DIAMOND PRICE PREDICTION
USING
MACHINE LEARNING

COEN -240 - MACHINE LEARNING
Spring 2022

Professor Dr. Alex, Sumarsono

Sruthi Thiyagarajan, Tony Mathen

Contents

I. Abstract

Diamond is a rare gem found naturally in the form of carbon on earth. The oldest diamonds are believed to crystallize 3.3 billion years ago with the first being discovered in India and then in Brazil. Unlike gold and silver whose prices mostly depend on their weight, the price of a diamond is determined by an array of factors like the cut, clarity, carat, and color known as the 4C's of a diamond. A comparative analysis of regressive machine learning models mainly between trivial and ensemble models like Decision Trees, Random Forest, and XGBoost to find the model that can predict the price of the diamond most accurately is performed. This project is based on the classic Diamonds dataset containing the price and various attributes of almost 54,000 diamonds to train the machine learning models. From the performance parameter metrics and the analysis, the Random Forest algorithm proved to be the most optimal algorithm.

II. Introduction

Diamonds are popular for their visual appeal due to their optimal property where it reflects and scatters light making them sparkle. Other factors contributing to its popularity are its rarity as well as a promotion by jewelers. Being the most rigid substance occurring naturally, it is also used in various machines and equipment for cutting and slicing metals and other strong substances. Thus, they are extensively used in real-world applications like medical equipment, and drilling machines as well in the aerospace sector. Due to their high heat conductivity, diamonds are also employed in several high-end semiconductor products.

Diamonds are very expensive, even a minor change in their properties would cause significant variations in the price. The price of a diamond is different from the time it is mined to the final product that we see in jewelry. At every stage from the mining to polishing and to the storefront, the prices keep increasing until they reach the retail price. Although costs are incurred in mining, cutting, and polishing, there are also other key factors that determine the price of a diamond. These factors are the carat, cut, clarity, color, length, width, depth, and table width. Among these, the most crucial factors are the carat, cut, clarity, and color of the diamond known as the 4C's of a diamond.

Our main motive is to use supervised machine learning techniques for regression to predict the price of diamonds using the Diamonds dataset. We have limited our analysis to three machine learning algorithms namely Decision Trees, Random Forest, and XGBoost. We would be comparing the accuracy of our models using four error metrics viz. R2 score, Adjusted R2 score, Mean Absolute Error, and Root Mean Squared Error.

Tools Used:

We have used Python 3 programming language along with libraries like matplotlib, NumPy, seaborn, and sci-kit learn. We have performed our data analysis and model evaluation using the Google Colab notebook.

III. Dataset & Algorithms

Part 1. Dataset

1. Data Acquisition

Kaggle is an online community of data scientists and machine learning practitioners. It hosts thousands of datasets and is often one of the best data sources for training machine learning models. Users can also train their models on the portal and share them it. We have used the Diamonds dataset from Kaggle to train our machine learning model.

2. Data Description

The Diamonds dataset consists of total of 53,940 unique samples of diamonds consisting the attributes of carat, cut, color, clarity and dimensional attributes x (length), y (width), z (depth), table (width of top of diamond relative to widest point) and depth (total depth percentage). Cut, color and clarity are categorical values whereas carat, x, y, z, table and depth are numerical values.

Price: Target variable, continuous variable.

Cut : Responsible for diamond’s sparkle and thus a poorly cut diamond will be dull even if it has high color and clarity grade. Five types of cut namely fair, good, very good, premium and Ideal with ‘Ideal’ marking the best and ‘Fair’ marking the least desirable.

Clarity: Determines its purity and rarity and is characterized by the absence of inclusions and blemishes in the diamond when seen from a 10-power magnification. Clarity types include Imperfect (I1, I2, I3), Slightly Included (SI1, SI2), Very Slightly Included (VS1, VS2) and Very Very Slightly Included (VVS1, VVS2) with I1 being the lowest grade of clarity and VVS2 being the highest grade.

Color: Color refers to the natural tint of white diamonds. The closer the diamond is to being colorless the rarer it is and thus priced accordingly. Denoted by range D (best) to J (worst).

Carat: Carat is most objective attribute in the 4C’s of a diamond. It refers to the weight of the diamond.

Length, width and height

Table: Flat surface of the diamond when the diamond is viewed from the top

Depth Percentage : Measures diamond’s height from the bottom tip to the table of the diamond and can be computed using the below formula.

$$\text{Total Depth Percentage} = 2 * z / (x + y)$$

Name	Values	Datatype
Price	**($326--$18,823)**	**Numerical**
Carat	**0.2 - 5.01**	**Numerical**
Cut	**(Fair, Good, Very Good, Premium, Ideal)**	**Categorical**
Color	**J (worst) to D (best)**	**Categorical**
Clarity	**(I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))**	**Categorical**
X (length)	**(0 - 10.74) mm**	**Numerical**
Y (Width)	**(0 - 58.9) mm**	**Numerical**
Z (Depth)	**(0 - 31.8) mm**	**Numerical**
Depth (total depth percentage)	**(43 - 79) mm**	**Numerical**
Table	**(43 - 95) mm**	**Numerical**

Table-1 Features of dataset

3. Data Preprocessing

i. Exploratory Data analysis

We identified three categorical variables cut, clarity, and color in the dataset and we plotted bar charts to understand the distribution of these variables. We observe that for every category of the three variables, there are enough rows for our model to learn from (figure -1). If the bars were skewed, the lesser number of data points for any of the categories would make it difficult for the ML algorithm to learn.

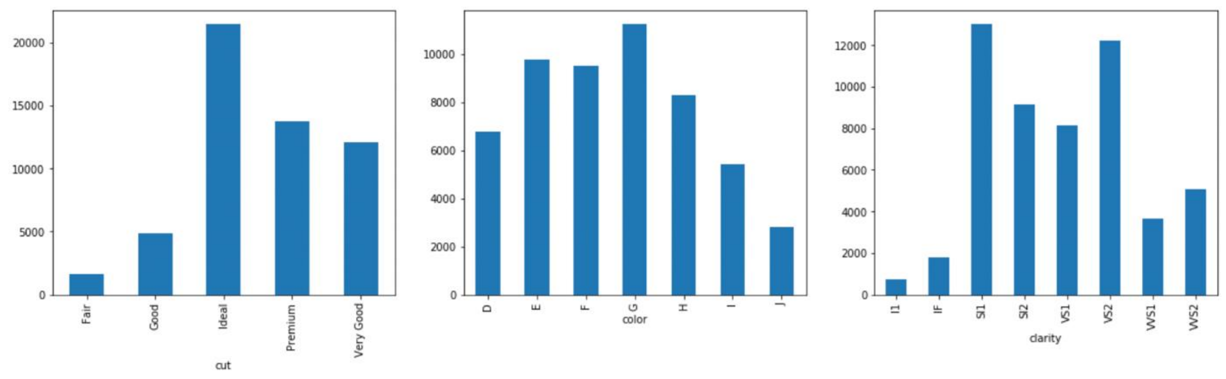


Figure-1 Distribution of Categorical Features

For the continuous variables, we plot the histograms to understand the distribution and ideally, we expect the distribution to be a bell curve or slightly skewed bell curve. We observe carat, depth and table have bell distribution whereas for variables x, y and z have skewed distribution and thus we should remove outliers to make their distributions closer to bell curve.

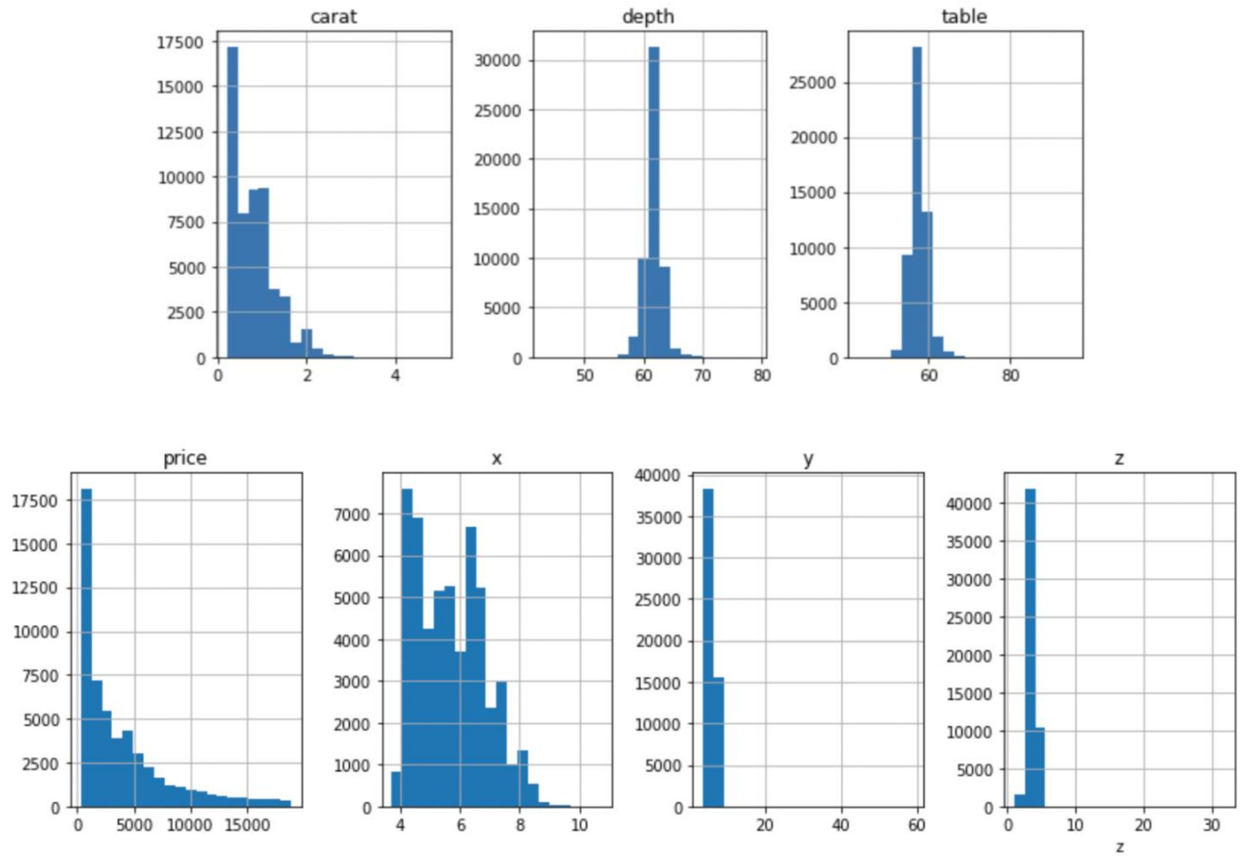
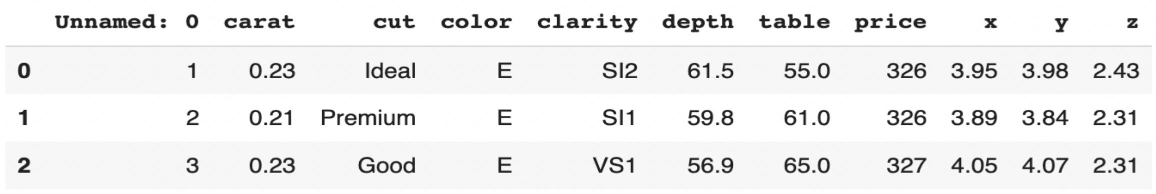


Figure-2 Distribution of Continuous Features

ii. Removal of unwanted data and null

We observe an 'unnamed' column in the dataset that is nothing but an index and doesn't add any meaningful value in our prediction and thus dropping it. We also observed **'0'** values for the dimension columns (x, y, and z) which represent dimensionless diamonds indicating they are faulty points while examining the statistical summary of 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

Figure-3 Unwanted column

iii. Identification and Removal of Outliers

Outliers are data points in the dataset whose values are far away from the rest of the data points. This may occur due to data variability or due to human or experimental error while gathering the data. Outliers can negatively affect our statistical analysis and the training process leading to lower accuracy in our models. Z-score, IQR (Interquartile Range) method, and DBSCAN clustering are some of the many techniques for detecting outliers. We have used the IQR method for removing outliers in our dataset after visualizing the data using boxplots. The technique can be summarized as follows:

1. Calculate the Interquartile Range (IQR) for every column
2. Calculate the constant which is equal to 1.5 * (IQR)
3. Add this constant to the third quartile and remove any data point greater than this upper limit
4. Subtract the constant from the first quartile and remove data points lesser than this lower limit

After examining the boxplots of the features (figure 4), we observe outliers in the dataset and thus proceed to remove them with the IQR technique before we train our model. The boxplots after removing the data are shown in figure 5.

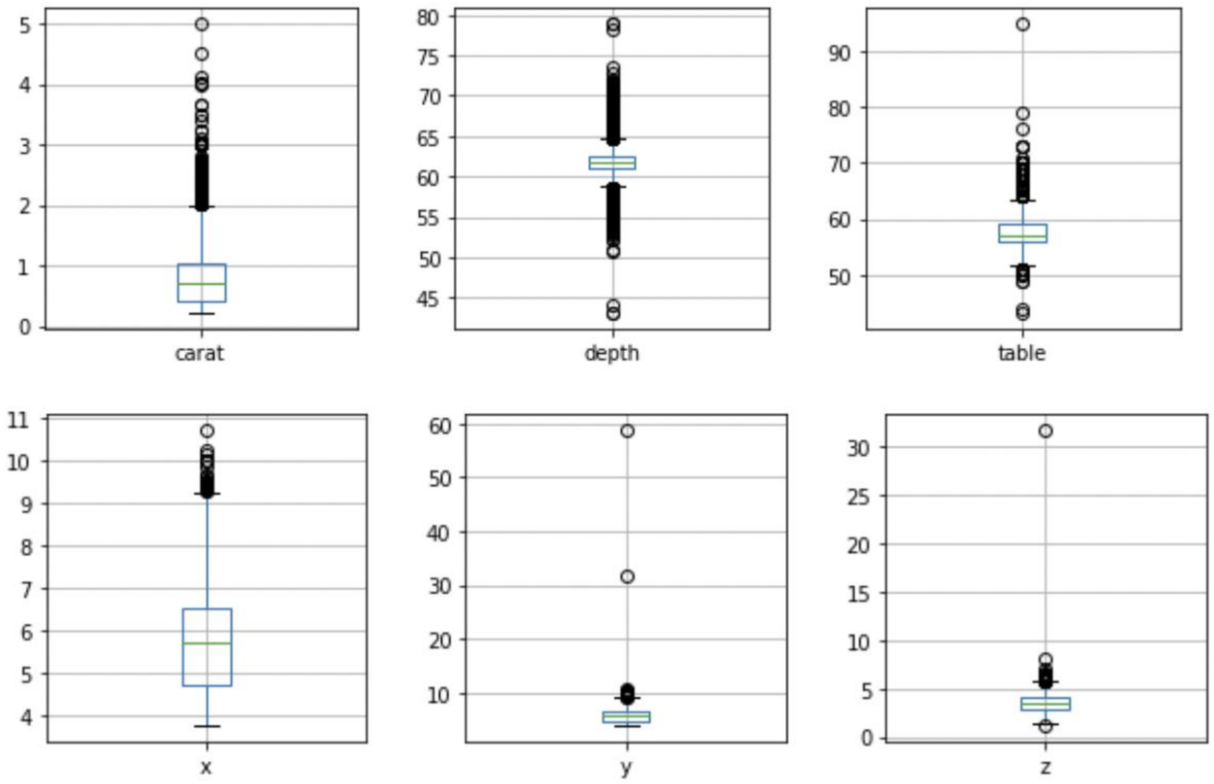


Figure-4 Boxplot before outlier removal

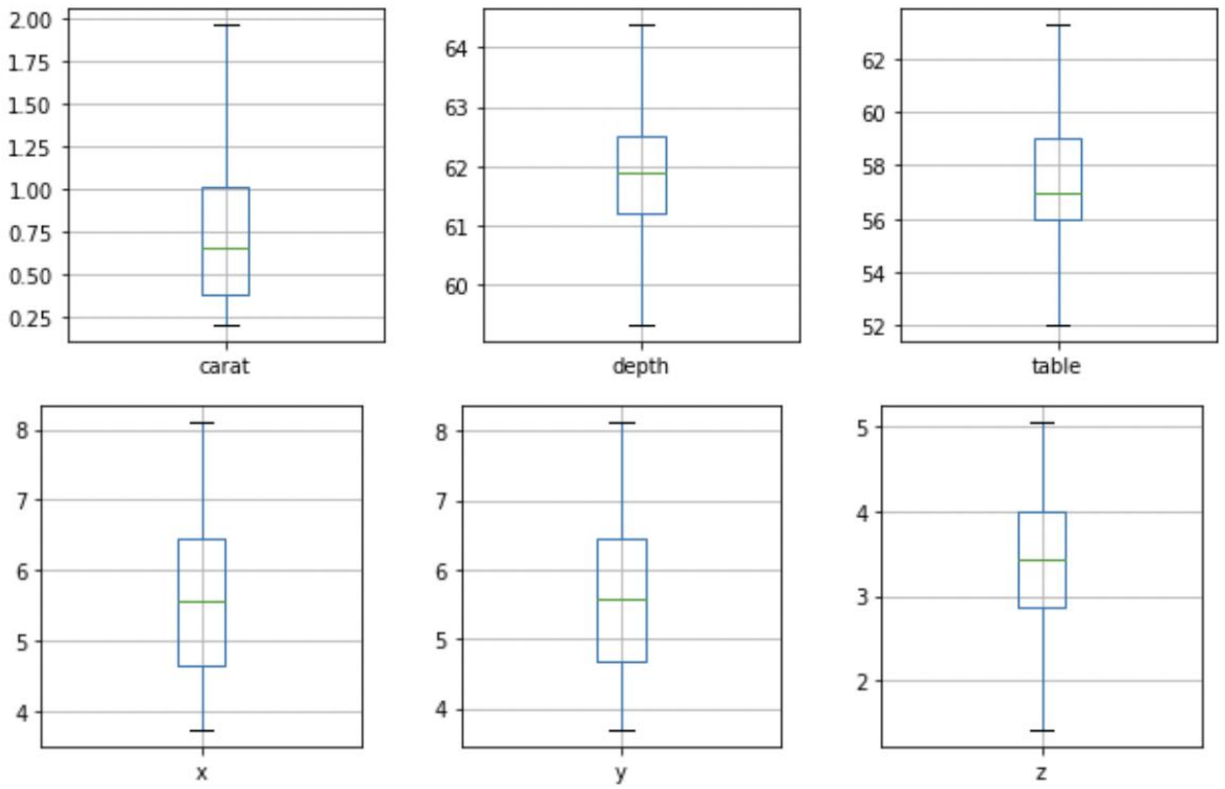
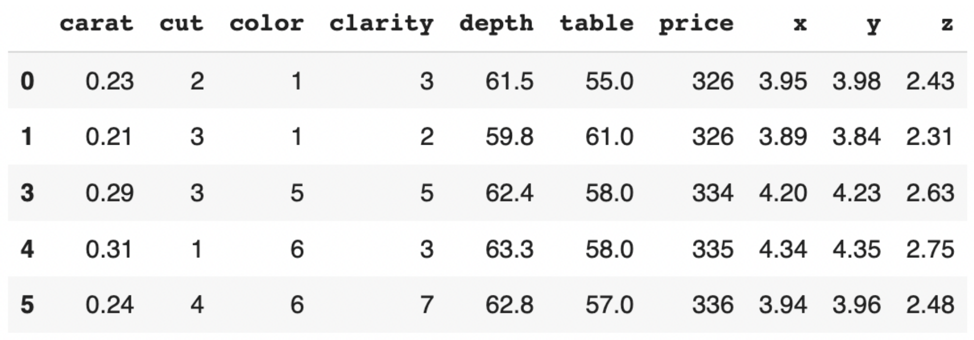


Figure-5 Boxplot after outlier removal

iv. Label Encoding

Label encoding converts label encoded values into numerical values which can be interpreted by machine learning algorithms. Since we have three categorical variables viz. Cut, Clarity, and Color, we have performed label encoding as part of data pre-processing (figure-6)



	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	2	1	3	61.5	55.0	326	3.95	3.98	2.43
1	0.21	3	1	2	59.8	61.0	326	3.89	3.84	2.31
3	0.29	3	5	5	62.4	58.0	334	4.20	4.23	2.63
4	0.31	1	6	3	63.3	58.0	335	4.34	4.35	2.75
5	0.24	4	6	7	62.8	57.0	336	3.94	3.96	2.48

Figure-6 Label Encoding for categorical features

v. Standardization

Data scaling is an important pre-processing step, that helps to improve the performance of our model. We have used the standardization technique for data scaling, and it works by subtracting the mean of the data for every variable and then dividing it by the standard deviation. This shifts the distribution with a mean of zero and standard deviation of one.

vi. Correlation of features

Since carat represents the weight of the diamond, we can expect a high correlation between price and the carat weight of a diamond. The pairplot of these two variables confirms this relationship. However, we also observe that higher carat diamonds have higher price volatility. A correlation matrix is used to find the correlation between different variables. Every cell in the correlation matrix computes the Pearson Correlation Coefficient for the two variables. Variables with high correlation will have their coefficient values closer to 1. Pearson Correlation Coefficient can be computed using the following formula:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

For representing the correlation matrix, we have used a heatmap to find features with high correlation. We observe that the features x, y, z, and carat have a high correlation to the target variable price. Thus, we can readily consider these features for training our model. The features ‘depth’, cut and ‘table’ show low correlation and therefore we can consider dropping them. However, we have decided to keep it as the number of features in the dataset is low.

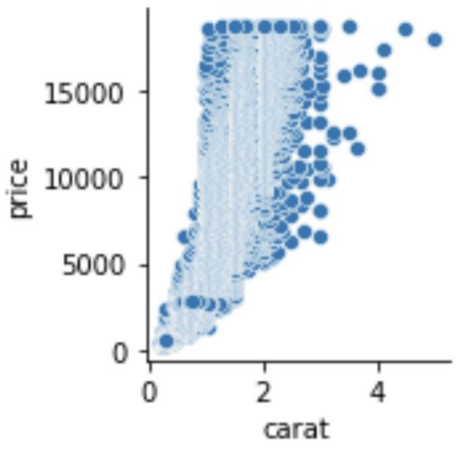
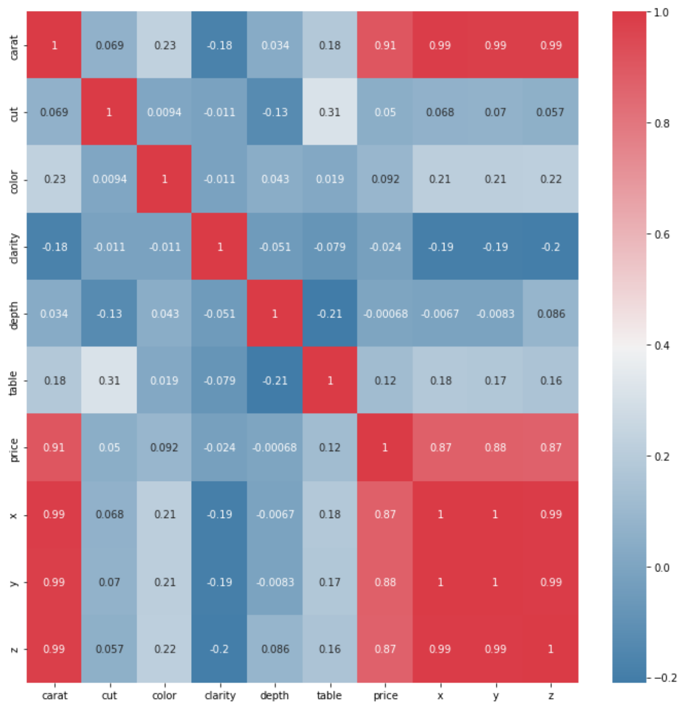


Figure-7 Pairplot for carat/price



	carat	cut	color	clarity	depth	table	price	x	y	z
carat	1	0.069	0.23	-0.18	0.034	0.18	0.91	0.99	0.99	0.99
cut	0.069	1	0.0094	-0.011	-0.13	0.31	0.05	0.068	0.07	0.057
color	0.23	0.0094	1	-0.011	0.043	0.019	0.092	0.21	0.21	0.22
clarity	-0.18	-0.011	-0.011	1	-0.051	-0.079	-0.024	-0.19	-0.19	-0.2
depth	0.034	-0.13	0.043	-0.051	1	-0.21	-0.00068	-0.0067	-0.0083	0.086
table	0.18	0.31	0.019	-0.079	-0.21	1	0.12	0.18	0.17	0.16
price	0.91	0.05	0.092	-0.024	-0.00068	0.12	1	0.87	0.88	0.87
x	0.99	0.068	0.21	-0.19	-0.0067	0.18	0.87	1	1	0.99
y	0.99	0.07	0.21	-0.19	-0.0083	0.17	0.88	1	1	0.99
z	0.99	0.057	0.22	-0.2	0.086	0.16	0.87	0.99	0.99	1

Figure-8 Correlation Matrix

Part 2. Algorithm Implementation

1. Theory - Overview of the Regression Models

i. Decision Tree Regression

This is a non-parametric, supervised machine learning algorithm. Programmatically, the Decision Tree is the “Nested IF ELSE condition”. This algorithm works by dividing the dataset into smaller subsets and corresponding tree building. The geometric intuition and DT in a nutshell is the “Axis-parallel” hyperplane for each decision node. Axis-parallel means the decision surface is parallel to any of the axes. Figure-1 depicts the terms used in decision trees.

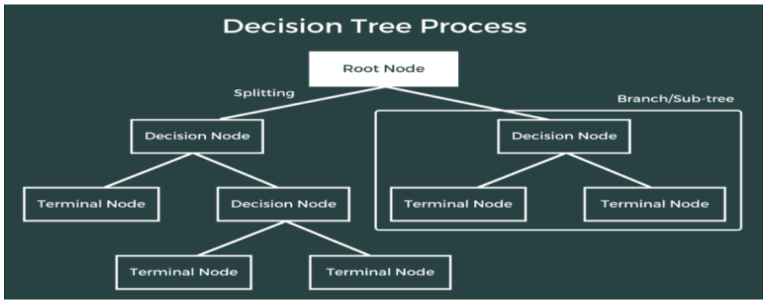


Figure-9 Decision Trees

Decision Tree Algorithm:
In sci-kit learn, range of algorithms from ID3 to CART(Classification and Regression Trees) is being used. The basic algorithm is the ID3(by Quinlan).

How does a Decision Tree is constructed for Regression?

Recursively, the tree is constructed by dividing the training samples based on the features of the dataset. The metric “**Residual or Mean Squared Error**” is evaluated for the construction.
Categorical/Discrete feature – All values are evaluated using the metric.
Continuous feature – Mean of two consecutives (ascending order) of the training data are calculated for thresholds.
Hence, each node has – features list each with different thresholds and calculated metric. The feature that gives the best metric value (based on the metric selected) is used.
For a regression tree, the prediction at the end is the mean of the values for the target variable at that leaf node.

Advantages	**Drawbacks**
Easy to interpret	High variance with large depth and high bias with shallow depth. (Overfitting and Underfitting)
Tolerant to missing values	Weak Learners
Require less training data than other ML models	

ii. Ensembling

Ensemble methods create multiple individual models and aggregates them to produce improved results. They aim at reducing bias and variance which in turn increases the generalization and accuracy of the models. The individual models are called base learners. Figure-1 represents the types of ensemble methods.

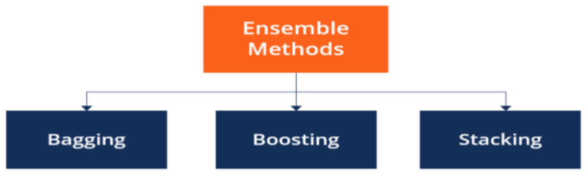


Figure-10 Ensemble Methods

Parallel Ensemble Technique:
Base learners are generated parallelly increasing the independence between the base learners.
Sequential Ensemble Technique:
Base learners are generated in a sequence creating a dependence. Performance of the model is increased by penalizing the misinterpreted learners.

a. Random Forest - Bagging: (Bootstrap +Aggregation)

Random Forest algorithm is a popular Bagging Technique. It mainly reduces the high variance faced by the base model (Decision Tree) resulting in greater accuracy and reducing overfitting.
Bootstrap: Sampling technique with replacement on the training data.
Aggregation: Combining the outcomes of sub models to make accurate prediction. In regression, it is the mean of the outcome models.
RF = RS + CS + Aggregation
where, RF – Random Forest, RS – Row Sampling, CS – Column/Feature Sampling

Hyperparameters:
They increase the performance, either in terms of accurate predictions or processing speed.

Performance related hyperparameters:
- **n_estimators** – number of trees the algorithm makes before averaging the predictions.
- ***max_features*** – maximum number of features random forest considers splitting a node.
- ***mini_sample_leaf*** – determines the minimum number of leaves required to split an internal node.

Fastness related hyperparameters:
- ***n_jobs***– Tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is -1 there is no limit.
- ***random_state***– controls randomness of the sample. Initializing it will produce the same result with same parameters and training data.
- ***oob_score*** – *OOB* means out of the bag. Cross-validation method.

b. XGBoost (Extreme Gradient Boosting) - Boosting:
Boosting technique learns from previous predictor mistakes. It works by arranging weak learners in a sequence, such that weak learners learn from the next learner in the sequence to create better

predictive models. **XGBoost** is one of the most efficient open-source implementation of the gradient boosting(GB) algorithm with some regularization parameter. Models with high bias and low variance can be enhanced with this method. Base learner is Decision Tree with shallow depth. Figure- depicts the features of XGBoost.

GBDTs iteratively train an ensemble of shallow decision trees, with each iteration using the error residuals of the previous model to fit the next model. The final prediction is a weighted sum of all of the tree predictions.
How does XGBoost work for regression?

In this project, the aim is to predict the price of the diamond (continuous variable) given the independent features. Let's understand workflow of XGBoost in this case.

- **Step 1:** Initial Prediction and calculate residuals
  This prediction can be anything. Let's consider the prediction to be the mean value of the price of the diamonds with a value M, mean. Calculating residuals
  Residuals is given by the form: Residuals = Observed values – Predicted values(M)
- **Step 2:** Building of XGBoost Tree
  Each tree starts with a single node containing all the residuals. A score called "Similarity Score" is computed for this node.

  $$Similarity\ Score = \frac{(Sum\ of\ Residuals)^2}{Number\ of\ Residuals + \lambda}$$

  λ (lambda) is a regularization parameter. The default value of λ is 1.
- **Step 3:** Splitting of the residuals node based on the thresholds of the predictors. Calculating the gain based on the similarity helps to decide whether to split or not with this predictor and threshold.

  $$Gain = Left_{Similarity} + Right_{Similarity} - Root_{Similarity}$$

  Gain for continuous and discrete variables is calculated differently. Same technique is recursively applied for the splits and finally for the leaf nodes, the output
  values are given by the formula.

  $$Output\ Value = \frac{Sum\ of\ Residuals}{Number\ of\ Residuals + \lambda}$$

c. Stacking:

Stacking is referred to as stacked generalization. This allows a training algorithm to ensemble several other similar learning algorithm predictions. It can used in regression, density estimations, distance learning, and classifications. It can also be used to measure the error rate involved during bagging.

2. Model Building

The main aim of this project is to analyze the regression models and choose the best for predicting the price of the diamond. We are using three models, DecisionTreeRegressor, RandomForest(Bagging) and XGBoost(Boosting).

The data is preprocessed comprising of outlier detection and removal, encoding of the categorical variables and standardization.

Steps involved in Model Building

- Features and Target Split
- Create instances for three different regressors.
- Fit all the models on training data
- Get the mean of cross-validation on the training set for all the models for root mean square error and R2
- Pick the model with the best cross-validation score
- Hyperparameter tuning of the best selected model for further accurate prediction
- Fit the best model on the test set and get the predicted values for performance evaluation

Let us go through the steps of implementing the models, finding the best fit model, predicting the values and performance evaluation.

I. Splitting the dataset into the Training set and Test set

Using sklearn train_test_split, the dataset is split into training and test data. For this, we use `test_size=0.25,` meaning 25% of the data is used for testing and remaining 75% is used for training.

```
# Assigning the featurs as X and trarget as y
y= df["price"]
X= df.drop(["price"],axis =1)
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.25, random_state=7)
```

II. Creating instance of the model with default configurations

```
dt  = DecisionTreeRegressor(random_state=0)
rf  = RandomForestRegressor(random_state = 0)
xgb = XGBRegressor(random_state =0,verbosity = 0, silent = True)
```

Default configurations:
class `sklearn.tree.`**`DecisionTreeRegressor`**(**, criterion='squared_error', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, ccp_alpha=0.0*)

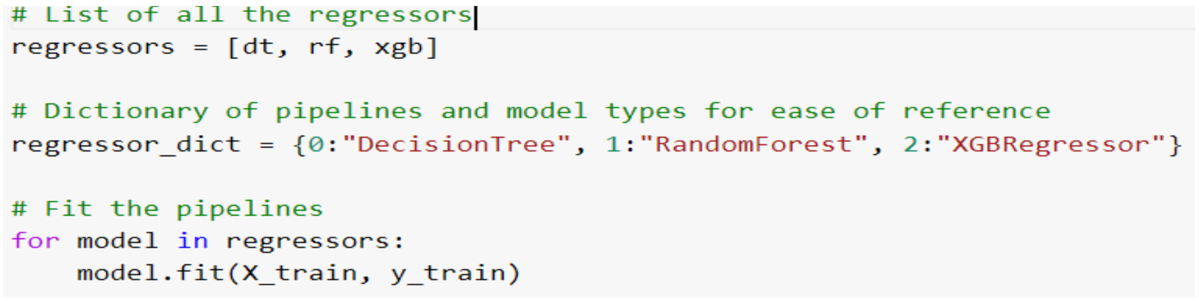
class `sklearn.ensemble.`**`RandomForestRegressor`**(*n_estimators=100*, ***, *criterion='squared_error'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=1.0*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *ccp_alpha=0.0*, *max_samples=None*)

```
from xgboost import XGBRegressor
```

(Open source library)

III. Fitting the models on the training dataset

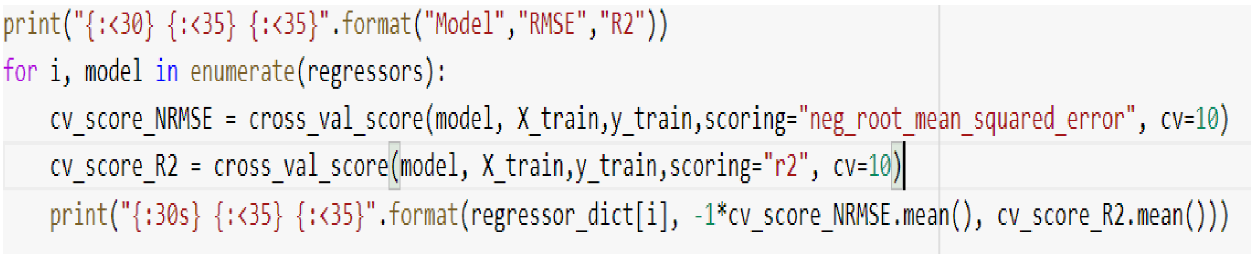
The sklearn provides a fit method for each ML algorithm that takes the responsibility of training the model with the given parameters. In Sklearn, fit() is used to find the attributes of the training set, including the mean, variance, maximum and minimum values, and so on.



```
# List of all the regressors
regressors = [dt, rf, xgb]

# Dictionary of pipelines and model types for ease of reference
regressor_dict = {0:"DecisionTree", 1:"RandomForest", 2:"XGBRegressor"}

# Fit the pipelines
for model in regressors:
    model.fit(X_train, y_train)
```

IV. Cross-validation of all models on the training data with scoring parameter RMSE and R2 score

Cross validation is essential to reduce overfitting and to increase the generalization. In CV, the training set is divided into training and validation data. In K-Fold CV, the training set is divided into k subsets , each time k-1 sets are used for training and kth set is used for validation.



```
print("{:<30} {:<35} {:<35}".format("Model","RMSE","R2"))
for i, model in enumerate(regressors):
    cv_score_NRMSE = cross_val_score(model, X_train,y_train,scoring="neg_root_mean_squared_error", cv=10)
    cv_score_R2 = cross_val_score(model, X_train,y_train,scoring="r2", cv=10)
    print("{:30s} {:<35} {:<35}".format(regressor_dict[i], -1*cv_score_NRMSE.mean(), cv_score_R2.mean()))
```

The result of the CV:

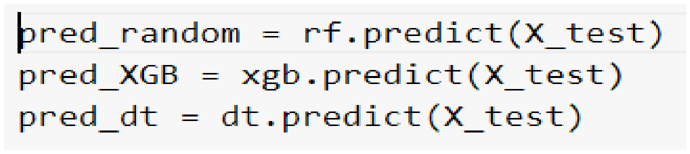
Model	RMSE	R2
Decision Tree	591.801095849941	0.968985726937316
RandomForest	429.3626180281159	0.98367031335571
XGBRegressor	530.140463448892	0.9751033112283014

Table-2 Performance metrics of Cross-validation

From this we can observe that, RandomForest has performed better with considerably lesser RMSE score and R2 score than other regression models. Predicting the results with all three models to evaluate the performance.

V. Predict using different models

We are using sklearn's predict function of each model to predict the values of the testing data.



```
pred_random = rf.predict(X_test)
pred_XGB = xgb.predict(X_test)
pred_dt = dt.predict(X_test)
```

Part 3. Evaluation, Experiment & Result

1. Theory – Performance Metrics

The most common evaluation metric for regression is the RMSE (Root Mean Squared Error). Evaluation is done to optimize the performance. We evaluate the performance of the model on the unseen dataset with also other metrics like R2, Adjusted R2, MAE (Mean Absolute Error) and RMSE, MSE. We characterize a regression model to be good and efficient if the difference between the actual and the predicted values is small, and unbiased for the train, validation and test sets.

i. Mean Absolute Error

It is the average of the absolute difference between actual and predicted values. Interpretation of the MAE is that the model is predicting "MAE" more or less on average than the actual value. The lesser the value of MAE, the better the model prediction.
Pros: Robust to outliers, same unit as the dependent variable.
Cons: Graph of MAE is not differentiable.

$$MAE = \frac{1}{N}\sum \left|Y - \hat{Y}\right|$$

ii. Mean Squared Error

Mean squared error is the squared difference between actual and predicted value. We perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.
Pros: Smoothly differentiable
Cons: Output of MSE is the squared unit of output of the dependent variable, sensitive to outliers.

$$MSE = \frac{1}{N}\sum (y - \hat{y})^2$$

iii. Root Mean Squared Error

RMSE is the square root of MSE. RMSE is measured by taking the square root of the average of the squared difference between the predicted and the actual values. RMSE is a better performance metric as it squares the errors before taking the averages hence large errors receive higher punishment.

Pros: Smoothly differentiable, same unit as the dependent variable.
Cons: Sensitive to outliers.
N= Total number of samples

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

iv. R Squared (R2)

R2 score tells about the performance of the model and how good is the model for given dataset. in terms of wells. It is also called as Goodness of fit. The value of R2 lies between 0 and 1. The higher the value the better the model. If R2 is 0, it represents the model is a simple mean model and negative R2 indicates that the model is much simpler than mean model.
SSR = Sum Square of Residuals(the squared difference between the predicted and the actual value)
SST = Sum Square of Total (the squared difference between the actual and average value)

$$R^2 = \frac{SSR}{SST} = \frac{\sum\left(\widehat{y_i} - \underline{y}\right)^2}{\sum\left(y_i - \underline{y}\right)^2}$$

v. Adjusted R2

The disadvantage of the R2 score is while adding new features in data the R2 score starts increasing or remains constant but it never decreases because it assumes that while adding more data variance of data increases.
The problem arises when adding an irrelevant feature to the dataset, R2 sometimes starts increasing which is incorrect. Adjusted R2 solves this by taking into consideration the number of features.
n = number of observations
k = number of independent variables
R = R2

$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1}\right) \times (1-R)^2\right]$$

The formula can be interpreted as, as K increases by adding some features so the denominator will decrease, n-1 will remain constant. R2 score will remain constant or will increase slightly so the complete answer will increase and when we subtract this from one then the resultant score will decrease. This is the case when we add an irrelevant feature in the dataset.

And if we add a relevant feature then the R2 score will increase and 1-R2 will decrease heavily and the denominator will also decrease so the complete term decreases, and on subtracting from one the score increases.

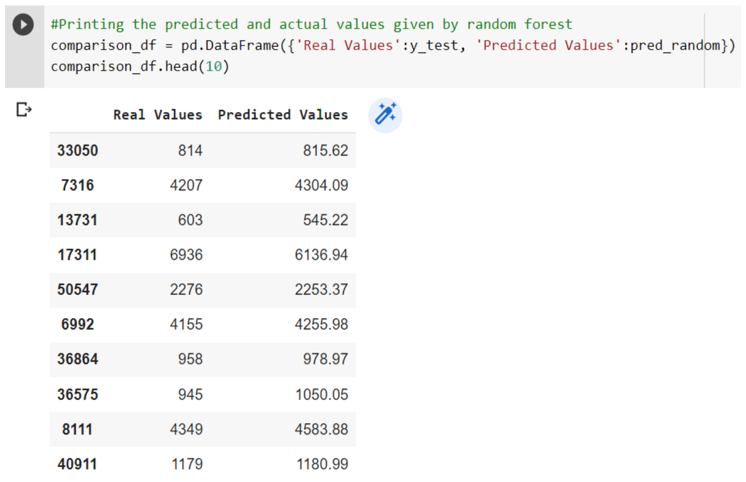
2. Experiment Result and Analysis

On giving the test data to all the models using the predict function as seen in above section, we found that Random Forest Regressor gave good results in terms of all metrics. Table-3 depicts the performance metrics obtained for all three models.

MODEL	MSE	RMSE	MAE	Adjusted R2	R2
Random Forest	182645.150	427.370	220.417	0.98377	0.98379
XGBoost	307828.105	554.822	304.621	0.97351	0.97353
Decision Trees	347931.789	589.857	299.539	0.97006	0.97008

Table-3 Performance metrics of all three models

From the above table, it is evident that Random Forest outperforms the other models in all metrics. From the theoretical understanding of the metric, it can be said that the predicted values of the diamond are "$220" [MAE of Random Forest] on average more or less than the actual value. Figure-4 represents the actual and predicted values by Random Forest for the price continuous variable.



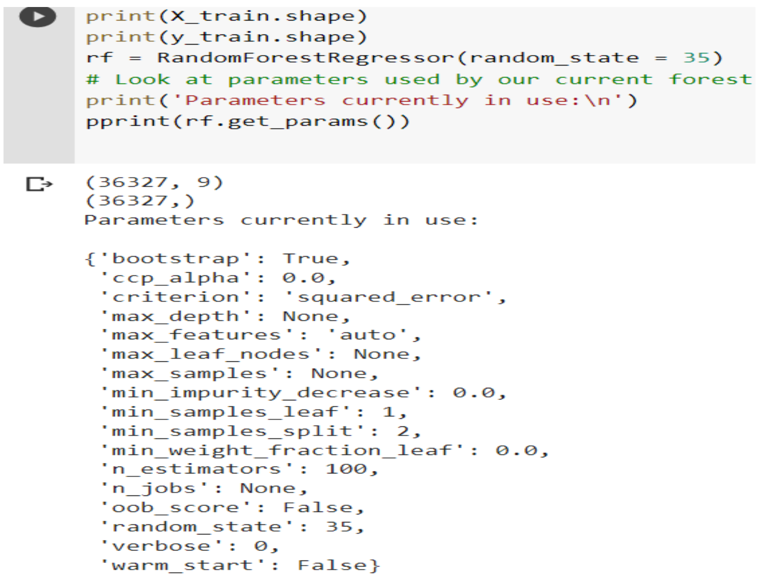
```
#Printing the predicted and actual values given by random forest
comparison_df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':pred_random})
comparison_df.head(10)
```

	Real Values	Predicted Values
33050	814	815.62
7316	4207	4304.09
13731	603	545.22
17311	6936	6136.94
50547	2276	2253.37
6992	4155	4255.98
36864	958	978.97
36575	945	1050.05
8111	4349	4583.88
40911	1179	1180.99

Figure-11 Comparison of Actual and Predicted values by RandomForest

From the observations, we can see that though some of the predictions are not most accurate, some are correctly predicted. The prediction can be improved by using more robust outlier detector and removal and hyperparameter tuning. Figure-5 shows the current configuration of parameters with which the above performance measurement was carried out. We can from the figure that, 36,327 samples were used for training.

In the section of Overview of Regression models, the hyperparameter tuning for RandomForest is discussed. Taking into consideration, the theoretical concept, we carried out RandomSearchCV to determine the hyperparameters from a range of values.



```
print(X_train.shape)
print(y_train.shape)
rf = RandomForestRegressor(random_state = 35)
# Look at parameters used by our current forest
print('Parameters currently in use:\n')
pprint(rf.get_params())
```

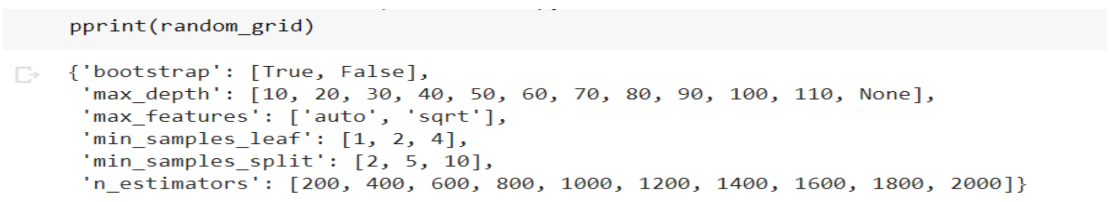
```
(36327, 9)
(36327,)
Parameters currently in use:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 35,
 'verbose': 0,
 'warm_start': False}
```

Figure-12 Default configuration of Random Forest

a. Hyperparameter Tuning:

Random Forest Hyperparameter Tuning Considerations:
n_estimators = number of trees in the forest
max_features = max number of features considered for splitting a node
max_depth = max number of levels in each decision tree
min_samples_split = min number of data points placed in a node before the node is split
min_samples_leaf = min number of data points allowed in a leaf node
bootstrap = method for sampling data points (with or without replacement)

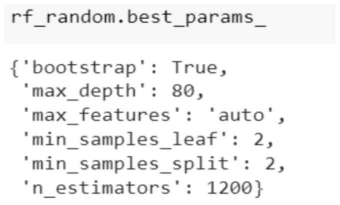
Creating a grid to select the appropriate hyperparameters. On each iteration, the algorithm will choose a different combination of the features. Hence, there are 2 * 12 * 2 * 3 * 3 * 10 = 4320 settings/combinations. Figure-13 shows the creation of the random grid to select the best hyperparameter. It is followed by the random search training to obtain the best hyperparameters. Figure-14 represents the best parameters.



```
pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```

Figure-13 Random grid to determine hyperparameters



```
rf_random.best_params_

{'bootstrap': True,
 'max_depth': 80,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 1200}
```

Figure-14 Best Parameters for RandomForest

The hyperparameter tuning has resulted in slight improvement in terms of the performance. Table-4 depicts the difference in metrics between the Default and Hyperparameter Tuned RF.

Random Forest	**Hyperparameter tuned Result**	**Default Parameter Result**
R^2	0.98397	0.98379
Adjusted R^2	0.98396	0.98377
MAE	219.128	220.417
MSE	180577.957	182645.150
RMSE	424.9446	427.370

Table-4 Results of Default and Hyperparameter Tuned Random Forest

Figure -15 and Figure- 16 shows the graphical representation of the predicted and actual values of the prices before and after hyperparameter tuning. We can observe small improvement between the two graphical representations. Figure-17 and Figure-18 shows the graphical representation of XGBoost and Decision Trees.

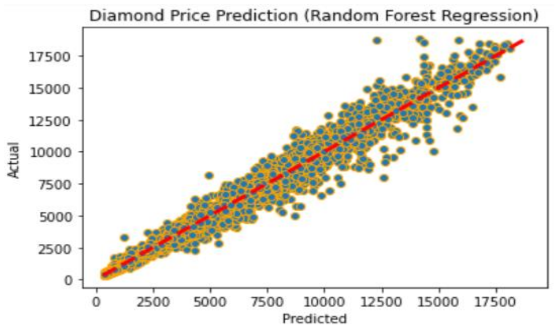


Figure-15 RF Default Parameters

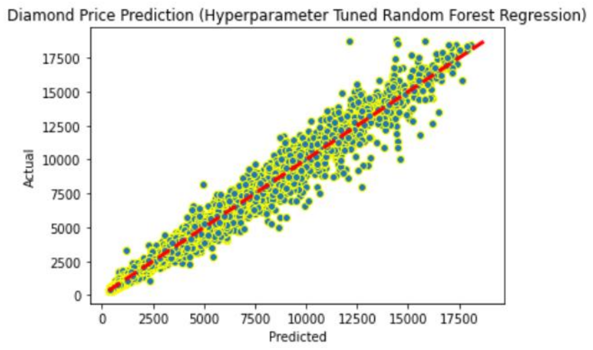


Figure-16 Hyperparameter Tuned

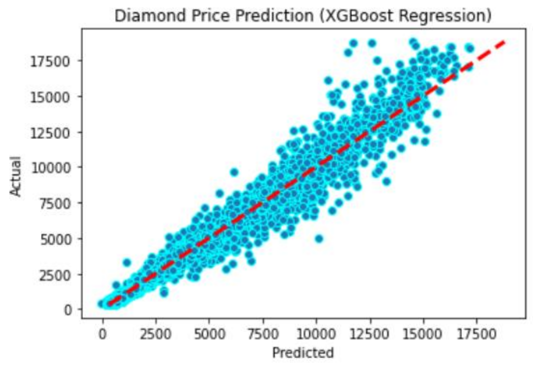


Figure-17 XGBoost Default Parameters

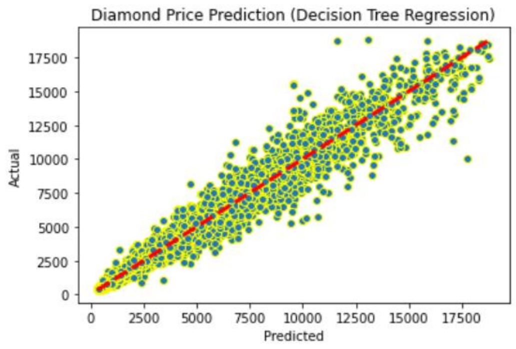


Figure-18 DT Default Parameters

IV. CONCLUSION

Our analysis and experiments conclude that among supervised learning algorithms like Decision Tress, Random Forests, and XGBoost , Random Forest algorithm performed better and is the most suitable algorithm for diamond price prediction concerning all the performance metrics. Different techniques of outlier removal and more significant hyperparameter tuning could further help in the accurate price prediction of diamond. Unsupervised techniques to predict the price of diamond with more accuracy and robustness is one area to explore. This could be taken into consideration for future study.

V. REFERENCES

[1] Ensemble Models - A guide to learning ensemble techniques in a simple walkthrough https://towardsdatascience.com/ensemble-models-5a62d4f4cb0c

[2] A. C. Pandey, S. Misra, and M. Saxena, "Gold and diamond priceprediction using enhanced ensemble learning," in *2019 Twelfth International Conference on Contemporary Computing (IC3)*. IEEE, 2019, pp. 1–4.

[3] S. Agrawal, "Diamonds dataset", May 25, 2017, Kaggle. Kaggledatasets repository. [Online]. Available: https://www.kaggle.com/shivam2503/diamonds Accessed on: May 31, 2022

[4] G. Sharma, V. Tripathi, M. Mahajan, and A. K. Srivastava,"Comparative analysis of supervised models for diamond price prediction," in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2021, pp. 1019–1022.

[5] A. Di Bella, L. Fortuna, S. Grazianil, G. Napoli2, M.G. Xibilia, "A Comparative Analysis of the Influence of Methods for Outliers Detection on the Performance of Data Driven Models", in *2007 Technology Conference, Instrumentation and Measurement*. IEEE 2007

[6] Sci-kit learn for Machine Learning – Repository of Machine Learning Algorithms - https://scikit-learn.org/stable/

[7] Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurelien Geron

[8] Ensemble methods: bagging, boosting and stacking - https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205