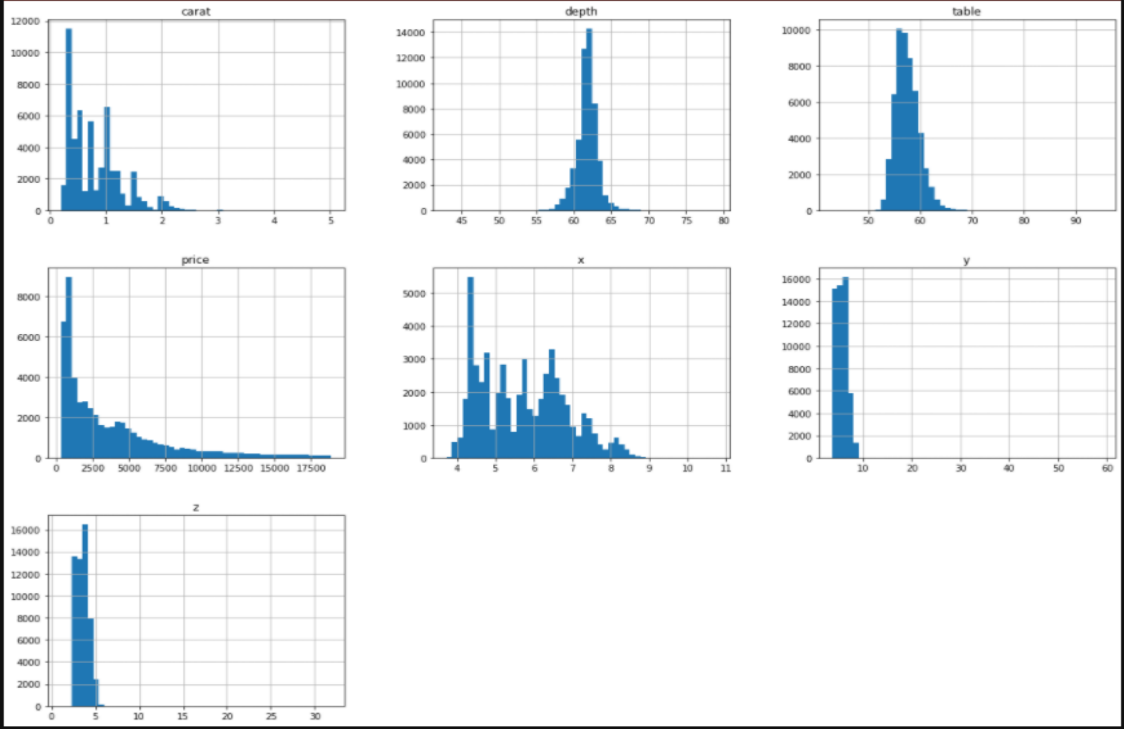
[20]:

	carat	depth	table	price	x	y	z
count	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000
mean	0.797698	61.749514	57.456834	3930.993231	5.731627	5.734887	3.540046
std	0.473795	1.432331	2.234064	3987.280446	1.119423	1.140126	0.702530
min	0.200000	43.000000	43.000000	326.000000	3.730000	3.680000	1.070000
25%	0.400000	61.000000	56.000000	949.000000	4.710000	4.720000	2.910000

```
[21]: # Function implemnet of graph ratio as per each level
```

```
[22]: import matplotlib.pyplot as plt
      data.hist(bins=50,figsize=(20,15))
      plt.show()
```

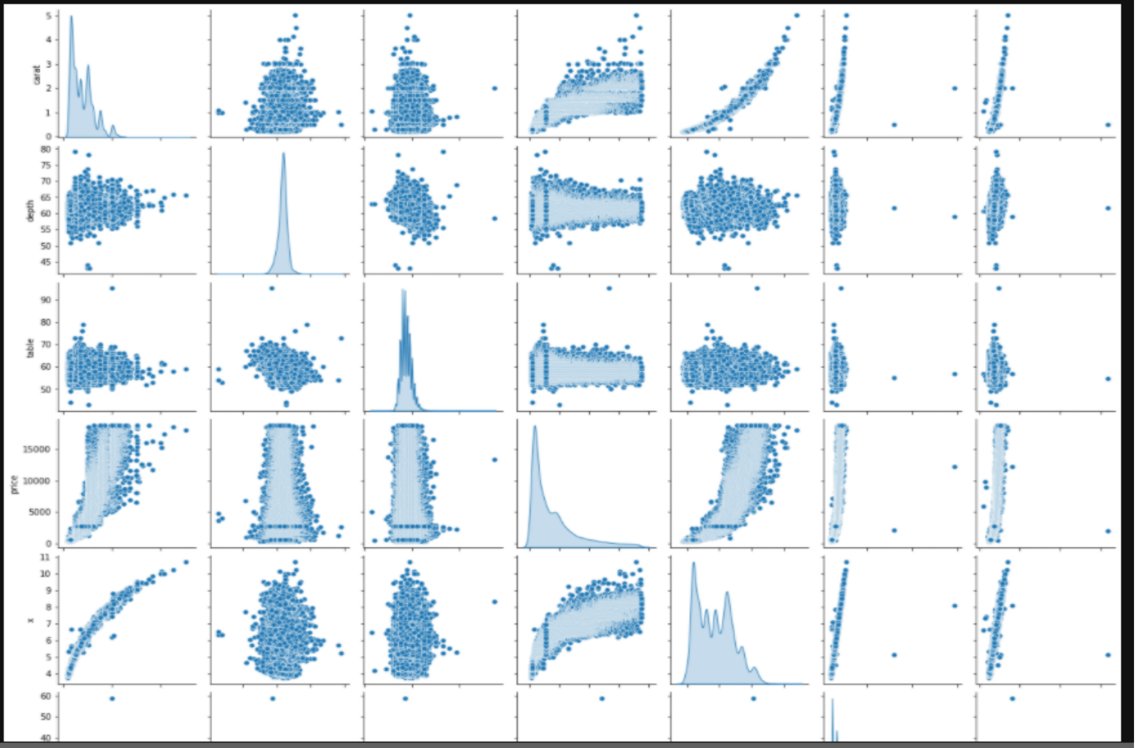
```
c:\users\amlan\appdata\local\programs\python\python39\lib\site-packages\pandas\plotting\_matplotlib\tools.py:
400: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be removed two minor releases later. Use
ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```



```
[23]:
```

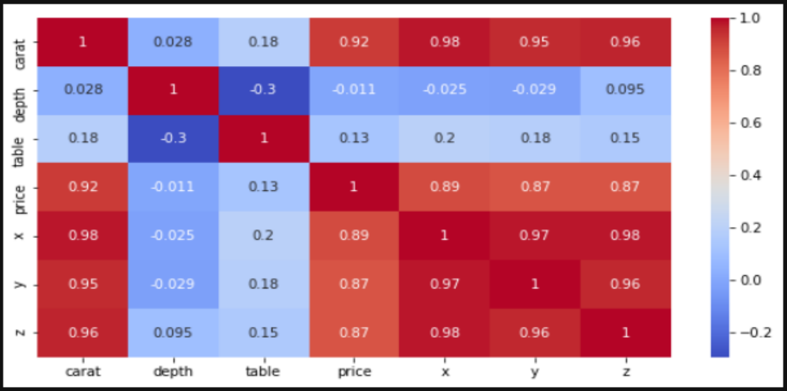
```
[24]: #Implemnet of seaborn.pairplot which is Visualizing Data with Pairs Plots
```

```
[25]: sns.pairplot(data , diag_kind = 'kde');
```

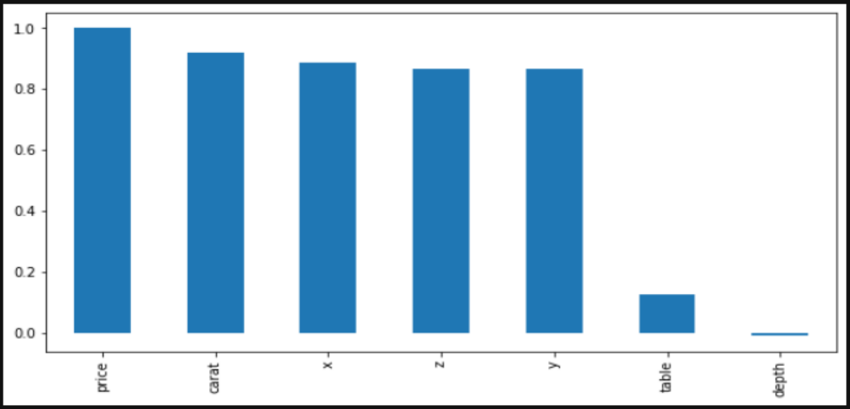


```
mple   0   1   Python 3 | Idle      Saving completed      Mode: Command      Ln 3, Col 11   Diamond_Price_Prediction_M
```

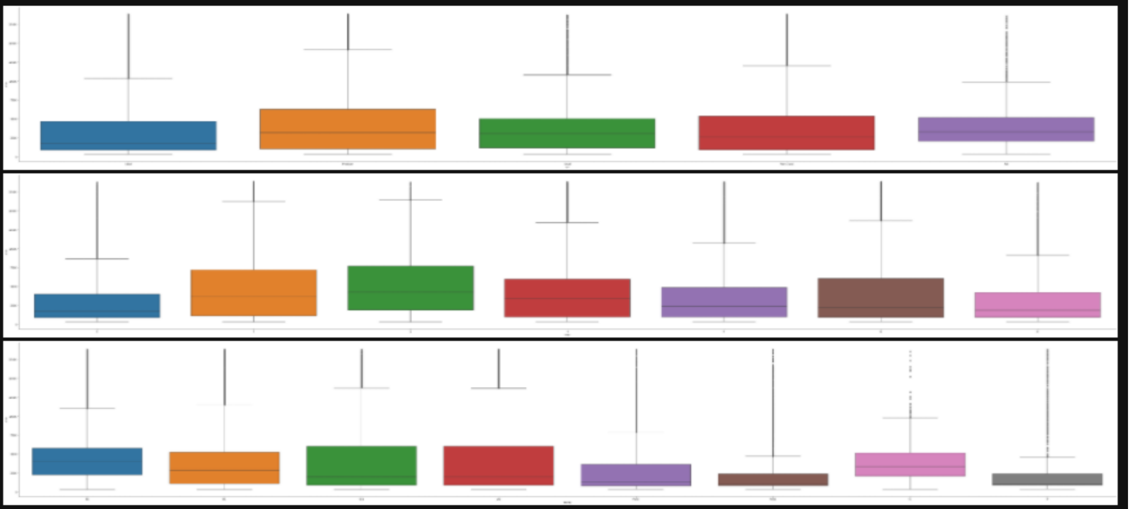
```
[28]: plt.figure(figsize = (10,5))
      sns.heatmap(data.corr(),annot = True , cmap = 'coolwarm' );
```



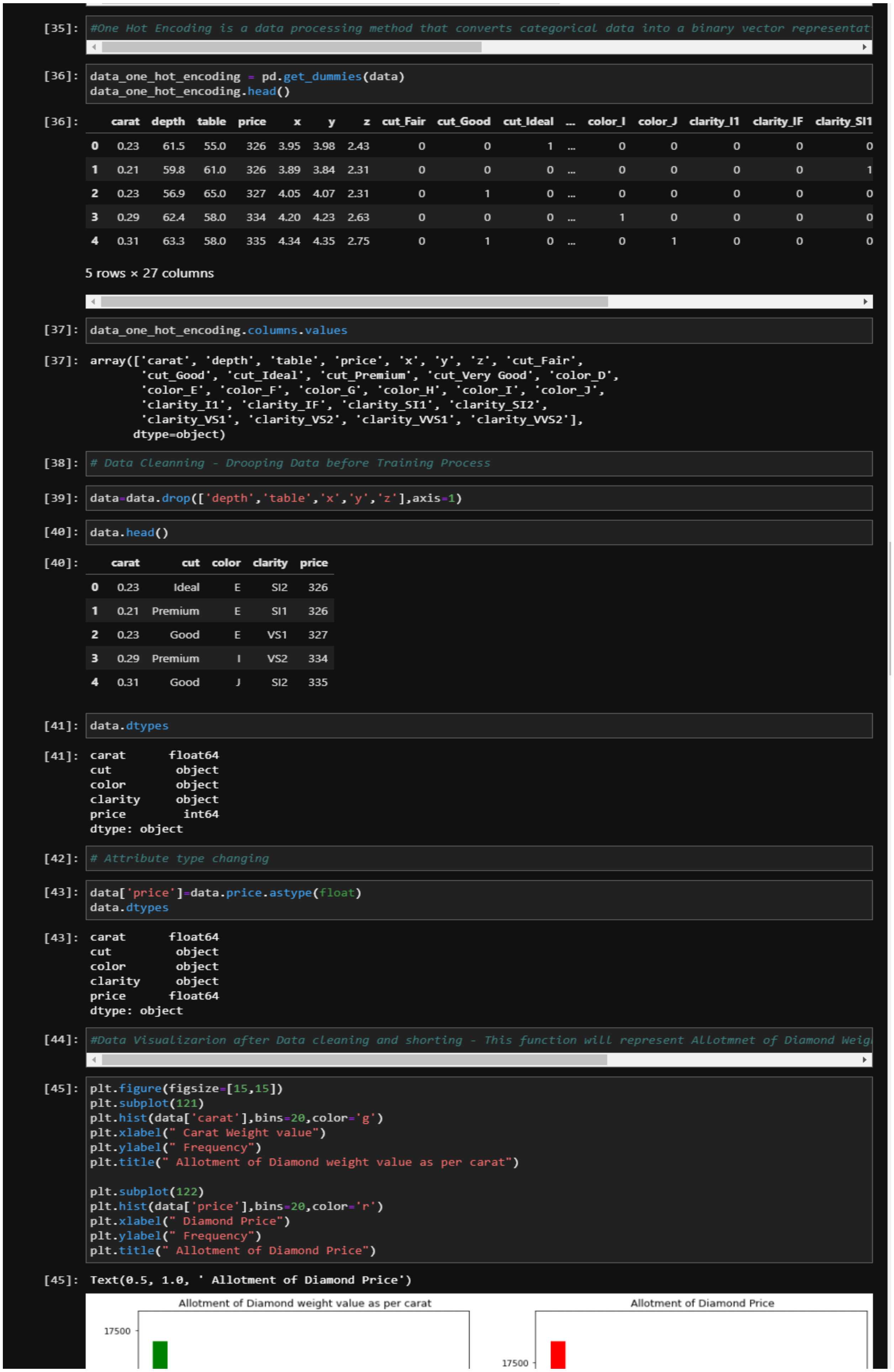
```
[29]:                                                   #(Sindiri, 2020)

[30]: #Implemnetation of Correlation Matrix

[31]: corr_mat = data.corr()
      plt.figure(figsize = (10,5))
      corr_mat['price'].sort_values(ascending = False).plot(kind = 'bar');
```



```
[32]: #Check which columns in DataFrame are Categorical and implement function Graph bar

[33]: input_cat_columns = data.select_dtypes(include = ['object']).columns.tolist()

      for col in input_cat_columns:
          sns.catplot(x=col, y="price",
                  kind="box", dodge=False, height = 10, aspect = 6,data=data);
```



```
[34]:

[35]: #One Hot Encoding is a data processing method that converts categorical data into a binary vector representat
```



```
[35]: #One Hot Encoding is a data processing method that converts categorical data into a binary vector representat
```

```
[36]: data_one_hot_encoding = pd.get_dummies(data)
      data_one_hot_encoding.head()
```

[36]:

	carat	depth	table	price	x	y	z	cut_Fair	cut_Good	cut_Ideal	...	color_I	color_J	clarity_I1	clarity_IF	clarity_SI1
0	0.23	61.5	55.0	326	3.95	3.98	2.43	0	0	1	...	0	0	0	0	0
1	0.21	59.8	61.0	326	3.89	3.84	2.31	0	0	0	...	0	0	0	0	1
2	0.23	56.9	65.0	327	4.05	4.07	2.31	0	1	0	...	0	0	0	0	0
3	0.29	62.4	58.0	334	4.20	4.23	2.63	0	0	0	...	1	0	0	0	0
4	0.31	63.3	58.0	335	4.34	4.35	2.75	0	1	0	...	0	1	0	0	0

5 rows × 27 columns

```
[37]: data_one_hot_encoding.columns.values
```

```
[37]: array(['carat', 'depth', 'table', 'price', 'x', 'y', 'z', 'cut_Fair',
             'cut_Good', 'cut_Ideal', 'cut_Premium', 'cut_Very Good', 'color_D',
             'color_E', 'color_F', 'color_G', 'color_H', 'color_I', 'color_J',
             'clarity_I1', 'clarity_IF', 'clarity_SI1', 'clarity_SI2',
             'clarity_VS1', 'clarity_VS2', 'clarity_VVS1', 'clarity_VVS2'],
            dtype=object)
```

```
[38]: # Data Cleanning - Drooping Data before Training Process
```

```
[39]: data=data.drop(['depth','table','x','y','z'],axis=1)
```

```
[40]: data.head()
```

[40]:

	carat	cut	color	clarity	price
0	0.23	Ideal	E	SI2	326
1	0.21	Premium	E	SI1	326
2	0.23	Good	E	VS1	327
3	0.29	Premium	I	VS2	334
4	0.31	Good	J	SI2	335

```
[41]: data.dtypes
```

```
[41]: carat       float64
      cut          object
      color        object
      clarity      object
      price         int64
      dtype: object
```

```
[42]: # Attribute type changing
```

```
[43]: data['price']=data.price.astype(float)
      data.dtypes
```

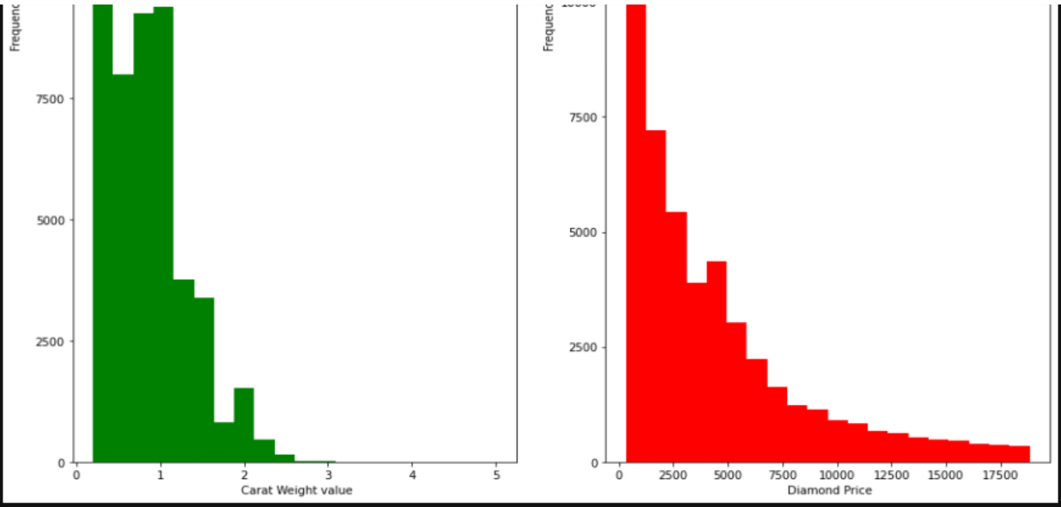
```
[43]: carat       float64
      cut          object
      color        object
      clarity      object
      price       float64
      dtype: object
```

```
[44]: #Data Visualizarion after Data cleaning and shorting - This function will represent Allotmnet of Diamond Weig
```

```
[45]: plt.figure(figsize=[15,15])
      plt.subplot(121)
      plt.hist(data['carat'],bins=20,color='g')
      plt.xlabel(" Carat Weight value")
      plt.ylabel(" Frequency")
      plt.title(" Allotment of Diamond weight value as per carat")

      plt.subplot(122)
      plt.hist(data['price'],bins=20,color='r')
      plt.xlabel(" Diamond Price")
      plt.ylabel(" Frequency")
      plt.title(" Allotment of Diamond Price")
```

```
[45]: Text(0.5, 1.0, ' Allotment of Diamond Price')
```

Allotment of Diamond weight value as per carat
17500
Allotment of Diamond Price
17500



```
[46]: #Reviewing first two data set value.
```

```
[47]: data.head(2)
```

[47]:

	carat	cut	color	clarity	price
0	0.23	Ideal	E	SI2	326.0
1	0.21	Premium	E	SI1	326.0

```
[48]: # Label Encoder converting catergorical data into numberic form. Transforming the prediction target (y). This
```

```
[49]: from sklearn.preprocessing import LabelEncoder
      l1=LabelEncoder()
      label=l1.fit_transform(data['cut'])
      l1.classes_
```

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

```
[50]: # Labelizing data to values
```

```
[51]: data['cut_label']=label
```

```
[52]: # Now representation data and justified the assigned valued displayed the tabel with number.
```

```
[53]: data.head()
```

[53]:

	carat	cut	color	clarity	price	cut_label
0	0.23	Ideal	E	SI2	326.0	2
1	0.21	Premium	E	SI1	326.0	3
2	0.23	Good	E	VS1	327.0	1
3	0.29	Premium	I	VS2	334.0	3
4	0.31	Good	J	SI2	335.0	1

```
[54]: # Same Labelizing and initiating number to Level as per desciption
```

```
[55]: l2=LabelEncoder()
      label1=l2.fit_transform(data['clarity'])
      data['clarity_label']=label1
      data.head()
```

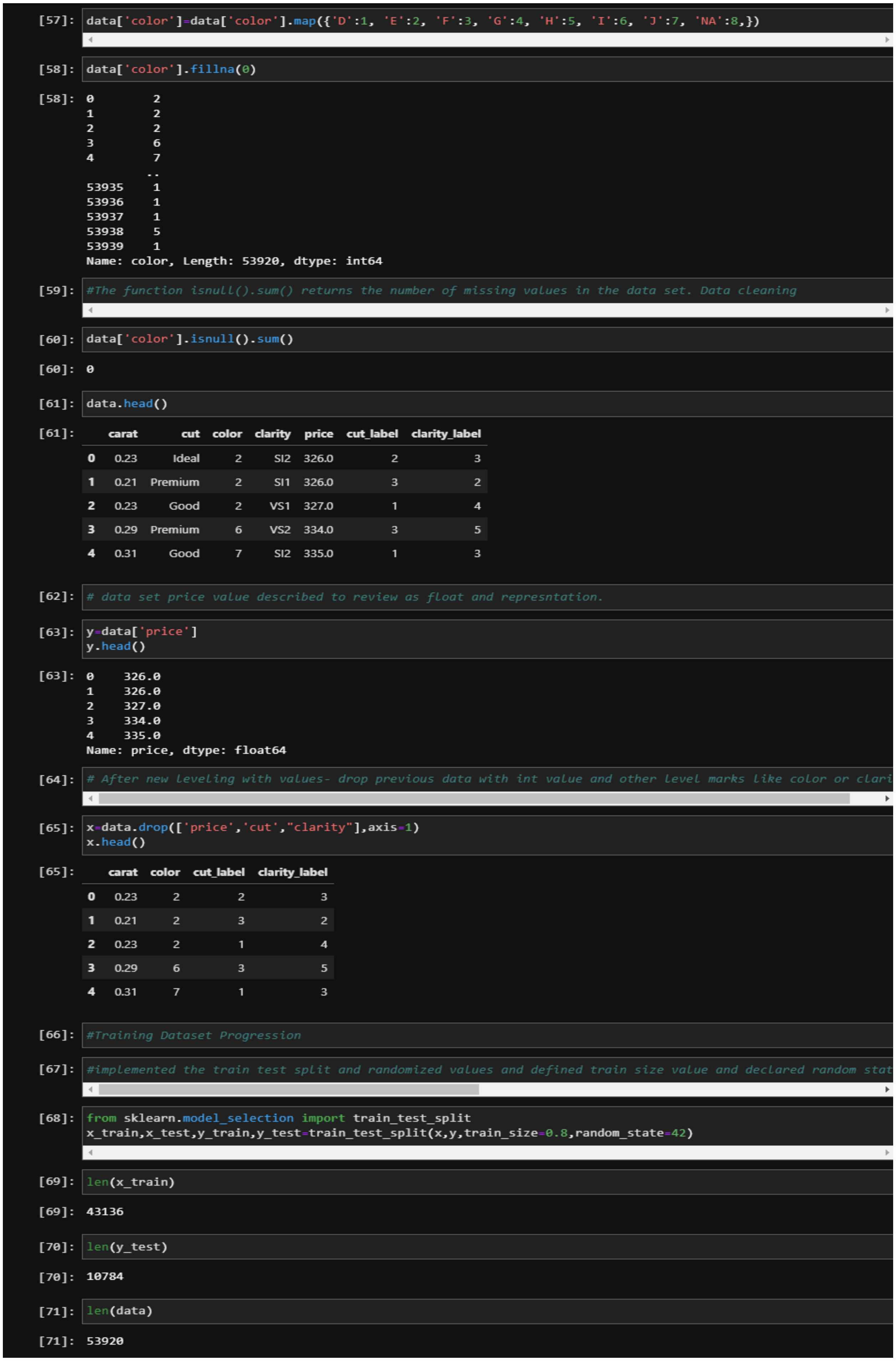
[55]:

	carat	cut	color	clarity	price	cut_label	clarity_label
0	0.23	Ideal	E	SI2	326.0	2	3
1	0.21	Premium	E	SI1	326.0	3	2
2	0.23	Good	E	VS1	327.0	1	4
3	0.29	Premium	I	VS2	334.0	3	5
4	0.31	Good	J	SI2	335.0	1	3

```
[56]: # Same Labelizing and initiating number to Level as per desciption - for color
```

```
[57]: data['color']=data['color'].map({'D':1, 'E':2, 'F':3, 'G':4, 'H':5, 'I':6, 'J':7, 'NA':8,})
```

imple 0 $_ 1 Python 3 | Idle Saving completed Mode: Command Ln 3, Col 11 Diamond_Price_Prediction_ML.ipynb



```
[57]: data['color']=data['color'].map({'D':1, 'E':2, 'F':3, 'G':4, 'H':5, 'I':6, 'J':7, 'NA':8,})
```

```
[58]: data['color'].fillna(0)
```

```
[58]: 0        2
      1        2
      2        2
      3        6
      4        7
              ..
      53935    1
      53936    1
      53937    1
      53938    5
      53939    1
      Name: color, Length: 53920, dtype: int64
```

```
[59]: #The function isnull().sum() returns the number of missing values in the data set. Data cleaning
```

```
[60]: data['color'].isnull().sum()
```

```
[60]: 0
```

```
[61]: data.head()
```

[61]:

	carat	cut	color	clarity	price	cut_label	clarity_label
0	0.23	Ideal	2	SI2	326.0	2	3
1	0.21	Premium	2	SI1	326.0	3	2
2	0.23	Good	2	VS1	327.0	1	4
3	0.29	Premium	6	VS2	334.0	3	5
4	0.31	Good	7	SI2	335.0	1	3

```
[62]: # data set price value described to review as float and represntation.
```

```
[63]: y=data['price']
      y.head()
```

```
[63]: 0    326.0
      1    326.0
      2    327.0
      3    334.0
      4    335.0
      Name: price, dtype: float64
```

```
[64]: # After new leveling with values- drop previous data with int value and other level marks like color or clari
```

```
[65]: x=data.drop(['price','cut',"clarity"],axis=1)
      x.head()
```

[65]:

	carat	color	cut_label	clarity_label
0	0.23	2	2	3
1	0.21	2	3	2
2	0.23	2	1	4
3	0.29	6	3	5
4	0.31	7	1	3

```
[66]: #Training Dataset Progression
```

```
[67]: #implemented the train test split and randomized values and defined train size value and declared random stat
```

```
[68]: from sklearn.model_selection import train_test_split
      x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=42)
```

```
[69]: len(x_train)
```

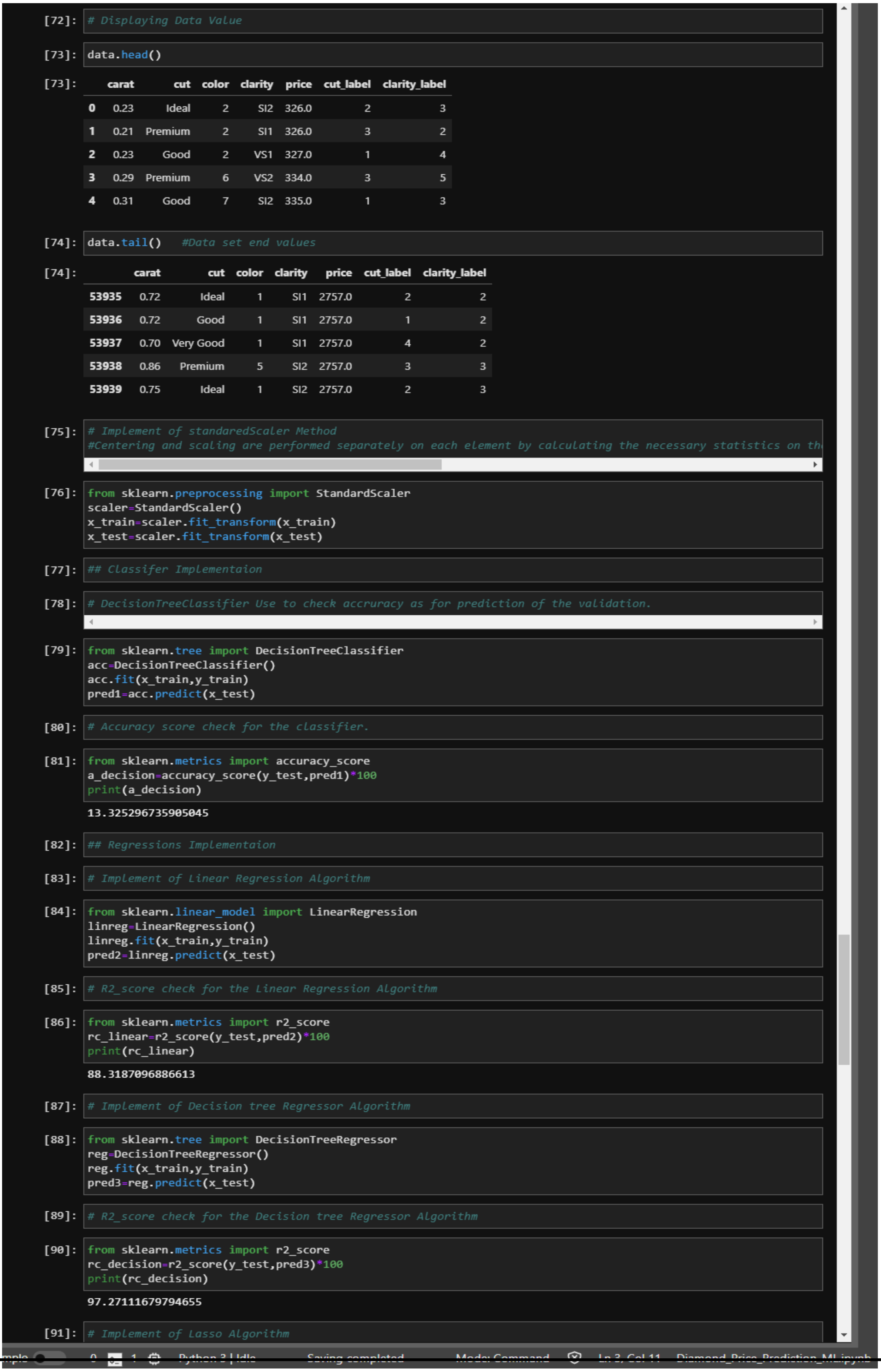
```
[69]: 43136
```

```
[70]: len(y_test)
```

```
[70]: 10784
```

```
[71]: len(data)
```

```
[71]: 53920
```



[72]:
```
# Displaying Data Value
```

[73]:
```
data.head()
```

[73]:

	carat	cut	color	clarity	price	cut_label	clarity_label
0	0.23	Ideal	2	SI2	326.0	2	3
1	0.21	Premium	2	SI1	326.0	3	2
2	0.23	Good	2	VS1	327.0	1	4
3	0.29	Premium	6	VS2	334.0	3	5
4	0.31	Good	7	SI2	335.0	1	3

[74]:
```
data.tail()   #Data set end values
```

[74]:

	carat	cut	color	clarity	price	cut_label	clarity_label
53935	0.72	Ideal	1	SI1	2757.0	2	2
53936	0.72	Good	1	SI1	2757.0	1	2
53937	0.70	Very Good	1	SI1	2757.0	4	2
53938	0.86	Premium	5	SI2	2757.0	3	3
53939	0.75	Ideal	1	SI2	2757.0	2	3

[75]:
```
# Implement of standaredScaler Method
#Centering and scaling are performed separately on each element by calculating the necessary statistics on th
```

[76]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

[77]:
```
## Classifer Implementaion
```

[78]:
```
# DecisionTreeClassifier Use to check accrruracy as for prediction of the validation.
```

[79]:
```
from sklearn.tree import DecisionTreeClassifier
acc=DecisionTreeClassifier()
acc.fit(x_train,y_train)
pred1=acc.predict(x_test)
```

[80]:
```
# Accuracy score check for the classifier.
```

[81]:
```
from sklearn.metrics import accuracy_score
a_decision=accuracy_score(y_test,pred1)*100
print(a_decision)
```
```
13.325296735905045
```

[82]:
```
## Regressions Implementaion
```

[83]:
```
# Implement of Linear Regression Algorithm
```

[84]:
```
from sklearn.linear_model import LinearRegression
linreg=LinearRegression()
linreg.fit(x_train,y_train)
pred2=linreg.predict(x_test)
```

[85]:
```
# R2_score check for the Linear Regression Algorithm
```

[86]:
```
from sklearn.metrics import r2_score
rc_linear=r2_score(y_test,pred2)*100
print(rc_linear)
```
```
88.3187096886613
```

[87]:
```
# Implement of Decision tree Regressor Algorithm
```

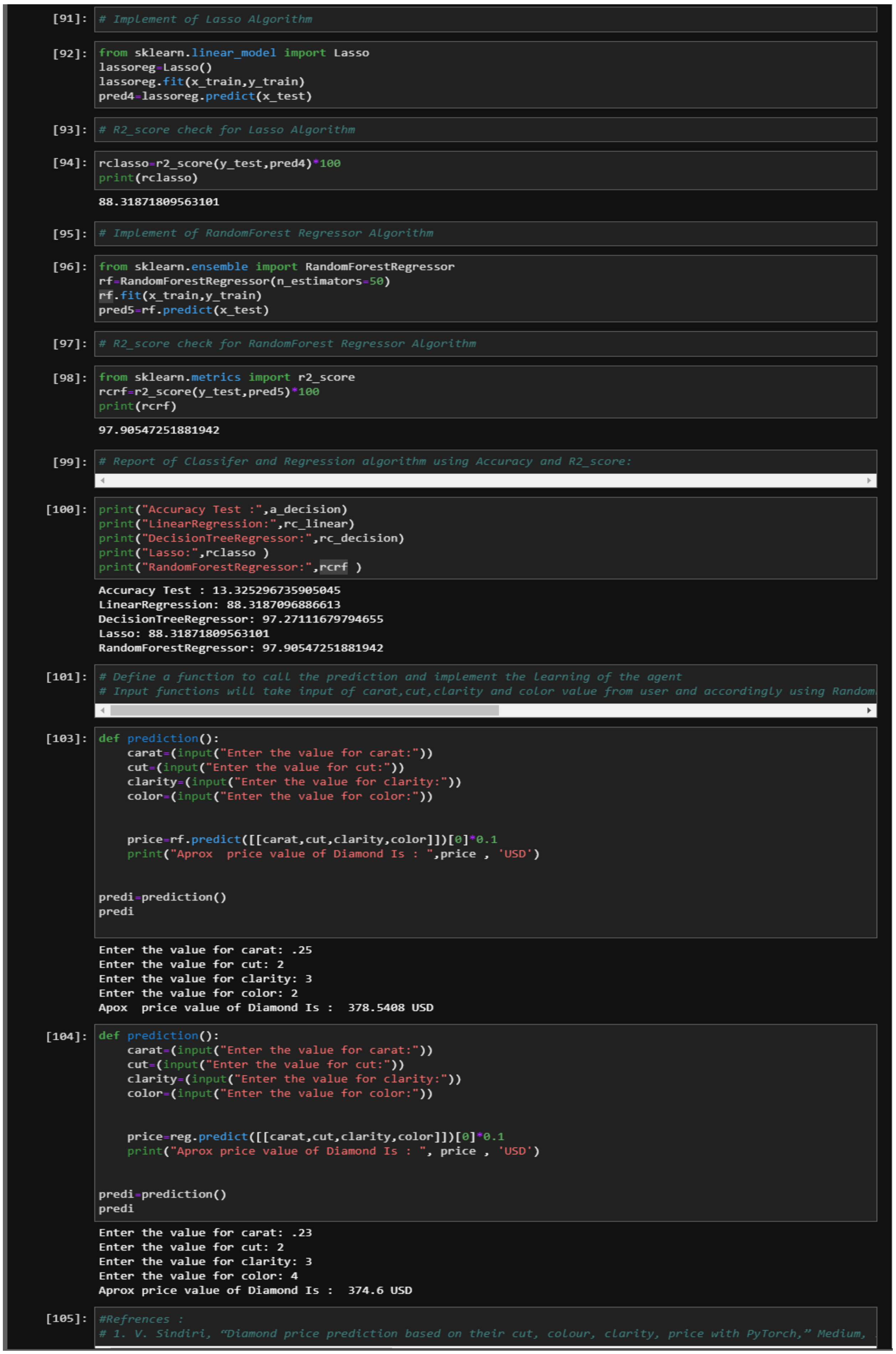
[88]:
```
from sklearn.tree import DecisionTreeRegressor
reg=DecisionTreeRegressor()
reg.fit(x_train,y_train)
pred3=reg.predict(x_test)
```

[89]:
```
# R2_score check for the Decision tree Regressor Algorithm
```

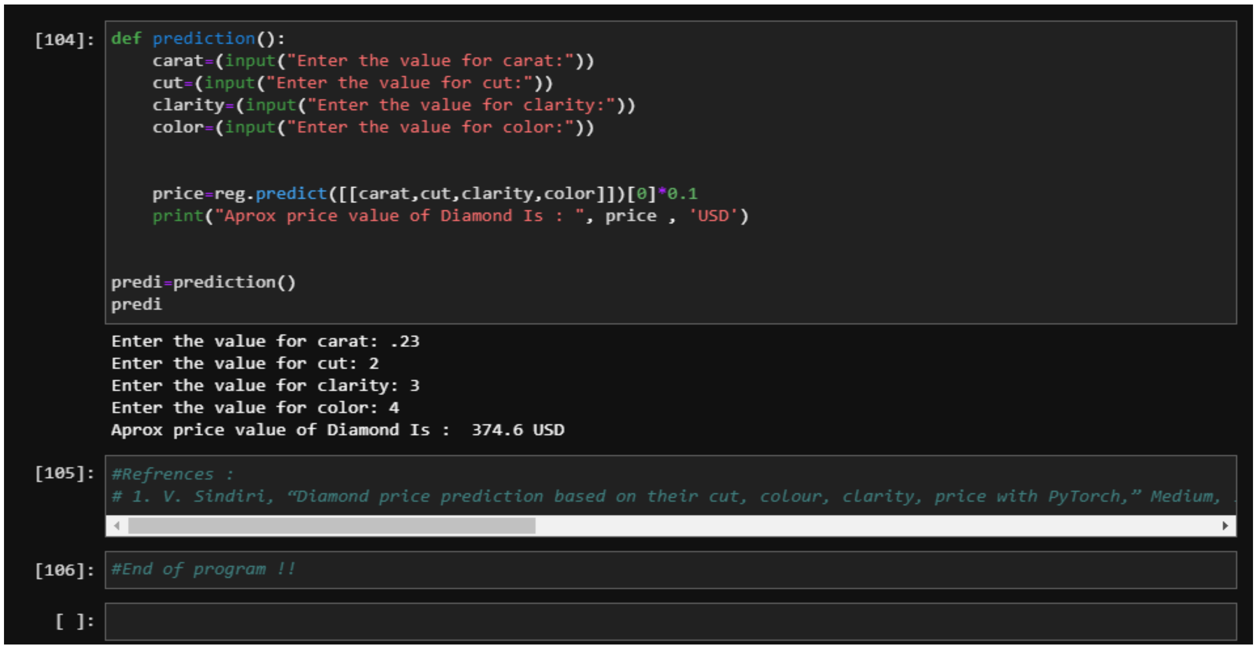
[90]:
```
from sklearn.metrics import r2_score
rc_decision=r2_score(y_test,pred3)*100
print(rc_decision)
```
```
97.27111679794655
```

[91]:
```
# Implement of Lasso Algorithm
```

mple 0 1 Python 3 | Idle Saving completed Mode: Command Ln 3, Col 11 Diamond_Price_Prediction_ML.ipynb



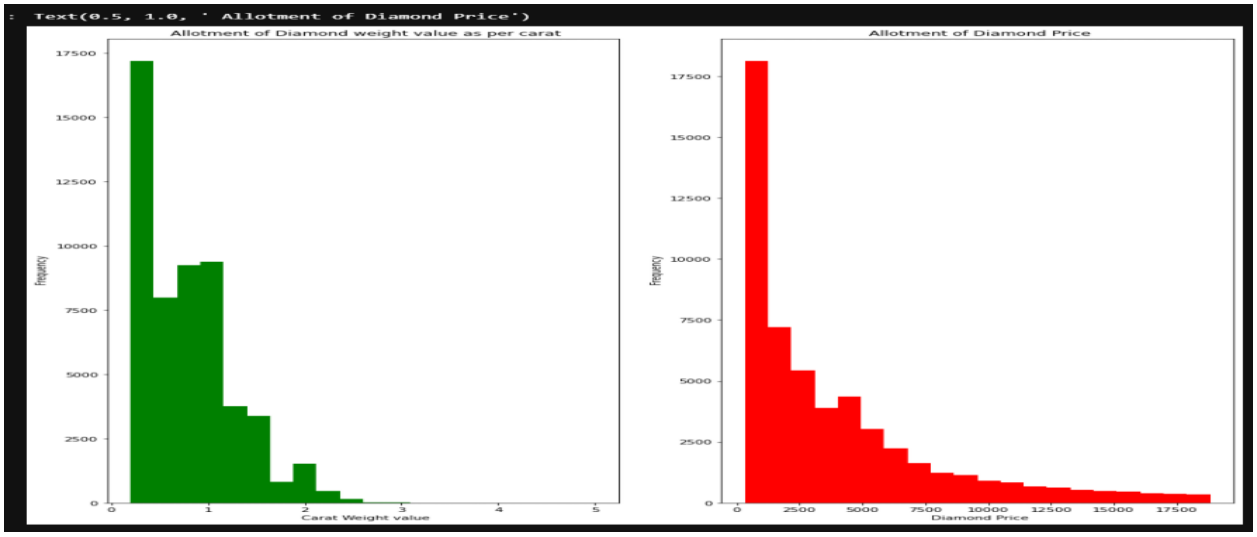
```
[91]: # Implement of Lasso Algorithm

[92]: from sklearn.linear_model import Lasso
      lassoreg=Lasso()
      lassoreg.fit(x_train,y_train)
      pred4=lassoreg.predict(x_test)

[93]: # R2_score check for Lasso Algorithm

[94]: rclasso=r2_score(y_test,pred4)*100
      print(rclasso)

      88.31871809563101

[95]: # Implement of RandomForest Regressor Algorithm

[96]: from sklearn.ensemble import RandomForestRegressor
      rf=RandomForestRegressor(n_estimators=50)
      rf.fit(x_train,y_train)
      pred5=rf.predict(x_test)

[97]: # R2_score check for RandomForest Regressor Algorithm

[98]: from sklearn.metrics import r2_score
      rcrf=r2_score(y_test,pred5)*100
      print(rcrf)

      97.90547251881942

[99]: # Report of Classifer and Regression algorithm using Accuracy and R2_score:

[100]: print("Accuracy Test :",a_decision)
       print("LinearRegression:",rc_linear)
       print("DecisionTreeRegressor:",rc_decision)
       print("Lasso:",rclasso )
       print("RandomForestRegressor:",rcrf )

       Accuracy Test : 13.325296735905045
       LinearRegression: 88.3187096886613
       DecisionTreeRegressor: 97.27111679794655
       Lasso: 88.31871809563101
       RandomForestRegressor: 97.90547251881942

[101]: # Define a function to call the prediction and implement the learning of the agent
       # Input functions will take input of carat,cut,clarity and color value from user and accordingly using Random

[103]: def prediction():
           carat=(input("Enter the value for carat:"))
           cut=(input("Enter the value for cut:"))
           clarity=(input("Enter the value for clarity:"))
           color=(input("Enter the value for color:"))


           price=rf.predict([[carat,cut,clarity,color]])[0]*0.1
           print("Aprox  price value of Diamond Is : ",price , 'USD')


       predi=prediction()
       predi

       Enter the value for carat: .25
       Enter the value for cut: 2
       Enter the value for clarity: 3
       Enter the value for color: 2
       Apox  price value of Diamond Is :  378.5408 USD

[104]: def prediction():
           carat=(input("Enter the value for carat:"))
           cut=(input("Enter the value for cut:"))
           clarity=(input("Enter the value for clarity:"))
           color=(input("Enter the value for color:"))


           price=reg.predict([[carat,cut,clarity,color]])[0]*0.1
           print("Aprox price value of Diamond Is : ", price , 'USD')


       predi=prediction()
       predi

       Enter the value for carat: .23
       Enter the value for cut: 2
       Enter the value for clarity: 3
       Enter the value for color: 4
       Aprox price value of Diamond Is :  374.6 USD

[105]: #Refrences :
       # 1. V. Sindiri, "Diamond price prediction based on their cut, colour, clarity, price with PyTorch," Medium,
```



```
[104]: def prediction():
           carat=(input("Enter the value for carat:"))
           cut=(input("Enter the value for cut:"))
           clarity=(input("Enter the value for clarity:"))
           color=(input("Enter the value for color:"))


           price=reg.predict([[carat,cut,clarity,color]])[0]*0.1
           print("Aprox price value of Diamond Is : ", price , 'USD')


       predi=prediction()
       predi

       Enter the value for carat: .23
       Enter the value for cut: 2
       Enter the value for clarity: 3
       Enter the value for color: 4
       Aprox price value of Diamond Is :  374.6 USD

[105]: #Refrences :
       # 1. V. Sindiri, "Diamond price prediction based on their cut, colour, clarity, price with PyTorch," Medium,

[106]: #End of program !!

[ ]:
```

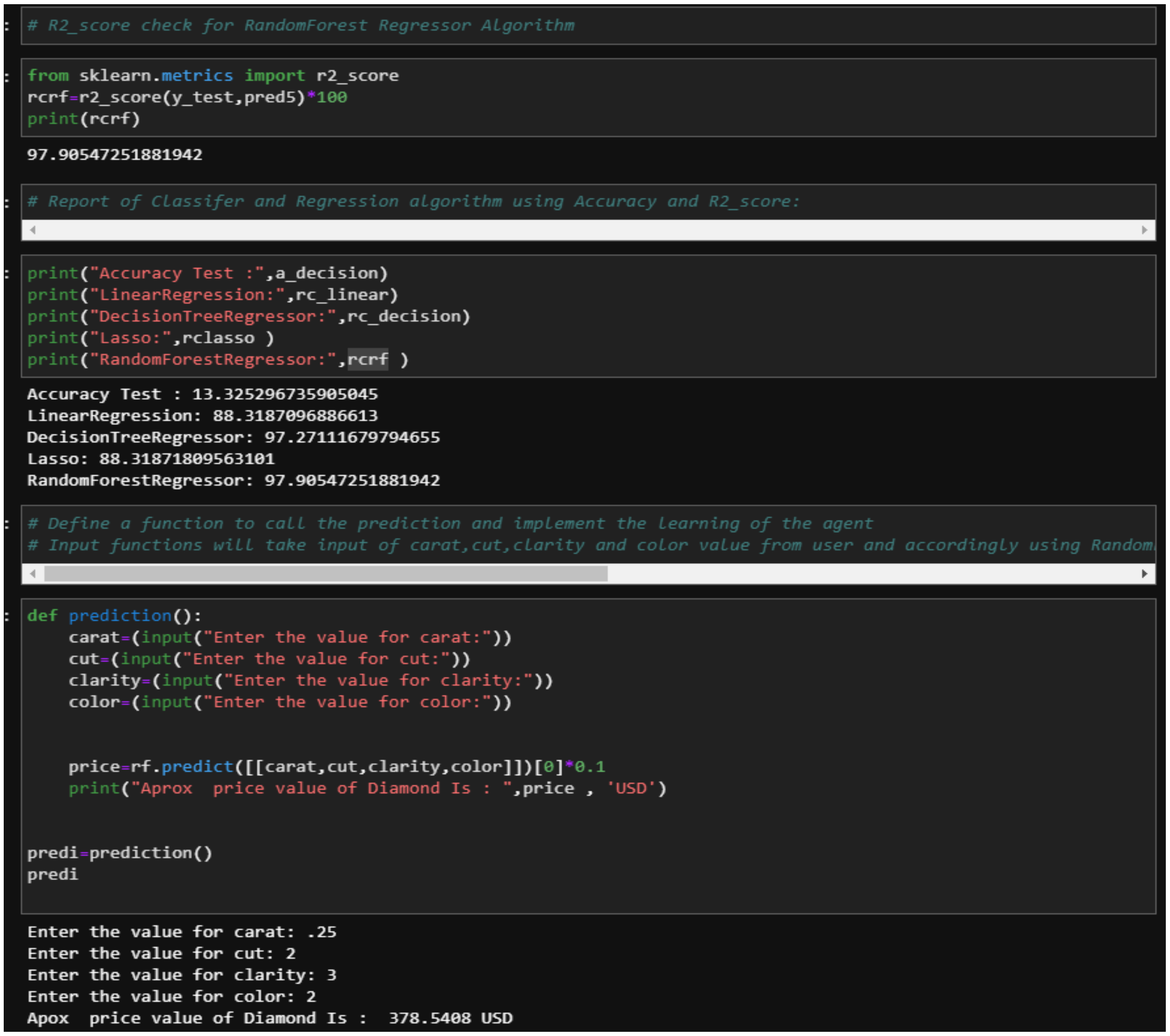
Results

After executing a trained ML agent, it is clear that now the agent can predict the diamond price as per customer preferences. It can also choose the best fit and accurate regression model to predict the value.



Here, from two graphs, allotment of diamond weight value as per carat and allotment of price.

Though the price value frequency and diamond price growth together, but time to time, diamond price increased as per its value and same as the diamond weight has been increased.



```
# R2_score check for RandomForest Regressor Algorithm
```

```
from sklearn.metrics import r2_score
rcrf=r2_score(y_test,pred5)*100
print(rcrf)
```

```
97.90547251881942
```

```
# Report of Classifer and Regression algorithm using Accuracy and R2_score:
```

```
print("Accuracy Test :",a_decision)
print("LinearRegression:",rc_linear)
print("DecisionTreeRegressor:",rc_decision)
print("Lasso:",rclasso )
print("RandomForestRegressor:",rcrf )
```

```
Accuracy Test : 13.325296735905045
LinearRegression: 88.3187096886613
DecisionTreeRegressor: 97.27111679794655
Lasso: 88.31871809563101
RandomForestRegressor: 97.90547251881942
```

```
# Define a function to call the prediction and implement the learning of the agent
# Input functions will take input of carat,cut,clarity and color value from user and accordingly using Random
```

```
def prediction():
    carat=(input("Enter the value for carat:"))
    cut=(input("Enter the value for cut:"))
    clarity=(input("Enter the value for clarity:"))
    color=(input("Enter the value for color:"))


    price=rf.predict([[carat,cut,clarity,color]])[0]*0.1
    print("Aprox  price value of Diamond Is : ",price , 'USD')


predi=prediction()
predi
```

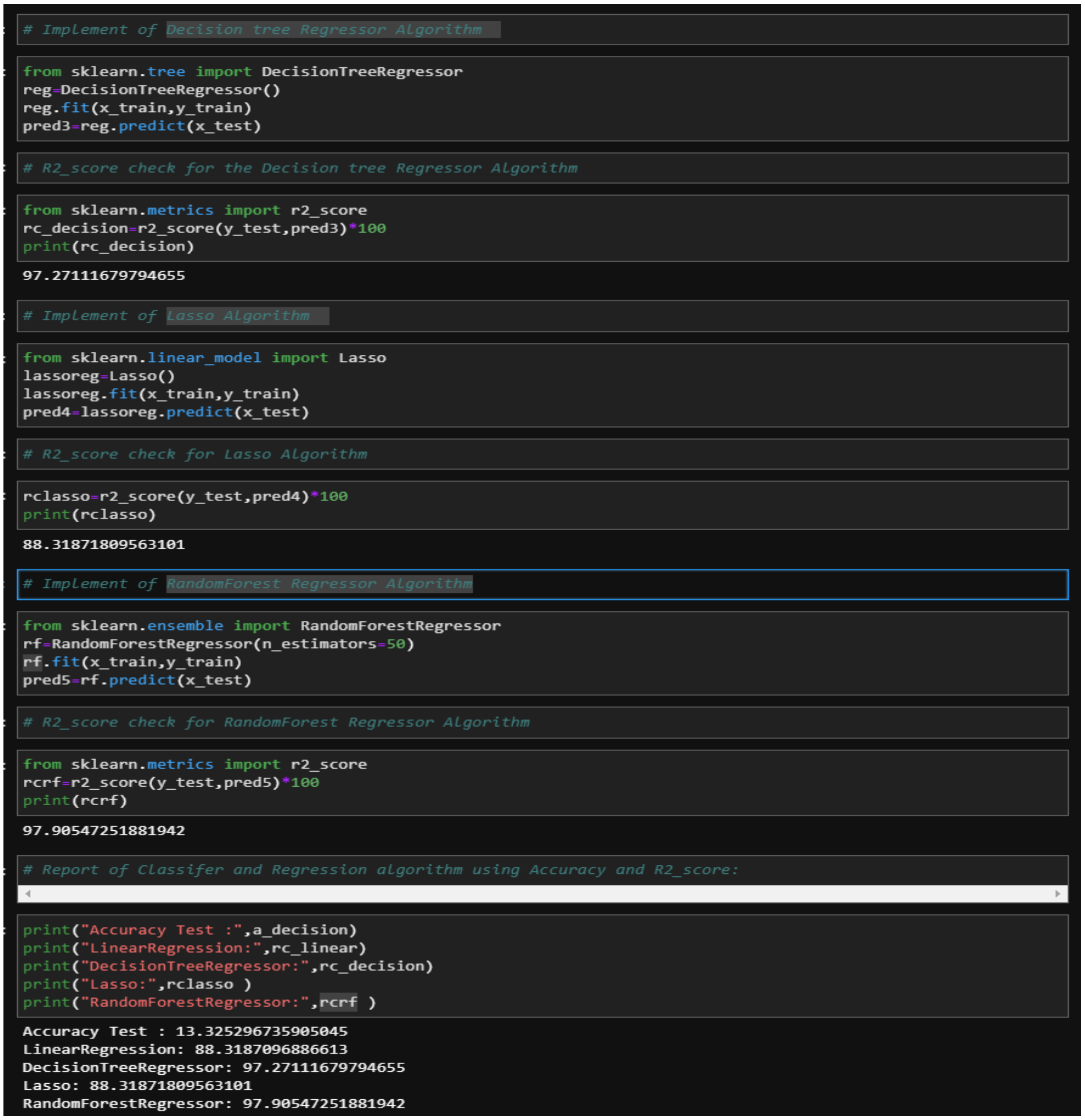
```
Enter the value for carat: .25
Enter the value for cut: 2
Enter the value for clarity: 3
Enter the value for color: 2
Apox  price value of Diamond Is :  378.5408 USD
```

As my Machine learning technique mainly focused on regression and Random Forest Regressor has an accurate top rate, it is clear that this model can predict the nearest value of future diamond as per specifications. Also, it is a more precise outcome that the second most successful model for this project is Decision tree regression. Throughout all, model r2_score was able to generate the most efficient and effective model value. Then finally, we can predict the most relevant value for diamonds.

Evaluation

In this project, mainly regression models are used. As discussed, especially one classifier and couple regression model used to find the more accurate value. For validation part primarily focused on accuracy tests and r2_score values. Mainly DecisionTreeClassifier algorithm as a classifier and also validated the value by accuracy tester.

Then I have used various regression tools for the validation part. This project is based on linear model regression and also implemented with other regression models. Then implementation of r2_score, which is a coefficient of determination function that regression score function. The best possible score is 1.0, and it can be negative. A constant model that always predicts the expected value of y, disregarding the input features, would get an R^2 score of 0.0. Then I have subsequently applied - Linear Regression Algorithm, Decision tree Regressor Algorithm, Lasso Algorithm and RandomForest Regressor Algorithm where I set the same standard for each regression model where the program is assigned to the specific model and assigned to x_train and y_train and set a prediction method to generate the validation score.



```
# Implement of Decision tree Regressor Algorithm
```

```
from sklearn.tree import DecisionTreeRegressor
reg=DecisionTreeRegressor()
reg.fit(x_train,y_train)
pred3=reg.predict(x_test)
```

```
# R2_score check for the Decision tree Regressor Algorithm
```

```
from sklearn.metrics import r2_score
rc_decision=r2_score(y_test,pred3)*100
print(rc_decision)
```

```
97.27111679794655
```

```
# Implement of Lasso Algorithm
```

```
from sklearn.linear_model import Lasso
lassoreg=Lasso()
lassoreg.fit(x_train,y_train)
pred4=lassoreg.predict(x_test)
```

```
# R2_score check for Lasso Algorithm
```

```
rclasso=r2_score(y_test,pred4)*100
print(rclasso)
```

```
88.31871809563101
```

```
# Implement of RandomForest Regressor Algorithm
```

```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor(n_estimators=50)
rf.fit(x_train,y_train)
pred5=rf.predict(x_test)
```

```
# R2_score check for RandomForest Regressor Algorithm
```

```
from sklearn.metrics import r2_score
rcrf=r2_score(y_test,pred5)*100
print(rcrf)
```

```
97.90547251881942
```

```
# Report of Classifer and Regression algorithm using Accuracy and R2_score:
```

```
print("Accuracy Test :",a_decision)
print("LinearRegression:",rc_linear)
print("DecisionTreeRegressor:",rc_decision)
print("Lasso:",rclasso )
print("RandomForestRegressor:",rcrf )
```

```
Accuracy Test : 13.325296735905045
LinearRegression: 88.3187096886613
DecisionTreeRegressor: 97.27111679794655
Lasso: 88.31871809563101
RandomForestRegressor: 97.90547251881942
```

Discussion and Conclusion

From the background research, previously developed projects, and another developer analyzed the report, I have determined my project target and prediction target. From my observation and analysis, I have tried to implement proper machine learning techniques. Also, consider all facts that work behind the diamond’s price and features. My target was to develop a project using Machine learning that would help determine the diamond price based on color, clarity, cut, and diamond. So, I analyzed the entire data set and generated graphs that helped me understand the price's facts. I took the necessary actions to clean up the data set and train accordingly to fit transform on my level set. I determined that the regression model would be the best fit for this Machine learning project from the analysis. From there, I found the best accurate rate for this project provided by the RandomForest Regressor model. My project is entirely a machine learning project. Because it just does not analyze the statistic. It adopts the data by train method and then observes the whole previous data set. I then implemented various regression models and validated them with a score. Finally, the user will get the prediction price using the best-fit regression model as it's a transparent machine learning model. It can consent all data set observations in memory and then take input from users as their preferences of diamond quality and specification then redirect the prediction price. So hence, it can clearly be said that the project is a clear reflection of a machine learning project.

Future Plan

After training and developing the machine Learning AI agent model, I am motivated by an AI system's power, precision, and prediction capability. If I can develop an Android or IOS app and Web Application based on this prediction which will mainly focus on predicting future diamond prices. Then, I summarized my idea and figured it out just for functional user purposes. If I developed a mobile app, maybe a few people would get interested in downloading it - mostly jewellers but not customers, perhaps because they don't buy diamonds every day they need an app. Customers or vendors or marketing agents, or business people always search in google about diamond current value and information. So it would be wise to develop a dynamic website (using Webapp) platform where I can regularly update data and continue development as per the dataset change and market fluctuation. So, I would develop a website. I will also spend more time observing more parts of deep learning, ad experimenting with other classifier and regression tools to find the top few fittest models for this project.

References

1. Sindiri, V. (2020, June 16). *Diamond price prediction based on their cut, colour, clarity, price with PyTorch*. Medium. https://towardsdatascience.com/diamond-price-prediction-based-on-their-cut-colour-clarity-price-with-pytorch-1e0353d2503b
2. Grover, J. (2021, March 28). *"Diamonds are forever" — price prediction using Machine Learning regression models and neural networks.* Medium. https://groverjatin.medium.com/diamonds-are-forever-price-prediction-using-machine-learning-regression-models-and-neural-f4f54b5fcf7f
3. *Build software better, together*. (2020). GitHub. https://github.com/GokuMohandas/madewithml/blob/main/notebooks/07_Linear_Regression/07_PT_Linear_Regression.ipynb
4. *Deep Learning with PyTorch: Zero to GANs*. (2020). Jovian Community. https://jovian.ai/forum/c/pytorch-zero-to-gans/18
5. *Diamonds*. (2017, May 25). Kaggle. https://www.kaggle.com/shivam2503/diamonds