REPORT ON DIAMOND PRICE PREDICTION

S. NO.	Topic	

1	Introduction
	
	• Business Problems We are trying to solve
	• Main goal
	• About Dataset
	  ○ Table1- DataSet Preview
	  ○ Table 2 - Columns in Dataset	

2	EDA & Business Implication
	
	• Analysis of the Data
	• Table 3- Checking of the Skewness in Data
	• Univariant Analysis – BoxPlot
	  ○ Plot 1- Boxplot
	• Analysis columns
	  ○ Plot 2 – Histogram of numical data
	  ○ Plot 3- Histogram of unnumical data
	• Bivariant Analysis – Pairplot
	  ○ Plot 17 - PairPlot
	• Bivariant Analysis – Heatmap
	  ○ Plot 18 - Heatmap	

3	Date Cleaning and Preprocessing
	
	• Approach using for identifying and treating missing values and outlier treatement
	  ○ Code Sample 1- Null Values
	  ○ Code Sample 2- Missing Values
	  ○ Code Sample 3- Outlier Detection
	  ○ Table 3- Outlier Detection
	• Need For variable transformation
	  ○ Code Sample 4 – Variable Transformation
	• Variables removed or added and why?	

4	Model Building
	
	• Model Selection and Why?
	  ○ Code Sample 5- Splitting Data
	  ○ Code Sample 6- List of Algo used
	  ○ Table 5 i- Algorithm Score
	  ○ Table 5 ii- Algorithm Score	

	○ Flow Diagram 1- Flow of the Model Used and Why ○ Table 5 iii- Algorithm Score ○ Table 5 iv- Algorithm Score • Efforts to improve model performance • Hyperparameter Tuning ○ Code Sample 7- Best parameter grid ○ Code Sample 8- Hyparparamter tuning ○ Table 5 v- Algorithm Score	
5	Model Validation • Checking significance of variable using p-value • Performance Metrices ○ Table 5 vi- Algorithm Performance	
6	Final Interpretation/Recommendation • Table 6 - Final Algorithm Performance Table • Recommendation • Final Words	

1. Introduction

Jewelery and diamonds are among the most valuable pieces that humans deal with, and they have been used since ancient times, whether for decoration or saving. Determining the price of diamonds depends on several factors, and understanding these factors enables the person concerned to determine the optimal price.

Business Problems We are Trying to Solve

- Problems we generally face during buying a diamond like:
    - We are not aware of the general factors that affect diamond prices
    - The precision of polishing plays an important role in determining the price
    - Here we forecast the selling price of diamonds based on the previous sold diamonds.
    - We also deduce what factors affect diamonds.

Main Goal

- Create an analytical framework to understand
    - Key factors impacting diamonds price.
- Develop a modelling framework
    - To estimate the price of a diamond that is up for sale

About Dataset

The dataset consisted of registries of different diamonds sold in the United States. It has different columns describing the diamond such as color, depth, flatness, dimensions and price. This data set is very useful for the diamond companies and the people who want to buy, they can also predict the price with the help of machine learning models.

This is what the dataset looks like,

```
[4] data.head()
```

	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

Table1- DataSet

It has lots of rows and columns, which is 53940 rows spread across 11 columns. Here is the list of columns this dataset has,

```
data.columns
```

```
Index(['Unnamed: 0', 'carat', 'cut', 'color', 'clarity', 'depth', 'table',
       'price', 'x', 'y', 'z'],
      dtype='object')
```

Table 2 - Columns in

2. EDA & Business Implication

EDA stands for exploratory data analysis where we explore our data and grab insights from it. EDA helps us in getting knowledge in form of various plots and diagrams where we can easily understand the data and its features.

Analysis Of the Data

	count	mean	std	min	25%	50%	75%	max
carat	53794.0	0.797780	0.473390	0.2	0.40	0.70	1.04	5.01
depth	53794.0	61.748080	1.429909	43.0	61.00	61.80	62.50	79.00
table	53794.0	57.458109	2.233679	43.0	56.00	57.00	59.00	95.00
price	53794.0	3933.065082	3988.114460	326.0	951.00	2401.00	5326.75	18823.00
x	53794.0	5.731214	1.120695	0.0	4.71	5.70	6.54	10.74
y	53794.0	5.734653	1.141209	0.0	4.72	5.71	6.54	58.90
z	53794.0	3.538714	0.705037	0.0	2.91	3.53	4.03	31.80

Table 3- Checking of the Skewness

Observation:

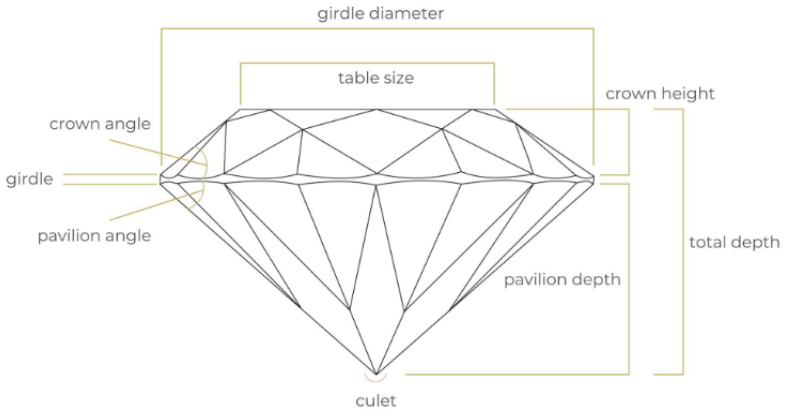
price: Our target column value is in 326 - 18823 range.

depth: the depth of diamonds ranges from 43 - 79.

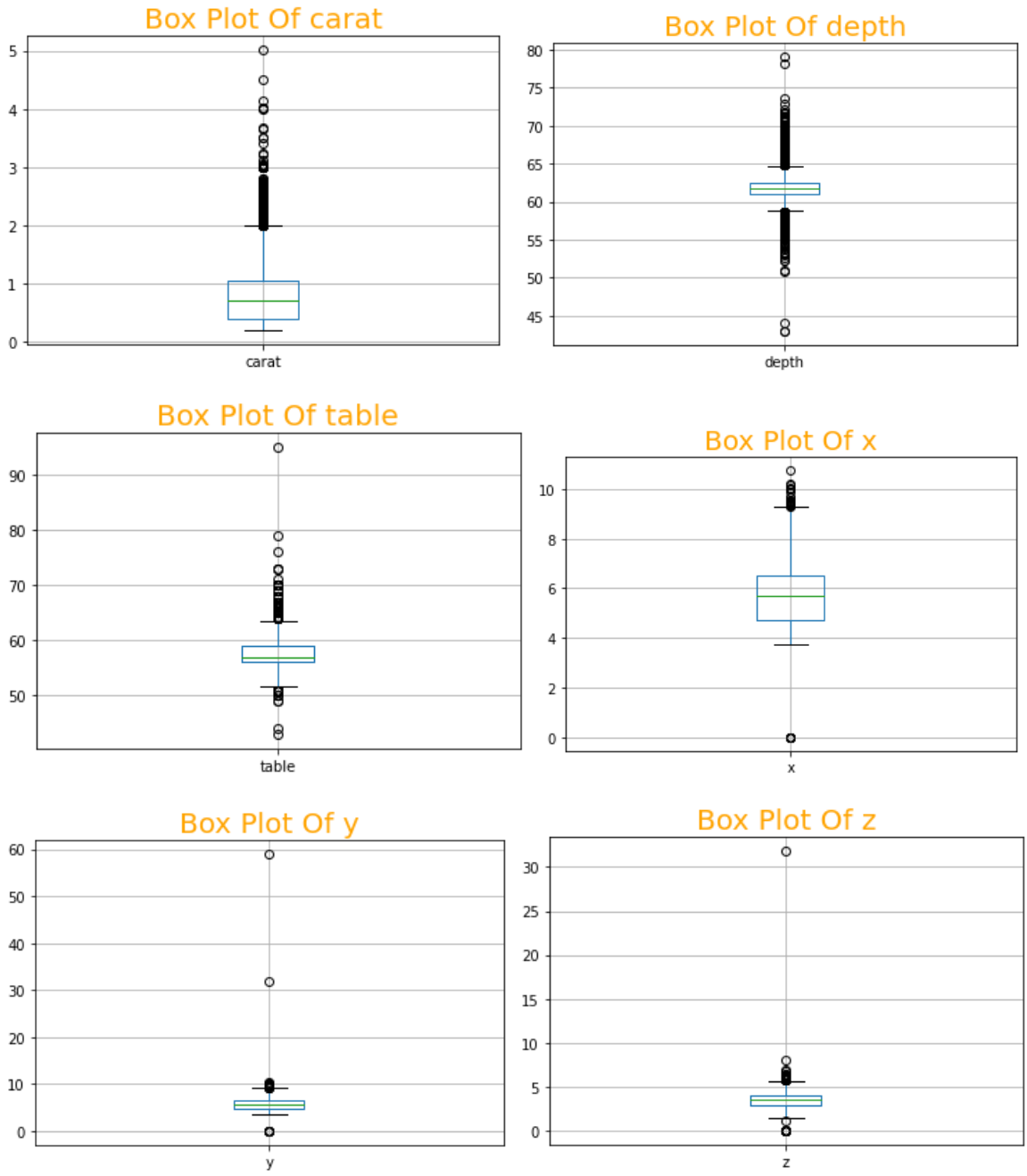
table: Measurement of the upper surface of the diamond range from 43 - 95.

carat: Measurement of diamond precision range from 0.7 - 5.

x,y,z: Measurement of Diamond Dimensions range .



Uni-variate Analysis - By BoxPlot



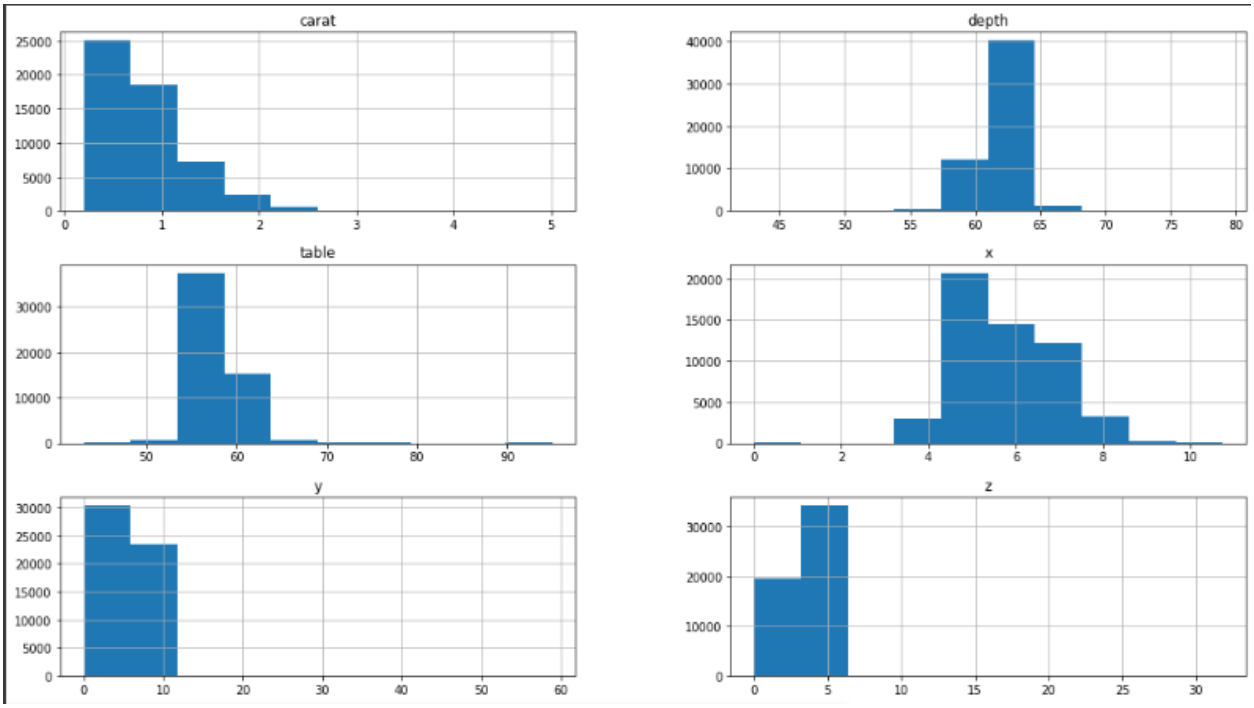
Plot 1 – Uni-variate Analysis -

Observation:

There are lots of feature which are having outliers**.**

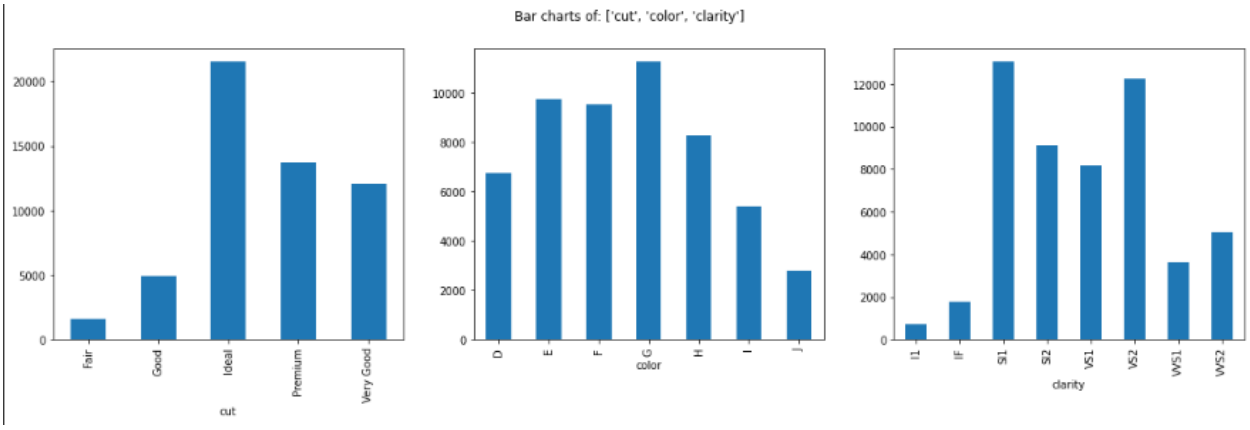
Analysis Column

Histogram of numerical columns



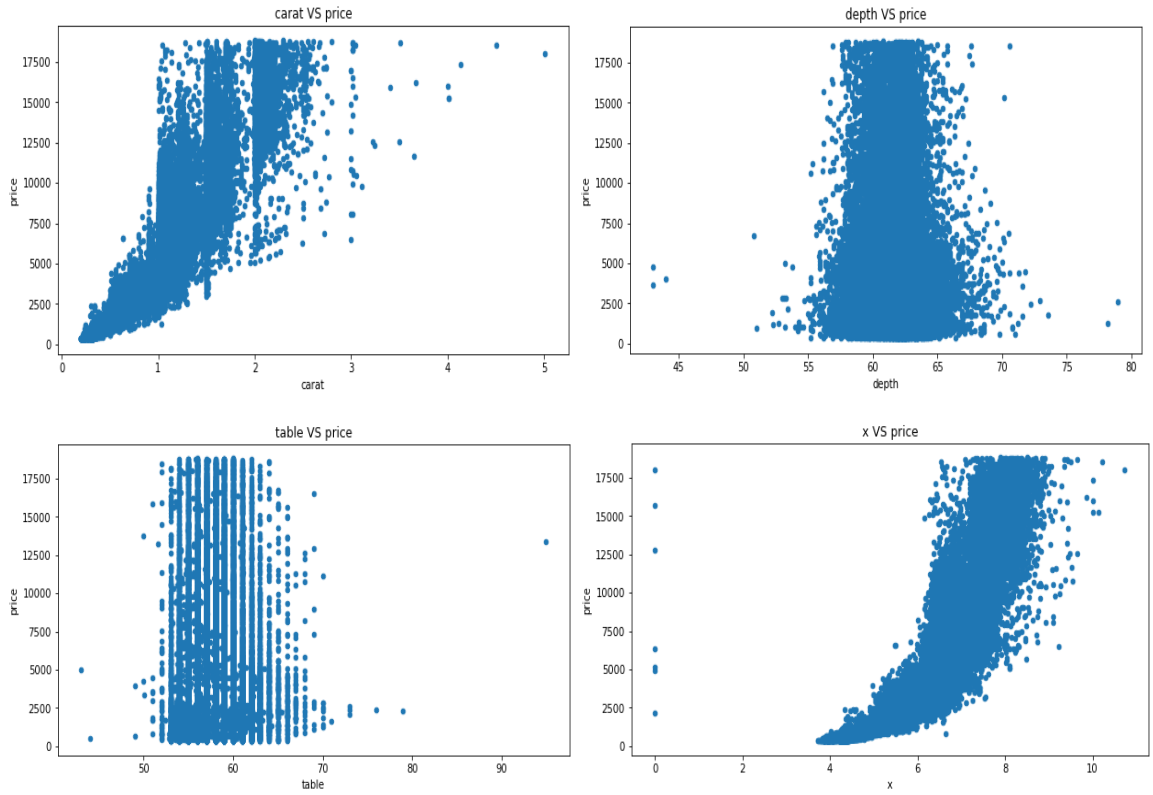
Plot 2 – Histogram of numerical

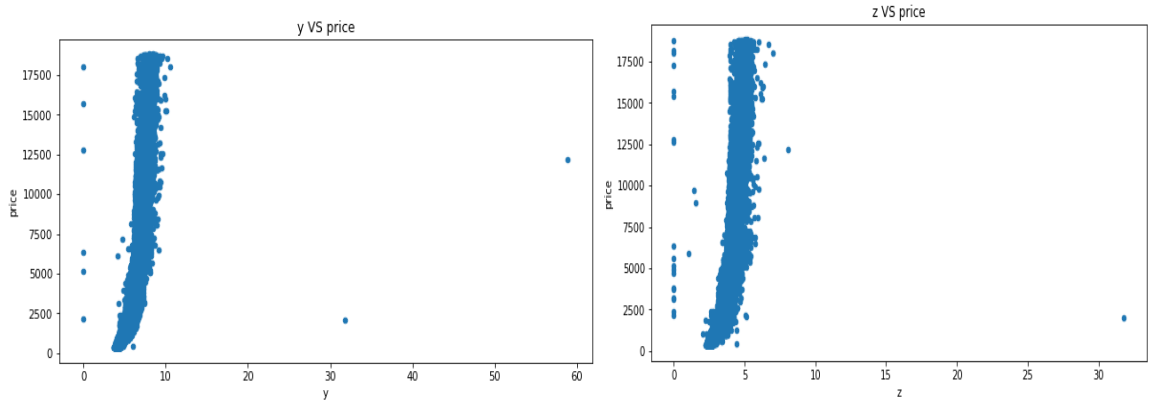
Histogram of unnumerical columns



Plot 2 – Histogram of unnumerical

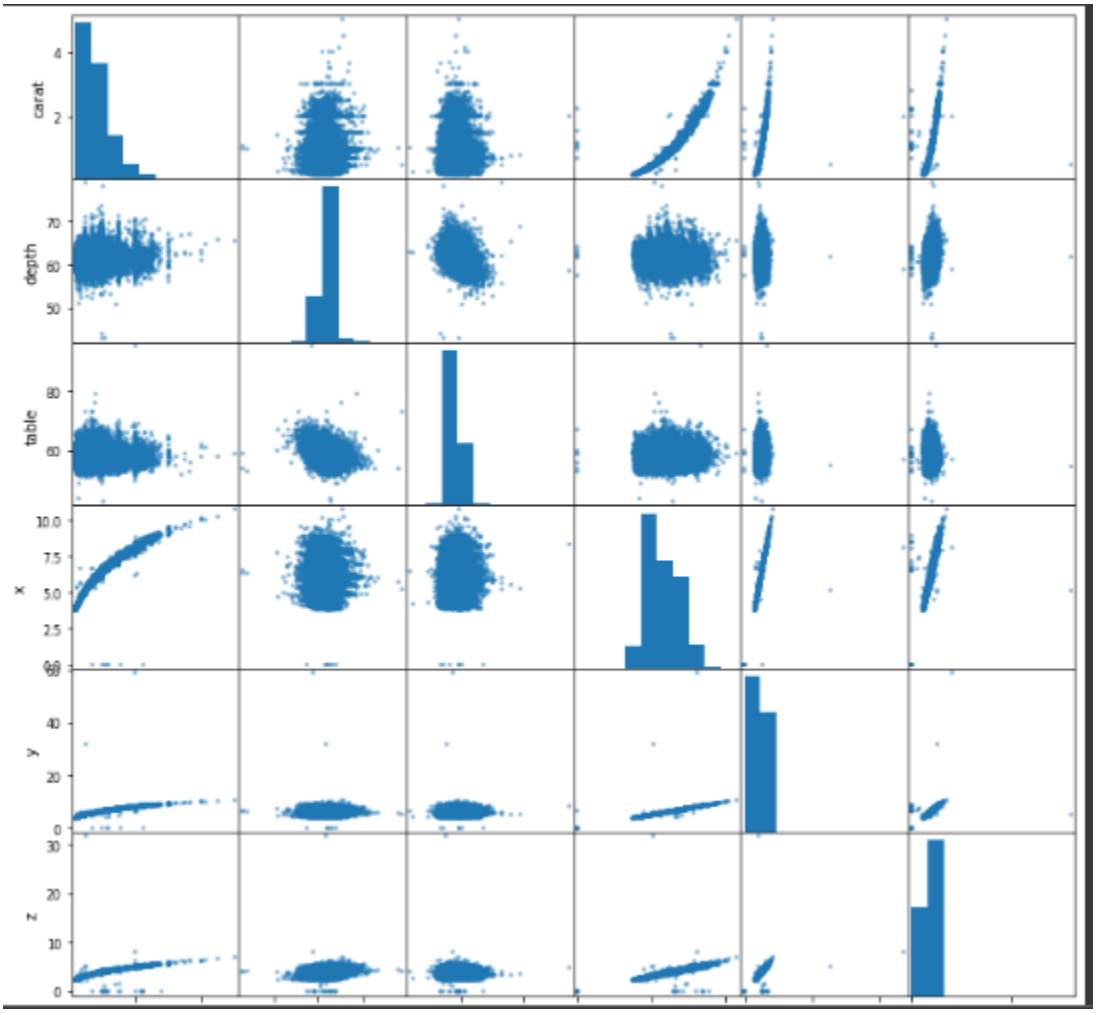
Plot of columns





Plot 3 – Plot of columns

Pair Plot of columns



Plot 4 – Pair plot

Heatmap

Plot 4 – Heatmap

3. Data Cleaning & Pre-processing

Data Cleaning is an important phase in any data science project, if our data is clean then only we can provide it to our machine learning model. Uncleaned Data can further lead our model with low accuracy. And, If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset.

The approach used for identifying and treating missing values and outlier treatment:-

Remove zero value:

```
train = train.drop(train[train["x"]==0].index)
train = train.drop(train[train["y"]==0].index)
train = train.drop(train[train["z"]==0].index)
```

Code Sample 2- Remove zero value

As you can have a look, How we have filled the missing values in a **categorical** variable using **mode**. And, How we have filled the missing values in a **numerical** variable using **Median**. This is how we have an approach for identifying and handling missing values.

While performing Preprocessing and Data cleaning we have to also deal with outliers. Dealing with outliers is also a necessary step to be taken for further analysis and model building. Outliers are data points in a data set that is distant from all other observations. A data point that lies outside the overall distribution of the dataset

```
train = train[(train["depth"] < 75) & (train["depth"] > 45)]
train = train[(train["table"] < 80) & (train["table"] > 40)]
train = train[(train["carat"] <10)]
train = train[(train["x"]<40)]
train = train[(train["y"]<40)]
train = train[(train["z"]<40)&(train["z"]>2)]
```

Code Sample 2- Outlier Detection

After performing Uni-variate, Bi-variate, and Multi-variate analysis we can get an idea about outliers present in our dataset and we solved it.

	carat	depth	table	price	x	y	z
count	53765.000000	53765.000000	53765.000000	53765.000000	53765.000000	53765.000000	53765.000000
mean	0.797449	61.748783	57.457110	3930.714349	5.731383	5.733773	3.539922
std	0.473130	1.419283	2.226326	3985.750608	1.118548	1.116131	0.701568
min	0.200000	50.800000	43.000000	326.000000	3.730000	3.680000	2.060000
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	5324.000000	6.540000	6.540000	4.030000
max	5.010000	73.600000	79.000000	18823.000000	10.740000	31.800000	31.800000

Code Sample 2- Final data

4. Modeling Building

Model Selection and Why?

After cleaning and processing the data then comes the modeling part which includes building Machine Learning models, let's first understand in brief what Machine Learning is?

Machine Learning is a technique that analyzes past data and tries to extract meaningful insights and patterns from them which can be further used to perform predictions in future. For example, classifying whether a tumor is benign or malignant, predicting stock prices, etc. One such application which we're using right here is predicting house prices. Before making predictions first we need to build a model and train it using past data.

First, we need to separate the dataset into two parts: features (property attributes) and labels (prices) which is the required format for any model to be trained on.

Then the data needs to be split into 3 sets

1. Training set - This will be the part of the dataset which the model will be using to train itself, the size should be at least 60-70% of the total data we've.
2. Validation set - This set is used for validating our model's performance for a different set of hyperparameters. After taking out the train set, the remaining set can be split into validation and test set.

3. Testing set - To evaluate how the model is performing on the unseen data on which the model will be doing future predictions on, test set is used. It helps to understand how much error is there between actual and predicted values.

Code Sample 5- Splitting Data

```
X= label_data.drop(["price"],axis =1)
y= label_data["price"]
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.25, random_state=7)
```

```
print(X_train.shape, y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(40323, 9) (40323,)
(13442, 9) (13442,)
```

We need to build different regression algorithms and using the validation set we can determine which model to keep for making final predictions.

Here is the list of all the algorithms we've to build and evaluated:

Code Sample 6- List of Algo used

```
pipeline_lr = Pipeline([("scalar1", StandardScaler()),
                        ("lr", LinearRegression())])

pipeline_lasso = Pipeline([("scalar2", StandardScaler()),
                           ("lasso", Lasso())])

pipeline_dt = Pipeline([("scalar3", StandardScaler()),
                        ("dt", DecisionTreeRegressor())])

pipeline_rf=Pipeline([("scalar4", StandardScaler()),
                      ("rf", RandomForestRegressor())])

pipeline_kn=Pipeline([("scalar5", StandardScaler()),
                      ("kn", KNeighborsRegressor())])

pipeline_xgb=Pipeline([("scalar6", StandardScaler()),
                       ("xgb", XGBRegressor())])

pipeline_gbr=Pipeline([("scalar7", StandardScaler()),
                       ("gbr", GradientBoostingRegressor())])
```

Efforts to improve model performance

Poly nominal features

```
poly = PolynomialFeatures(2)
poly_train = poly.fit_transform(X_train)
poly_test = poly.fit_transform(X_test)
poly_train
```

5. Model Validation

Performance Metrices

Just building is not enough, as we need to evaluate it using different metrics based on the problem we're solving. Model Validation helps us to understand how well the model is generalizing on the real-world data, the data which it has not seen during the training phase.

For regression problems the evaluation metrics we've used are:

- **RMSE** (Root Mean Squared Error)
- **MSE** (Mean Squared Error)
- **MAE** (Mean Absolute Error)
- **MAPE** (Mean Absolute Percentage Error)
- **Adjusted R^2**

The R^2 score is between 0 and 1 (it can also get -ve when the model is performing worse). The closer the value to 1 the better the performance of the model will be and a model which always predicts constant value irrespective of the input values gets an R2 score of 0.

MAE is on average how far those predicted values are from actual values, whereas MSE is the average of squared differences between actual and predicted values.

Let's look at these metrics for the best-performing algorithms on the test set,
Here are the top 10 algorithms based on the test score, MAE, MSE, MAPE, and Adjusted R2. Depending on increasing or decreasing which metric helps solve the business problem, we can pick the appropriate model for final deployment.
Note: Other than Test scores all scores are sorted in ascending order

Table 5 vi- Algorithm Performance

```
LinearRegression -> score: 0.8857851014193359
LinearRegression -> MSE: 1817713.4824095797
LinearRegression -> RMSE: 1348.2260501894998
Lasso -> score: 0.8857630116611238
Lasso -> MSE: 1818065.0376954847
Lasso -> RMSE: 1348.3564208678226
DecisionTree -> score: 0.9999962197780534
DecisionTree -> MSE: 60.161681918508044
DecisionTree -> RMSE: 7.75639619401356
RandomForest -> score: 0.9973756972676765
RandomForest -> MSE: 41765.39591373631
RandomForest -> RMSE: 204.36583842153343
KNeighbors -> score: 0.973432236741276
KNeighbors -> MSE: 422822.08427249955
KNeighbors -> RMSE: 650.2477099325299
[21:24:15] WARNING: /workspace/src/objective/regression
XGBRegressor -> score: 0.9731597579163301
XGBRegressor -> MSE: 427158.5451014283
XGBRegressor -> RMSE: 653.5736722829556
GradientBoostingRegressor -> score: 0.97316346783732
GradientBoostingRegressor -> MSE: 427099.5022490001
GradientBoostingRegressor -> RMSE: 653.5285014817641
```

Recommendation

- The best performance is given by Random forest
- The top key features that drive the price of the property are: 'carat, ‘X’, 'y','z',
- The above data is also reinforced by the analysis done during bivariate analysis.
- Using machine learning algorithms in recommender systems helps buyers find property details and unique insights of it.

Final Words:

- We can conclude from above that poly nominal is giving better results compared to that of tuning done by graphical method of individual parameters or RandomSearchCV
- The ensemble models have performed well compared to that of linear,KNN,SVR models, logistic
- The best performance is given by Random forest model
- The top