Dimond Price Predictor

The aim of this project is to predict the price of diamonds based on some input values using different regression models, following a Kaggle competition.

**Contents**

[**1-Introduction** 1](#_Toc134755791)

[**2-Proposed algorithms** 1](#_Toc134755792)

[**2-1 Linear Regression** 1](#_Toc134755793)

[**2-2 Lasso Regression** 2](#_Toc134755794)

[**2-3 Decision Tree** 2](#_Toc134755795)

[**2-4 Random Forest** 2](#_Toc134755796)

[**2-5 XGBoost Regressor** 2](#_Toc134755797)

[**2-6 Voting Regressor** 2](#_Toc134755798)

[**3-Implementaion** 2](#_Toc134755799)

[**4-Performance measure** 7](#_Toc134755800)

[**5-Experimental Results** 8](#_Toc134755801)

[**6-Summary** 8](#_Toc134755802)

# **1-Introduction**

This project uses the data set provided by Kaggle in which the contestants were given a train/test data.

# **2-Proposed algorithms**

In this project, different regression models were tested:

Linear Regression, Decision Tree, Random Forest, XGB Regressor, Lasso Regressor.

## **2-1 Linear Regression**

It is one of the simplest models for doing regression tasks, it is a linear model that predicts an output by computed a weighted sum of input features and adding a constant called *bias.*

The equation that describes linear regression is:



Here we have:

is the predicted value.

is the bias.

are the model parameters including bias.

is the number of features.

are the feature values.

## **2-2 Lasso Regression**

It is a powerful statistical tool that is widely used in data analysis for its ability to improve the accuracy of predictions. It is a form of linear regression that uses a technique called shrinkage to bring data values closer to a central point, such as the mean. This approach encourages the development of simpler, more straightforward models that contain fewer parameters. Lasso regression is particularly well-suited for datasets that exhibit high levels of multicollinearity, which can make it challenging to identify the underlying relationships between different variables. Additionally, the use of lasso regression can be a great way to automate aspects of model selection, such as variable selection and parameter elimination. By leveraging L1 regularization technique, Lasso regression can effectively perform feature selection, making it an indispensable tool for working with complex datasets containing numerous features

## **2-3 Decision Tree**

 They are a powerful, yet simple, supervised learning method that uses a tree-like model of decisions and their potential outcomes. They are used in both classification and regression problems. Unlike highly complex SVMs, decision trees provide excellent visualization of the decision-making process and feature importance. The goal is to create a model that predicts the value of a target variable by learning simple decision rules from the data features. A tree can be viewed as an approximation of a piecewise constant function.

## **2-4 Random Forest**

is a widely-used supervised machine learning algorithm used to solve nonlinear problems. It is commonly used in both classification and regression problems. The method builds decision trees on different samples and takes their majority vote for classification or average for regression.

## **2-5 XGBoost Regressor**

It is a powerful machine learning library that enables scalable and distributed gradient-boosted decision tree algorithms, making it a leading choice for regression, classification, and ranking problems. To fully understand XGBoost, it is essential to comprehend the underlying concepts and algorithms of supervised machine learning, decision trees, ensemble learning, and gradient boosting. XGBoost is highly favored in Kaggle competitions due to its ease of use, efficiency, accuracy, and feasibility. It boasts an easy installation process and a highly developed R/Python interface for users. Additionally, XGBoost offers automatic parallel computation on a single machine and can be run on a cluster, making it a highly versatile tool. It is also highly customizable, with the ability to set objectives and evaluation metrics and fine-tune parameters.

## **2-6 Voting Regressor**

A voting regressor is an ensemble technique that fits several base regressors, each on the whole dataset. Then it averages the individual predictions to form a final prediction.

# **3-Implementaion**

The steps for implementing this project were:

1. **Loading and investing data set:**

The data set contains train data that will be used during training and validation, and test data that will be used for making the final predictions and saving results.

The train data contains 43152 rows and 11 columns.

The columns are:

* Id.
* Carat: or size of the diamond.
* Cut.
* Color.
* Clarity.
* Depth.
* Table.
* x: length of the diamond.
* y: width of the diamond.
* z: height of the diamond.
* Price.

Test data contains 10788 rows and 10 columns:

The columns are the same as the train data except for the price column that we aim to find its values.

1. **Analyzing data:**

the results we got after analyzing the train data using pandas and NumPy:

* The train data has three categorical columns which are ‘cut’, ‘color’ and ‘clarity columns and they are all ordered.
* The x, y and z columns have the min value zero which indicates fault values, and they have their max value differs from the 75% value which indicates outliers.
* There is a strong positive relation between carat, x, y and z of the diamond and its price.
* There are 5 zero values in x column, 4 zero values in x column and 17 zero values in z column.

1. **Visualizing data:**

For visualizing data, we used boxplots to detect outliers and their values, scatter plot to investigate the relation between variables and heatmap for features’ correlation.

Some insights we got after visualizing the train data using pandas, seaborn and matplotlib:

* Outlier values in x, y and z columns are less than 30.
* Outlier values in depth column are in range ]45, 75[
* Outlier values in table column are in range ]40, 80[

1. **Preprocessing:**

In order to make data more efficient, we:

* Removed zero values from z, y and z columns.
* Removed outliers from x, y, z, depth and table columns.
* Column transformation:

For numerical columns we used:

* **Simple Imputer(strategy=median)** so any missed nan value will be replaced by the median value of its column.
* **CombinedAttributeAdder:** to add the volume column which is x\*y\*z values.
* **StandardScaler:** to scale the data for better training.

For categorical columns we used:

* **OrdenalEncoder**: to convert categorical values in the ‘cut’, ’color’ and ‘clarity’ columns into numerical values which will take values like 1, 2, 3 …
* **StabdardScaler.**

Combine all of these transformations in a single pipeline.

The output of this stage is:

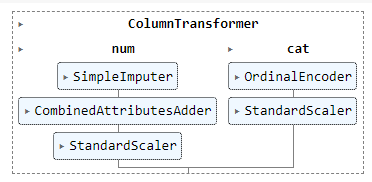


Figure 1 full pipeline

1. **Select and train models:**

for this project we choose 5 regression models.

For the training part, we first define the model with the transformation that will be applied to the data, then we fit the model with training data.

The result of training each model:

linear regression:

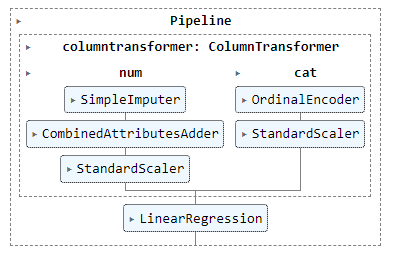


Figure 2 fit the linear regression model

lasso regression with max iterations=10000:

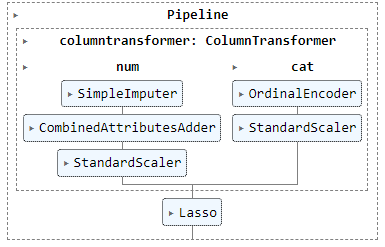


Figure 3 fit the lasso regression model

decision tree:

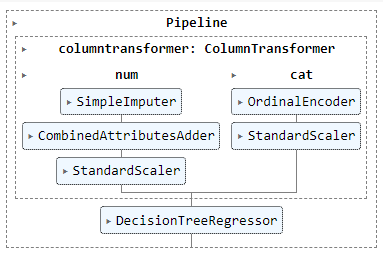


Figure 4 fit the decision tree model

random forest with random state 42 and number of estimators=50:

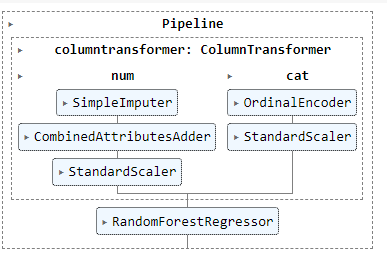


Figure 5 fit the random forest model

XGBoost regression with 300 estimators and learning rate equals 0.1:

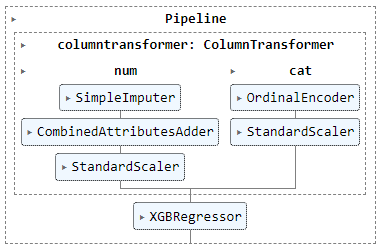


Figure 7 fit the XGBoost regressor model

Voting regressor composed of two models:

1. Random forest.
2. XGBoost regressor.

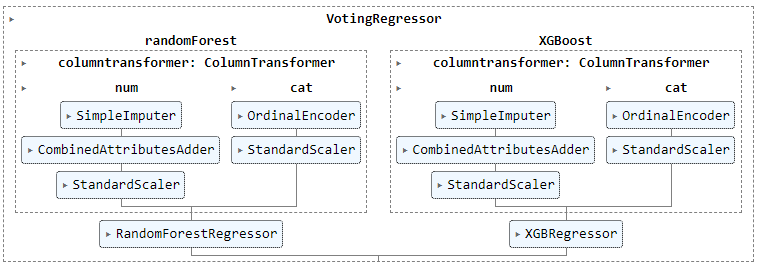


Figure 8 fit the voting- regressor model

1. **Evaluating using cross validation.**
2. **Saving the results:**

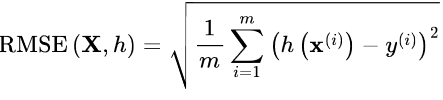
In this step predictions on the test set were performed using the chosen model, then saved the result as an csv file.

# **4-Performance measure**

For evaluating the model, RMSE, mean and standard deviation were calculated using cross validation method.

The **RMSE** metric is the square root of the mean square error.

The equation is:



The cross-validation technique is used for evaluating a model performance, it works by dividing the training data into k number of subsets, fitting the model with the training subset and evaluating the model using the test subset (which is the remaining data samples of the data after obtaining the training subset).

This process is repeated n number of times.

The cross-validation technique gives more accurate evaluation of the model than test train split and helps in overcoming the overfitting of the chosen model.

We use the cross\_val\_score method provided by sklearn library to perform cross validation

**Our cross\_val1-score method:**

In order to get the RMSE value we must set the scoring parameter of the cross\_val\_score to ‘neg\_root\_mean\_squared\_error’.

We put the cv parameter equals to 10 for all m odes except for the XGBoost model, we put cv=14.

# **5-Experimental Results**

**The results are taken after removing zero values and outliers, setting number of estimators 300 with 0.1 learning rate for XGBoost.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Linear Regression** | **Decision Tree** | **Random Forest** | **Lasso** | **XGBoost** | **Voting Regressor** |
| Without new feature | Mean | 1337.03 | 748.83 | 553.2 | 1337.23 | 522.71 | 524.3 |
| Std | 31.68 | 29.88 | 20.62 | 31.04 | 17.83 | 20.76 |
| Adding Volume feature | Mean | 1308.05 | 746.76 | 553.32 | 1308.71 | 524.36 | 524.11 |
| Std | 22.76 | 25.01 | 22.34 | 23.42 | 22.30 | 21.24 |
| Adding carat and volume feature | Mean | 1274.11 | 747.24 | 550.1 | 1277.53 | 525.53 | 522.99 |
| Std | 18.76 | 31.44 | 19.45 | 19.36 | 18.77 | 19.05 |

There was an attempt to use a length to width ratio feature, but it gave bad results.

The **best result** was using voting regressor with random forest (n\_estimators=1000, random\_state=42)

And XGBoost (n\_estimators=300, lr= 0.1) with just adding volume feature.

# **6-Summary**

This project was a contribution in the Kaggle competition to predict the price of a diamond using certain input features.

Different machine learning models were tested to give the best score and different measures were used to perform the comparison.