1. **Introduction**
2. **Motivation & Environment**
3. **Dataset & Algorithms**

**Part 1. Dataset**

1. **Dataset Choosing**
2. **Preprocessing**

**Part 2. Algorithm Implementation**

1. **Theory – Overview of the Regression Models**
2. **Decision Tree Regression**
3. **Ensembling**
4. **Random Forest**
5. **XGBoost**
6. **Model Building**

**Part 3. Evaluation, Experiment & Result**

1. **Theory – Performance Metrics**
2. **Experiment Result and Analysis**
3. **Hyperparameter Tuning**

**IV: Conclusion**

**V: Reference**

**III. Dataset & Algorithms**

**Part 1. Data**

1. **Data Acquisition**

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

1. **Data Preprocessing**

We aim to predict the target variable **‘price’** and the predictors are the 4C’s and the dimensions. Before training the model with the dataset, we do some basic data exploration to gauge the overall data. We examine the features to determine their type, remove null values and normalize the features.

Table

Description automatically generated

**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 dimensions column (x, y, and z) which represent dimensionless diamonds indicating they are faulty points while examining the statistical summary of the dataset. From the initial data exploration we discovered the categorical and numerical variables along with their ranges (refer Table-1).

**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 confirm this relationship. However, we also observe that higher carat diamonds have higher price volatility.

Chart, scatter chart

Description automatically generated

The pairplots for cut, clarity and color against <Little More>

For the continuous variables, we plot the histograms to understand the distribution of these variables which would help us in identifying outliers in our dataset.

Chart

Description automatically generated

Chart, histogram

Description automatically generated

Chart, line chart

Description automatically generated

**Removing Outliers:**

Outliers are datapoints in the dataset whole values are far away from the rest of the datapoints. This may occur due to data variability or due 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 method and DBSCAN clustering are some of the many techniques for detecting outliers. We have used IQR (Interquartile Range) for removing outliers in our dataset after visualizing the data using boxplots.

After examining the boxplots of the features, we observe outliers in the dataset and thus proceed to remove them before we train our model with the data.

We have used the interquartile method for outlier detection which 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 datapoint greater than this upper limit
4. Subtract the constant from the first quartile and remove datapoints lesser than this lower limit

**Correlation Matrix:**

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:

Calendar

Description automatically generated

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 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 it. However, we have decided to keep it as the number of features in the dataset are low.

**Part 2. Algorithm Implementation**

**1. Theory - Overview of the Regression Models**

1. **Decision Tree Regression:**

This is a non-parametric, supervised machine learning algorithm. Programmatically, the Decision Tree is the “Nested IF ELSE condition”. It is implemented by the “Tree” Data structure. This algorithm breaks down the dataset into smaller subsets while at the same time an associated decision tree is built. Decision trees can handle both categorical and numerical data.

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. The advantages of DT are easy to understand, lesser data cleaning is required, non-linearity does not affect the model’s performance and the number of hyper-parameters to be tuned is almost null. Figure-1 depicts the terms used in decision trees.

Diagram

Description automatically generated

*Figure-1 Decision Trees*

* **Root Node:** This represents the topmost node of the tree that represents the whole data points.
* **Splitting:** It refers to dividing a node into two or more sub-nodes.
* **Decision Node:** They are the nodes that are further split into sub-nodes, i.e., this node that is split is called a decision node.
* **Leaf / Terminal Node:** Nodes that do not split are called Leaf or Terminal nodes. These nodes are often the final result of the tree.
* **Branch / Sub-Tree:** A subsection of the entire tree is called a branch or sub-tree.
* **Parent and Child Node:**A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of the parent node. In the figure above, the decision node is the parent of the terminal nodes (child).
* **Pruning:** Removing sub-nodes of a decision node is called pruning. Pruning is often done in decision trees to prevent overfitting.

**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 built for Regression ?

Decision trees are built by recursively splitting the training samples based on the features of the dataset. This is done by evaluating certain metrics, like ***Residual or Mean Squared Error*** for regression trees. The process is also different for discrete and continuous feature. **For** **discrete features** all of its possible values are evaluated, resulting in N calculated metrics for each of the variables, being N the number of possible value for each categorical value. For continuous features the mean of each two consecutive values (ordered from lowest to highest) of the training data are used as possible thresholds.

The result of this process is, for a certain node, a list of variables, each with different thresholds, and a calculated metric (MSE) for each variable/threshold tandem. The variable/threshold combination that gives us the**highest/lowest value**for the specific metric that we are using for the resulting children nodes (the **highest reduction or increase in the metric**) is selected.

**For a regression tree**, the prediction at the end is the mean of the values for the target variable at that leaf node.

Advantages:

* Easy to interpret
* Tolerant to missing values
* Require less training data than other ML models

Drawbacks:

* High variance with large depth and high bias with shallow depth. (Overfitting and Underfitting)
* Weak Learners

1. **Ensembling:**

Ensemble methods create or use multiple models and then combine them to produce improved results. They reduce bias and variance which in turn increases the generalization and accuracy of the models. Figure-1 represents the types of ensemble methods.

A picture containing timeline

Description automatically generated

*Figure-1 Ensemble Methods*

**Parallel Ensemble Technique:**

Base learners – the base model on which ensembling is based upon. In this parallel technique, base learners are generated parallely 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 assigning/penalizing the misinterpreted learners.

**Bagging: (Bootstrap Aggregation)**

**Random Forest** algorithm is a popular Bagging Technique. This technique mainly reduces the high variance faced by the base model resulting in greater accuracy and reducing overfitting. Bagging consists of two methods, Bootstrap and Aggregation.

* **Bootstrap:** Sampling technique where samples are derived from the training set using the replacement procedure. The sampling with replacement method is a randomized procedure. The base learning algorithm is run on the samples to complete the procedure.
* **Aggregation**: It incorporates all possible outcomes of the prediction and randomizes the outcome. Without aggregation, predictions will not be accurate because all outcomes are not put into consideration. Therefore, the aggregation is based on the probability of bootstrapping procedures or on the basis of all outcomes of the predictive models.

**Boosting:**

An ensemble technique that learns from previous predictor mistakes to make better predictions in the future. Boosting 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.

**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.

1. **Random Forest:**

Random Forest is one of the most popular bagging techniques. The primary aim of this is to reduce the high variance of the base learner. The base learner here is the “Decision Tree” with max depth, which has high variance and low bias. Sampling is done not only based on the samples/instances but also on features. Then aggregation is performed on the outcomes of the base learners making them more reliable and reducing overfitting.

Let’s say, if there is a Training set D, it is sampled with replacement technique, both rows and column sampling into k subsets. Each subset has different set of samples both rows and column(features) and are passed to the base learners. The outcome of the each k DT is then combined to get a more robust output. Figure-2 represents the bagging technique.

Important Features of Random Forest:

**1. Diversity-**Not all attributes/variables/features are considered while making an individual tree, each tree is different.

**2. Immune to the curse of dimensionality-** Since each tree does not consider all the features, the feature space is reduced.

**3. Parallelization-**Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

**4.  Train-Test split-**In a random forest we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.

**5.  Stability-**Stability arises because the result is based on majority voting/ averaging.

Diagram

Description automatically generated with medium confidence

*Figure-2 Bagging Technique*

Table -1 represents the comparison between Decision Trees and Random Forest

|  |  |
| --- | --- |
| **Decision trees** | **Random Forest** |
| 1. Decision trees normally suffer from the problem of overfitting if it’s allowed to grow without any control. | 1. Random forests are created from subsets of data and the final output is based on average or majority ranking hence the problem of overfitting is taken care of. |
| 2. A single decision tree is faster in computation. | 2. It is comparatively slower. |
| 3. When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction. | 3. Random Forest randomly selects observations, builds a decision tree and the average result is taken. It doesn’t use any set of formulas. |

*Table -1 Comparison of DecisionTrees and RandomForest*

Hyperparameters:

In random forest, they are used to increase the performance, that is more accurate predictions or make the model fast.

Performance related hyperparameters:

* **n\_estimators** – number of trees the algorithm builds 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****–*it 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. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
* ***oob\_score****– OOB* means out of the bag. It is a random forest cross-validation method. In this one-third of the sample is not used to train the data instead used to evaluate its performance. These samples are called out of bag samples.

1. **XGBoost: (Extreme Gradient Boosting)**

**XGBoost** is one of the most efficient Boosting techniques. As seen before, boosting technique involves sequential model building. It is an open-source implementation of the gradient boosting algorithm with some regularization parameters. It is applied mainly to reduce the bias error. The base learner of this algorithm is the decision tree with shallow depth which has high bias. Unlike other boosting algorithms where weights of misclassified branches are increased, in Gradient Boosted algorithms the loss function is optimised. Figure-3 depicts the features of the XGBoost.

A picture containing chart

Description automatically generated*Figure-3 XGBoost*

**Overview of Gradient Boosting Algorithm:**

The term “gradient boosting” comes from the idea of “boosting” or improving a single weak model by combining it with a number of other weak models in order to generate a collectively strong model. [Gradient boosting](https://developer.nvidia.com/blog/gradient-boosting-decision-trees-xgboost-cuda/) is an extension of boosting where the process of additively generating weak models is formalized as a gradient descent algorithm over an objective function. Gradient boosting sets targeted outcomes for the next model in an effort to minimize errors. Targeted outcomes for each case are based on the gradient of the error (hence the name gradient boosting) with respect to the prediction.

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. Lets understand how does XGBoost work in this case.

* **Step 1:** Initial Prediction and calculate residuals

This prediction can be anything. But for simplicity, lets consider the prediction to be the average value of the price of the diamonds. Lets say M is the average.

Residuals is given by the form : Residuals = Observed values – Predicted values

Calculating the residuals with observed values as the values of price in the diamond dataset and predicted value is the 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.

λ (lambda) is a regularization parameter that reduces the prediction’s sensitivity to individual observations and prevents the overfitting of data.The default value of λ is 1.

* **Step 3:** Splitting of this 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 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;

**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.

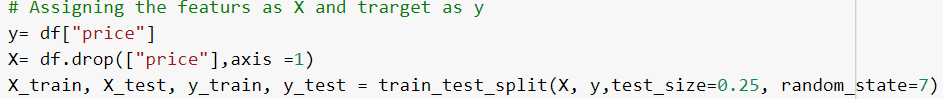
**Steps involved in Model Building**

* Setting up features and target
* 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.

1. **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.



1. **Creating instance of the model with default configurations**

**A screenshot of a computer

Description automatically generated with low confidence**

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)

1. **Fitting the models on the training dataset**

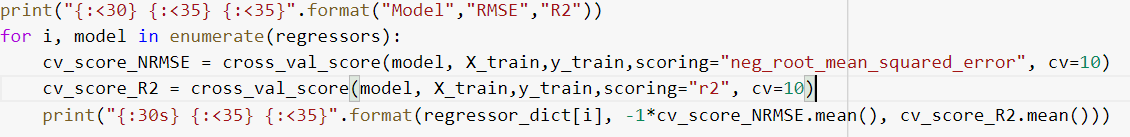
The sklearn provides a fit method for each ML algorithm that takes the responsibility of training the model with the given parameters. In Sklearn, fit() is used to find the attributes of the training set, including the mean, variance, maximum and minimum values, and so on.

**Graphical user interface, text, application, email

Description automatically generated**

1. **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.

****

The result of the CV:

|  |  |  |
| --- | --- | --- |
| **Model** | **RMSE** | **R2** |
| DecisionTree | 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.

1. **Predict using different models**

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

Graphical user interface, text, application

Description automatically generated

**Part 3. Evaluation, Experiment & Result**

**1. Theory – Performance Metrics**

The most common evaluation metric for regression is the RMSE (Root Mean Squared Error). To build and deploy a generalized model evaluate the model on different metrics which helps us to better optimize the performance, fine-tune it, and obtain a better result. 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.

A regression model is good if the difference between the actual and the predicted values is small, and unbiased for the train, validation and test sets.

1. **Mean Absolute Error**

Simplest of all the metrics. 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.

Figure-4 represents the equation of the MAE.

1. **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.

It is calculated using the formula:

1. **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. It represents the sample standard deviation of the differences between predicted values and observed values (also called residuals). RMSE is a better performance metric as it squares the errors before taking the averages. For that, large errors receive higher punishment.

Pros: Smoothly differentiable, same unit as the dependent variable.

Cons: Sensitive to outliers.

It is calculated using the following formula:

N= Total number of samples

1. **R Squared (R2)**

R2 score is a metric that tells about the performance of the model, not in the sense of loss or error rather how many wells did the model perform. It is also called as Goodness of fit. It helps you to understand how well the independent variable adjusted with the variance in your model. That means how good is the model for the given dataset. 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.

The mathematical representation for R^2 is given below :

SSR = Sum Square of Residuals(the squared difference between the predicted and the average value)

SST = Sum Square of Total(the squared difference between the actual and average value)

1. **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 is 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. Adjusted R2 is given by the following formula where

n = numberofobservations

k = number of independent variables

R = R2

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 RandomForest 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** |
| RandomForest | 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 RandomForest 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 RandomForest] on average more or less than the actual value. Figure-4 represents the actual and predicted values by RandomForest for the price continuous variable.

Table

Description automatically generated

*Figure-4 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 actually 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.

Text

Description automatically generated

*Figure-5 Default configuration of Random Forest*

**a. Hyperparameter Tuning:**

Random Forest Hyperparameter Tuning Considerations:

n\_estimators = number of trees in the foreset

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-6 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-7 represents the best parameters.

Text

Description automatically generated

*Figure-6 Random grid to determine hyperparameters*

Text

Description automatically generated

*Figure-7 Best Parameters for RandomForest*

The hyperparameter tuning has resulted in slight improvement in terms of the performance.

|  |  |  |
| --- | --- | --- |
| **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 |

Figure -9 and Figure- 10 shows the graphical representation of the predicted and actual values of the prices before and after hyperparameter tuning.

**Chart, scatter chart

Description automatically generated**

**Chart, scatter chart

Description automatically generated**

*Figure-9 Default Parameters Figure-10 Hyperparameter Tuned*

**IV. CONCLUSION**

Chart, scatter chart

Description automatically generated