# Regression Model Creation and Execution

This report provides an overview and evaluation of five Regressor models developed for predicting the price of diamonds (target variable) in 2024. The model training and validation process involved several key steps aimed at developing accurate and reliable predictive models.

# Data Cleaning

The dataset had been cleaned previously during exploratory data analysis. Consulting with our Alumni lead, Urenna, we realized we missed a few things so we went back and removed the index and year column, and only removed null and 0 columns in the carat column rather than in the dimensions columns.

# Encoding categorical variables

Most machine learning algorithms require numerical input data. The categorical variables were encoded to convert them into numerical format suitable for modeling.

# Train-test split

The choice of train-validation split depends on several factors, including the size of the dataset, the complexity of the model, and the goal of analysis, for example complex models tend to have a higher risk of overfitting, and in such cases, it's important to set aside a larger portion of the data for testing to ensure that the model generalizes well to new data. Similarly, building a predictive model for deployment in the real world may necessitate simulating real-world performance as closely as possible by allocating a larger portion of the data to the test set.

In practice, common train-test split ratios include 70/30, 80/20, or 90/10, where the first number represents the percentage of data allocated to the training set and the second number represents the percentage allocated to the test set. However, there's no one-size-fits-all answer, and the optimal split ratio may vary depending on the factors mentioned above.

The diamond sales dataset was split into training and validation sets using a 70-30 split ratio. All the models were trained on a dataset containing 10000 diamond samples and ten features, i.e carat, cut, color, depth, table, length, width, height, clarity, year. The training set was used to train the model, the validation set was used to tune hyperparameters and evaluate performance during training, and the test set was used to assess the final model's performance.

# Model building

Various machine learning algorithms (Table 1) were employed to train regression models on the training data, including decision tree regression, linear regression, random forest regression, k-nearest neighbors regression, and support vector regression. The respective classification and regression models were built and fitted on the training data by leveraging the respective python packages.

**Table 1:** Machine learning Algorithms used to train the models

| **Model** | **Algorithm used** |
|---|---|
| Decision Tree Regression | Decision Tree |
| Linear Regression | Ordinary Least Squares |
| Random Forest Regression | Random Forest |
| Support Vector Regression | Support Vector Machine |
| K Nearest Neighbor Regression | kNN |
| Extreme Gradient Boosting | XGBRegressor |

# Model Performance, Accuracy, and Limitations

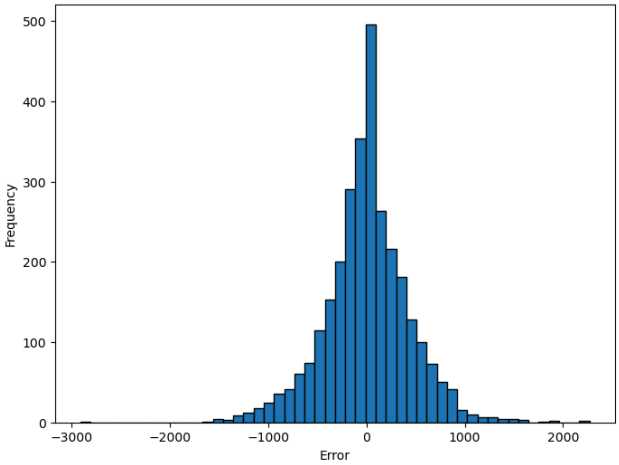
Once the models were trained, they were evaluated on the validation set to assess their performance. Evaluation metrics such as mean absolute error (MAE), mean squared error (MSE), and coefficient of determination (R-squared) were used to quantify the models' accuracy and performance. MAE measures the average absolute errors between the actual and predicted values. MSE quantifies the average squared differences between actual and predicted values. Lower MAE and MSE indicate higher accuracy. Additionally, visualization techniques such as histograms of modeling errors were utilized to gain insights into the models' predictive capabilities and identify any potential limitations or areas for improvement.

**Table 2:** Model Performance, Accuracy, Performance and Limitations

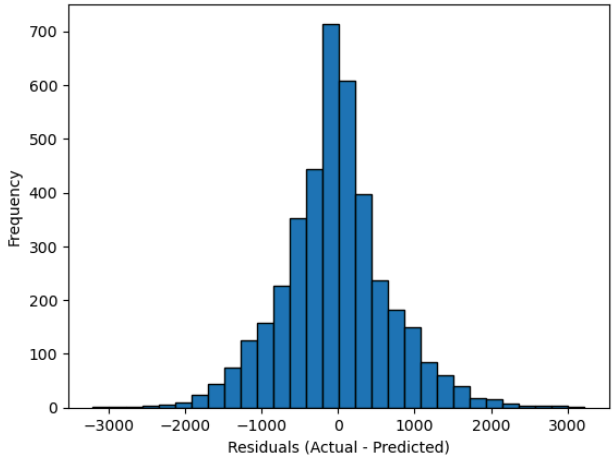
| Model | Regression statistics: training | | | Regression statistics: validation | | | Model Limitations |
|---|---|---|---|---|---|---|---|
| | MAE | MSE | R-squared | MAE | MSE | R-squared | |
| K Nearest Neighbor | 305.399 | 282237.539 | 0.986 | 344.639 | 351275.094 | 0.983 | ● Sensitive to outliers in the data<br>● May not perform well with a large number of predictors<br>● May not predict well beyond the range of values input in the training data<br>● Choice of k-value can significantly impact predictions |
| Random Forest | 294.99 | 258058.667 | 0.987 | 336.529 | 331733.186 | 0.984 | ● Requires more memory than other algorithms because it stores multiple trees. This can be a problem if the dataset is large, just like was the case for the diamond wholesale data<br>● The training time can be longer than other algorithms, especially if the number of trees and the depth of the trees are high<br>● Random Forest can be less interpretable than a single decision tree because it involves multiple trees. It can be difficult to understand how the algorithm arrived at a particular prediction.<br>● Although Random Forest is less prone to overfitting than a single decision tree, it can still overfit the data if the number of trees in the forest is too high or if the trees are too deep |
| Support Vector | 0.0722 | 0.008 | 0.98 | 0.0886 | 0.0115 | 0.97 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Linear regression | 891.587 | 1889332 | 0.90 | 871.6296 | 1790658.78 | 0.91 | Model easily affected by multicollinearity<br>Prone to underfitting<br>Sensitive to outliers<br>Non-linearity<br>Constant error variance<br>Simplistic in some cases<br>Linearity constraints |
| XGBRegressor (Gradient Boosting) | 421.89 | 541929.847 | 0.973 | 427.157 | 556557.6362 | 0.972 | ● Very accurate and Efficient with large data sets<br>● Sensitive to outliers<br>● Efficient for diff data types<br>● Effective for complex relationships<br>● Good with overfitting compared to other models especially when dealing with high dimensional data |

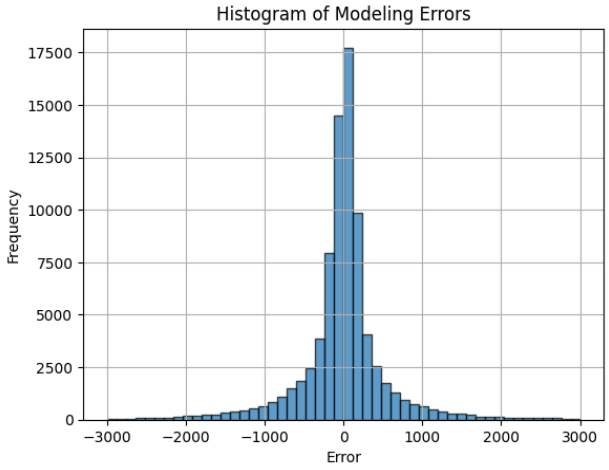
Histogram of modeling errors: Decision Tree Regression



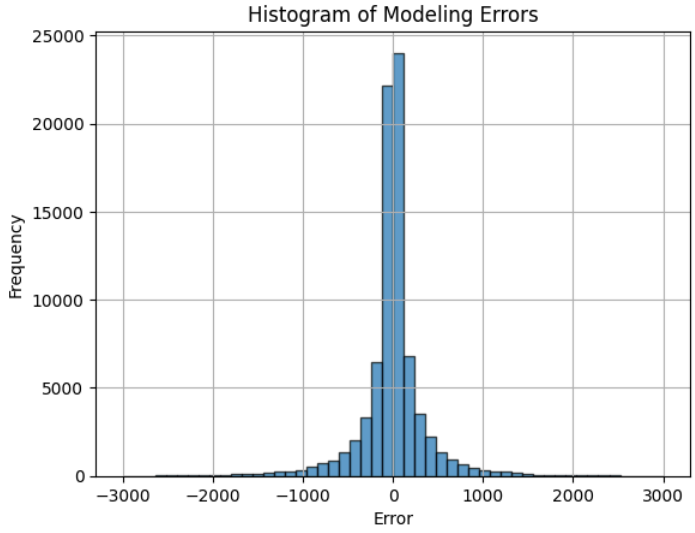

Histogram of modeling errors: k Nearest Neighbor




Histogram of modeling errors: Linear Regression




Histogram for XGB




Histogram for RandomForest