Predicting Diamond Prices Using Machine Learning Techniques

Team 11

Sudhagar Kalyanasundaram

Akash Reddy Yaddala

Sangamitra Balakumarasentthil

Department of Computer and Information Science, University of Michigan – Dearborn

CIS 579 - 002: Artificial Intelligence

Dr Khalid Kattan

December 19, 2024

1.Introduction

Abstract:

The final project challenges developing the best model for price prediction of a diamond using machine learning to provide valuable insights for buyers, sellers, and any other market analysts of the gemstone industry. Several factors determine the price of a diamond, basically the "Four Cs": carat for the weight, cut for the quality of the cut, color for the degree of colorlessness, and clarity for the absence of internal flaws. The estimation of prices in the jewelry or investment markets is very important for proper decision making and for complete transparency. The chosen topic revolves around supervised machine learning, where labeled data is used to train predictive models. Our approach integrates a fully functioning prototype that not only predicts diamond prices but also explores feature importance and the relationships between various attributes. The project involves developing an API service to make the predictions accessible in real-time, ensuring its practical applicability. This work will aim at showing the implementation of Machine Learning to solve a real-world problem. It talks about all changes in data pre-processing, mapping of categorical variables, and optimization of algorithms, which is iterative in any data science work.

Problem Statement or Objectives:

The motivation of this study is to offer the diamond market which traditionally relies on manual appraisals that are usually subjective and incoherent, hence very inefficient and unequal competent and easily accessible tool. The present project will help us both in learning and establishing the efficiency of different machine learning models, namely Linear Regression, Decision Tree, KNN, and XGBoost, for estimating the price of a diamond. We shall also know the impact which carat, cut, color, and clarity of a diamond may have on pricing.

2.Background and AI type:

Here, the chosen AI type is supervised machine learning, one of the strongest methods for predictive modeling. In supervised learning, algorithms are trained on a labeled dataset for which the input features and their respective output values are known. The model learns the pattern and relationship of such inputs and outputs to make the right prediction over new, unseen data.

The dataset contains features of diamonds, such as carat, cut, color, clarity, and x, y, z dimensions, while the target variable of interest would be the diamond price. In such a way, the labeled data will provide the necessary ground to develop predictive models able to estimate the price of any given diamond. The reason for choosing supervised learning is that it will be more applicable to deal with structured data whose output is well defined, such as continuous value prediction in regression analysis problems. In the project, several machine learning-supervised algorithms were implemented: Linear Regression, Decision Tree, K-Nearest Neighbors (KNN), and XGBoost all modified to handle nonlinear relationships and categorical variables. Such models were used to analyze the impact of the characteristics of a diamond on its price, listing the most relevant features responsible for the discrepancies in prices. These types of AI not only enabled correct forecasting of prices but took various stakeholders like jewelers, investors, and buyers closer to a scientific basis for conventionally subjective processes of valuation.

3.Dataset Description

The dataset provided for this work was sourced from Kaggle and involves very detailed information on 53,940 different diamonds, making it extensive and reliable for predictive modeling. It contains 10 variables that capture the key attributes of a diamond, some numerical and some categorical. The key target variable is the diamond price, ranging from $326 to $18,823, while the independent variables include carat (the unit of weight), cut-a measure of the quality of cut, color-color grade, clarity-purity grade, depth, table, and dimensions x, y, and z. These are basic Four Cs of diamond valuation supplemented with geometric proportions, giving an all-encompassing view of what could, in principle, affect diamond pricing. It's cleaned data: well-structured, doesn't contain any missing values or duplicate columns. Now it is machine learning ready.

In the exploratory data analysis, it was also found that some attributes such as price, carat, and dimensions had potential outliers. Treatment of these had to be done very delicately, as it would skew the model's predictions. Overall, this dataset was very sound for training and testing different machine learning algorithms; more precisely, some preprocessing steps and statistical methods increase its quality and usability for accurate diamond price prediction.

4. Data Acquisition Challenges

Dataset was easy to find on Kaggle, so the process of acquisition was quite smooth. Yes, during the time of preparation, there were some challenges because, though the dataset was big enough, some of the categorical features like cut, color, and clarity needed to be mapped into numerical equivalents for the training of the machine learning model. It will ensure that such qualitative features are understood by the model; however, it shall be done with due care not to introduce bias or distortion of the relationships.

Other challenges have included dealing with probable noise in the data. Most of these have been outliers in the numerical attributes comprising price, carat, and the different dimensions. These extreme values normally have an effect on model accuracy and, therefore, require some kind of EDA and statistical techniques that could reduce their influence. Also, assurance that the dataset is representative of real-world scenarios has called for closeness to feature distributions and relationships. Despite this, the data was enough to construct an accurate and meaningful prediction to support the objectives of this project.

```
color_mapping = {'D': 7, 'E': 6, 'F': 5, 'G': 4, 'H': 3, 'I': 2, 'J': 1}
clarity_mapping = {'IF': 8, 'VVS1': 7, 'VVS2': 6, 'VS1': 5, 'VS2': 4, 'SI1': 3, 'SI2': 2, 'I1': 1}
cut_mapping = {'Ideal': 5, 'Premium': 4, 'Very Good': 3, 'Good': 2, 'Fair': 1}

df['color'] = df['color'].map(color_mapping)
df['clarity'] = df['clarity'].map(clarity_mapping)
df['cut'] = df['cut'].map(cut_mapping)

print(df.head())
```

```
   carat  cut  color  clarity  depth  table  price     x     y     z
0   0.23    5      6        2   61.5   55.0    326  3.95  3.98  2.43
1   0.21    4      6        3   59.8   61.0    326  3.89  3.84  2.31
2   0.23    2      6        5   56.9   65.0    327  4.05  4.07  2.31
3   0.29    4      2        4   62.4   58.0    334  4.20  4.23  2.63
4   0.31    2      1        2   63.3   58.0    335  4.34  4.35  2.75
```

5. Exploratory Data Analysis:

This is one of the important stages in this project, which would help in finding the pattern, relation, and issues that are possible in the dataset.

5.1 Descriptive Statistics:

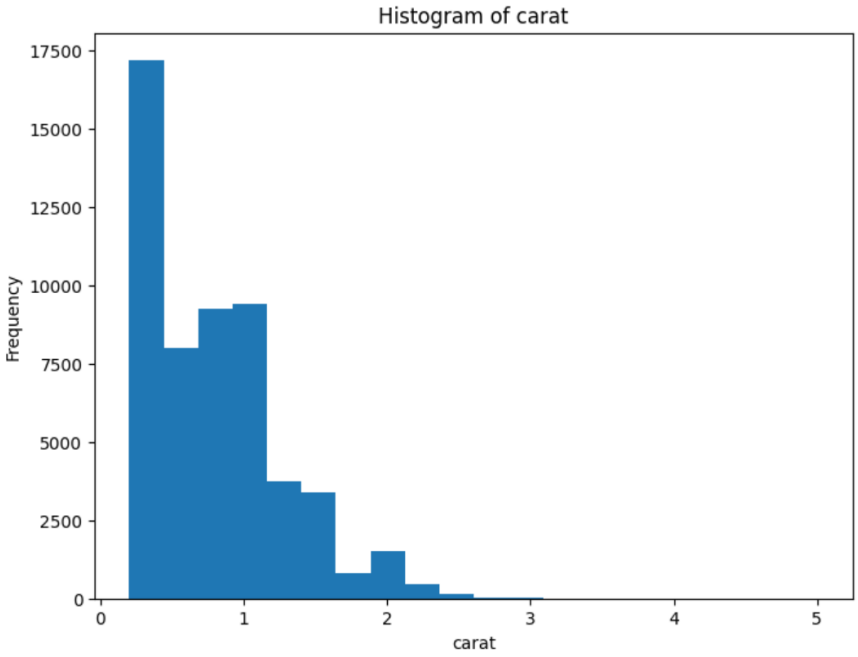
Summary statistics are the numeric features of this data: carat, price, depth, and dimensions x, y, and z. Each of these features has a mean, median, standard deviation, minimum, and maximum showing the distribution of data and pointing toward outliers.

```
print(df.describe())
```

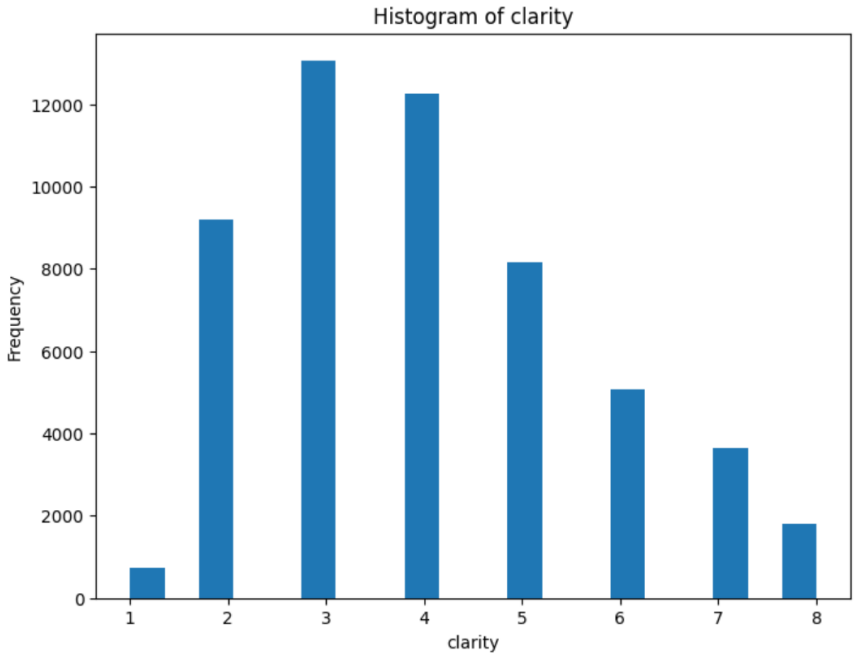
```
              carat           cut         color       clarity         depth  \
count  53940.000000  53940.000000  53940.000000  53940.000000  53940.000000
mean       0.797940      3.904097      4.405803      4.051020     61.749405
std        0.474011      1.116600      1.701105      1.647136      1.432621
min        0.200000      1.000000      1.000000      1.000000     43.000000
25%        0.400000      3.000000      3.000000      3.000000     61.000000
50%        0.700000      4.000000      4.000000      4.000000     61.800000
75%        1.040000      5.000000      6.000000      5.000000     62.500000
max        5.010000      5.000000      7.000000      8.000000     79.000000

              table         price             x             y             z  \
count  53940.000000  53940.000000  53940.000000  53940.000000  53940.000000
mean      57.457184   3932.799722      5.731157      5.734526      3.538734
std        2.234491   3989.439738      1.121761      1.142135      0.705699
min       43.000000    326.000000      0.000000      0.000000      0.000000
25%       56.000000    950.000000      4.710000      4.720000      2.910000
50%       57.000000   2401.000000      5.700000      5.710000      3.530000
75%       59.000000   5324.250000      6.540000      6.540000      4.040000
max       95.000000  18823.000000     10.740000     58.900000     31.800000

               size
count  53940.000000
mean     129.849403
std       78.245262
min        0.000000
25%       65.136830
50%      114.808572
75%      170.842451
max     3840.598060
```

5.2 Histogram:

Carat: Most diamonds weigh between 0.3 and 1.5 carats a right-skewed distribution.

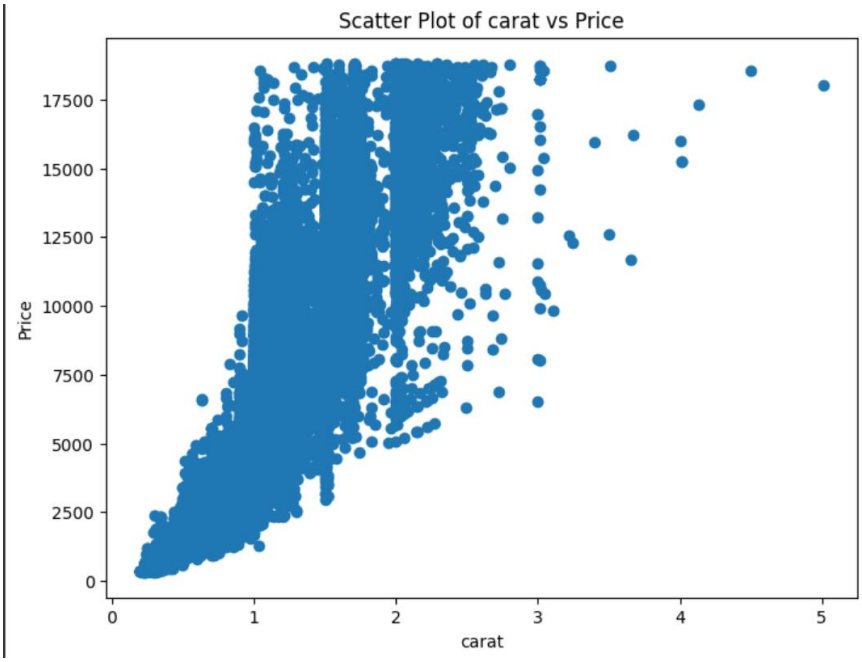


Clarity: Most common grades in the dataset, 3 and 4, correspondingly SI1 and VS2, characterize that the main part of diamonds in this dataset will be of medium clarity, which is normally considered acceptable because of a reasonable appearance and affordability.

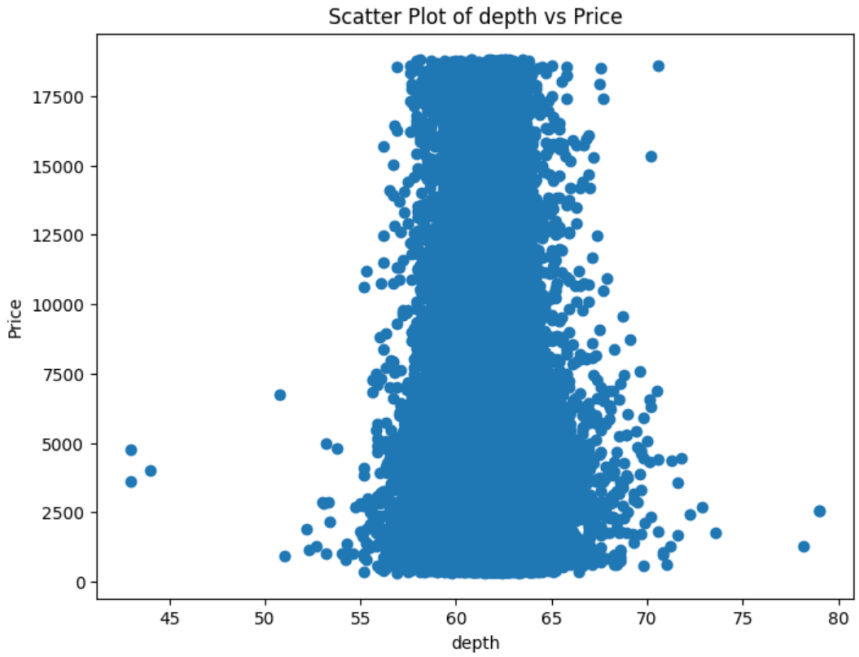


5.3 Scatter Plots:

Carat vs. Price: A positive correlation that proves the larger the diamond, the greater the increased price.



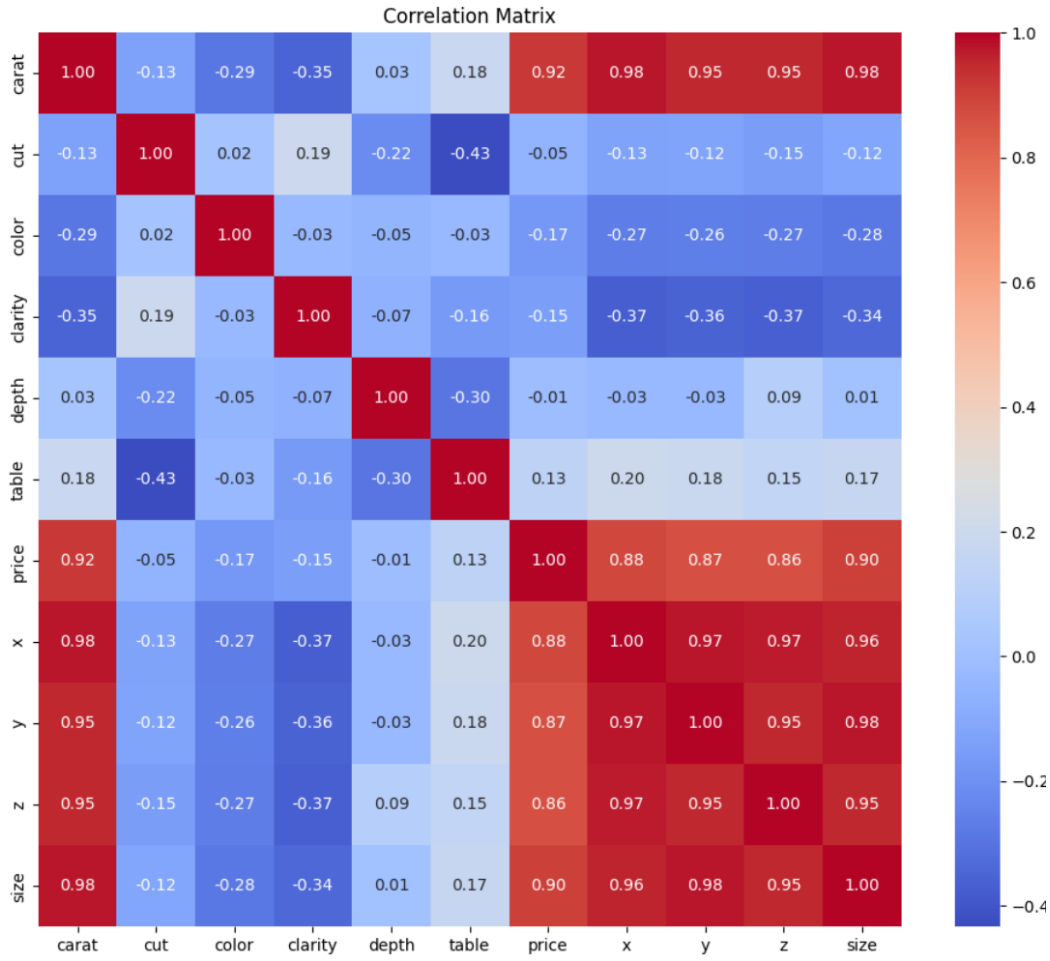
Depth vs. Price: It doesn't detect the relation to be linear, hence other factors influence the price more.



5.4 Correlation Matrix:

The following heatmap gives the correlation coefficient for all pairs of features. Important observations are:

- Strong positive correlation between carat and price.
- Weak correlation between depth and price; therefore, depth contributes less to pricing.



6. Model Overview:

This would therefore allow pricing a diamond based on characteristics derived through regression, with the help of a supervised machine learning model. Considering diversity in data characteristic handling besides their precision, Linear Regression, Decision Tree, KNN, and XGBoost have been taken into account to determine these models.

6.1: Tools and Technologies Used

Programming Language: Python

Libraries and Frameworks: Pandas, NumPy, Matplotlib, Seaborn**,** Scikit-learn

Development Environment: Jupyter Notebook and Visual Studio Code

Deployment Tool: Flask API for model integration into a real-time prediction service.

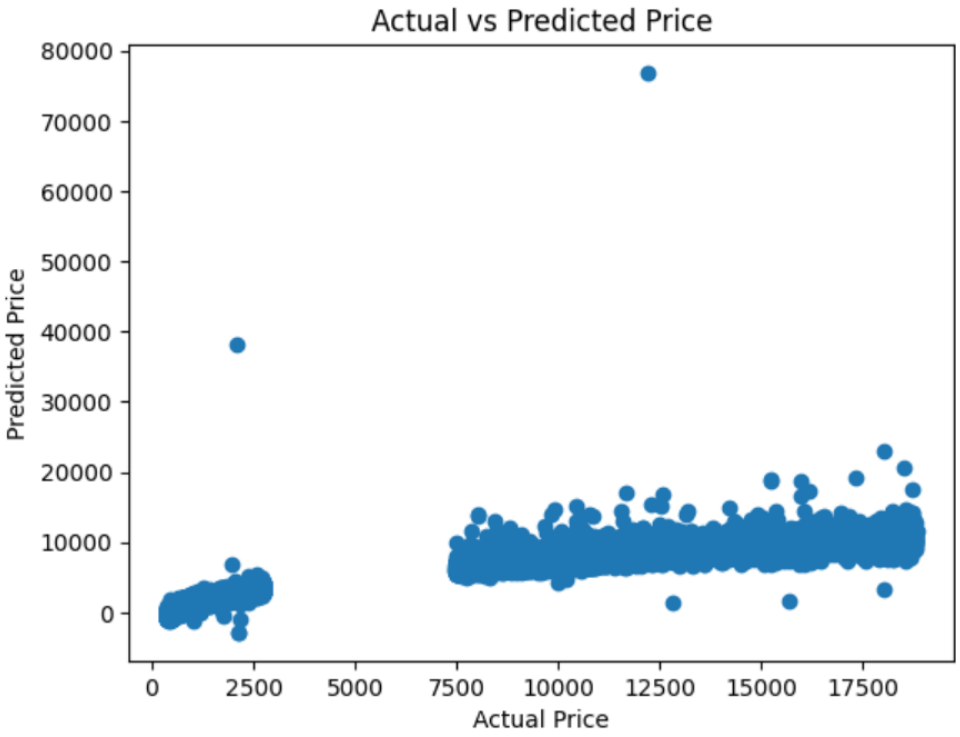
Testing Tool: Postman for testing the API's functionality and endpoints.

6.2: AI Models Used:

6.2.1: Linear Regression:

```
[ ]
    # linear regression model
    model = LinearRegression()

    model.fit(x, y)

    y_pred = model.predict(x_test)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'Mean Squared Error: {mse}')
    print(f'R-squared: {r2}')

    plt.scatter(y_test, y_pred)
    plt.xlabel('Actual Price')
    plt.ylabel('Predicted Price')
    plt.title('Actual vs Predicted Price')
    plt.show()

    Mean Squared Error: 4554071.705448246
    R-squared: 0.7975541615572482
```

A simple Linear Regression has been carried out that could serve as the baseline for the price prediction of the diamonds. This model is trained on this dataset using the class LinearRegression from Scikit-learn and was evaluated using the mean squared error and R^2 metric. With the MSE at about 4,554,071.7, this implies large variance for some of the predictions is the measure of prediction error. The R^2 value of 0.797 explained that the model explicated about 79.7% of the variance in the price of a diamond and fitted the model moderately well to the data.



A scatter plot of the actual versus predicted prices indicated that most of the predictions were very close to the actual values, though there were cases of large deviations, especially with high-priced diamonds. It also indicates the inability of the model to handle the outliers and non-linear interactions in the data. Similarly, the predictions for lowly priced diamonds are clumped together due to the nature of the data because most of the diamonds price less than $10,000. Although the Linear Regression model provided a quick and interpretable baseline, its assumption of linearity and sensitivity to outliers limited its ability to fully model the complexity of diamond pricing. These results point out the need for more advanced models to improve accuracy and handle the dataset's characteristics effectively.

6.2.2: Decision Tree:

```
# Decision tree

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


X = df[['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y', 'z']]
y = (df['price'] > df['price'].mean()).astype(int)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(random_state=42)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9668149796069707
```

Decision Tree is developed based on the features showing whether the price is above or below the mean. Features are Carat, Cut, Color, Clarity, Depth, Table, and Dimensions of the Diamond-Regarding X, Y, and Z. Instance segmentation is done for both training and testing of mentioned data in the ratio of 70:30. It achieved a fabulous accuracy of about 96.68%, while being able to estimate the complex decision boundary and nonlinear relationship among a set of features to the target price using a DecisionTreeClassifier from Scikit-learn. Again, this confirms the strength of the decision tree in performing iterative splits so as to find the importance among the important features.

6.2.3: KNN:

```
# KNN

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

X = df[['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y', 'z']]
y = (df['price'] > df['price'].mean()).astype(int)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9561549870226177
```

KNN was then applied to Scikit-learn's KNeighborsClassifier to classify a diamond having a price either above or below the average price in the data set. Some of the features the model used to make a prediction include carat, cut, color, clarity, depth, table, dimensions in x, y, z. For this, the data was divided into 70% training and 30% test sets. The best model was with n_neighbors = 5, giving an accuracy of 95.62%. That would mean that the performance was relatively good in the prediction of the classification of diamond prices. It is efficient and considers high accuracy because K-NN classifies the data according to the proximity that occurs within the feature space.

6.2.5: XGBoost:

```python
# XGBoost
import joblib
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier

X = df[['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y', 'z']]
y = (df['price'] > df['price'].mean()).astype(int)

categorical_columns = ['cut', 'color', 'clarity']
X = pd.get_dummies(X, columns=categorical_columns, drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

params = {
    'objective': 'binary:logistic',
    'max_depth': 4,
    'alpha': 10,
    'learning_rate': 1.0,
    'n_estimators': 100
}

xgb_clf = XGBClassifier(**params)

xgb_clf.fit(X_train, y_train)

y_pred = xgb_clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("XGBoost Model Accuracy:", accuracy)


joblib.dump(model, 'diamond_price_model.pkl')
print("Model saved as diamond_price_model.pkl")
```

```
XGBoost Model Accuracy: 0.9730255839822024
Model saved as diamond_price_model.pkl
```

The two will be used, being XGBoost, by using the XGBClassifier of the XGBoost library, on the classification of the diamonds as above or below average in price in the set. Some of the

features used include carat, cut, color, clarity, depth, table, and dimension (x, y, z). Since these are categorical features, one-hot encoding should be used in order for feature numerical representation to occur. The best settings were the max depth is equal to 4, the learning rate was 1.0 with 100 estimators, and yielded an astonishing result of 97.30% on the test set. That is quite remarkable performance from the point of view of handling complicated relationships coming through in this dataset between gradient boosting and trees. Additional code was implemented to save the model into a '. Pkl' file that can be reused for real-time predictions. XGBoost's robustness and scalability make it an excellent choice for this classification task, outperforming simpler models like Decision Trees and KNN.

7.Results:

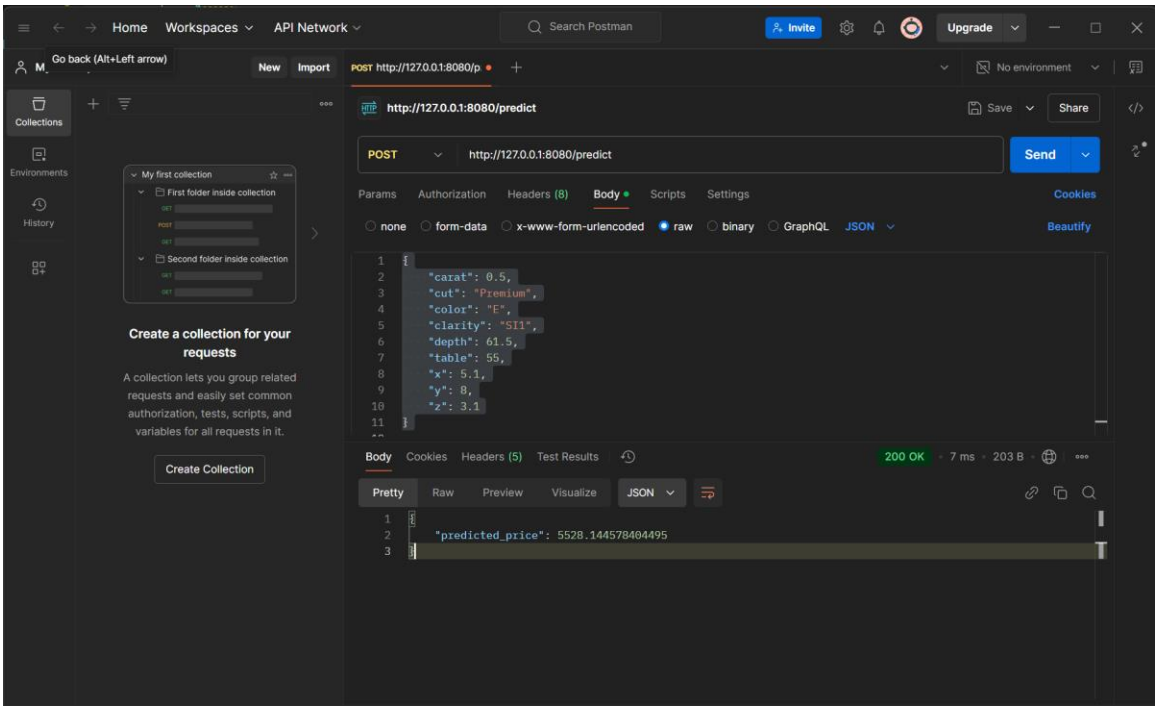
The use of scatter plots, histograms, and feature importances visualizations supported EDA. Important trends such as carat and cut being strong influencers of prices were thus found. Other steps include one-hot encoding for categorical variables, outlier pre-processing, and hyperparameter tuning for further upliftment of the model's accuracy.

```
# Feature importances
feature_importances = pd.DataFrame({'feature': X.columns, 'importance': clf.feature_importances_})

feature_importances = feature_importances.sort_values('importance', ascending=False)

print(feature_importances)
```

```
   feature  importance
7        y    0.826683
0    carat    0.051017
3  clarity    0.034575
2    color    0.033598
8        z    0.014120
6        x    0.013144
4    depth    0.011599
5    table    0.010444
```

Demonstrated capability in adequately forecasting diamond prices has been found most suitable using Linear Regression, Decision Tree, K Nearest Neighbors, and XGBoost. Of all these, XGBoost comes up with the best performing model with 97.30%, which is way better than the rest due to its handling of the functioning of real world problems. While the other models, such as Decision Tree at 96.68% and KNN at 95.62%, also did very well and proved to be effective

classifiers, Linear Regression, though handy for a baseline performance, recorded a low R^2 of 79.7%; the limitation of modeling nonlinear relationships reflected in the higher prediction error, MSE of 4,554,071.7.



A best performing model of XGBoost was implemented through Flask, which is one of the most popular lightweight Python web frameworks. The file app.service.py includes the Flask API code where the endpoint to make a real time prediction exists. It takes the diamond attributes in JSON input and utilizes the XGBoost model already trained in processing the data and returning the predicted classification. With that, it was put to intense testing through Postman, which tested the correctness and robustness of the API.

The hypothesis has been proved right because the models gave very accurate and reliable prediction about diamond prices. More interestingly, while working with XGBoost, performance improved drastically, indicating that this is one of the finest algorithms to work on complicated datasets. Starting from data preprocessing, passing through robust models, up to deployment using Flask, this project has successfully achieved its goals in the development of a usable and scalable solution for diamond price prediction.

8.Conclusion:

8.1: Project Impact:

With that, it also succeeded in proving that diamond price prediction can be done using key attributes like carat, cut, color, clarity, and dimensions with machine learning techniques. These insights are of much value for diamond pricing and contribute a lot to scientific understanding when it comes to making decisions based on data. This will have a social impact, showing how such AI models will impose much-needed transparency and consistency on traditionally subjective-dominated markets, such as valuations in jewelry. The implementation of a real-time prediction system via a Flask API, e-commerce, and valuation platforms also gave a hands-on explanation.

8.2: Future Plans:

This project has successfully predicted the prices using the machine learning models developed in Visual Studio Code and the tested Flask API deployed in Postman. Further developments for the future include developing this system into a very user-friendly website that will include a very intuitive interface with a database in its backend for persisting data related to diamonds and interactions created by the users. This would provide a basic layout where a user can input the attributes of the diamond and immediately get the price prediction, with other extra data visualization tools showing trends in prices and feature importance. We also wish that our system becomes more robust by synthesizing datasets from it to deal with bias and enhance generalizability, and adding extra data such as market trends and economic indicators, which could serve to further improve predictions. In the future, we want to continuously retrain the model with live data so it stays up-to-date for some time. These improvements will make the system more amicable to the end user and scalable and position the solution to this kind of AI-driven pricing challenge in various domains.

8.3: Summary:

8.3.1: Reflection on the Project:

That indeed was a great journey of mingling data analysis, machine learning, and real-world application development. I enjoyed the exploratory data analysis phase where the pattern in the data gave very interesting insights. But what hit most was the fact that as the models became more advanced, such as XGBoost, the performance improved a lot; again, this is a case study in the choice of algorithm. What annoyed me most was the handling of outliers and encoding of categorical variables, though these have taught me more in data pre-processing. The part that I am proud of most is how I have managed to deploy the model using a Flask API just to show its practicality and scalability.

8.3.2: Project Overview:

This project focused on predicting diamond prices using machine learning models and deploying the solution through a Flask API for real-time predictions. A dataset of over 53,000 diamonds was preprocessed and analyzed, with models like Linear Regression, Decision Tree, KNN, and XGBoost trained to classify diamonds based on price. XGBoost achieved the highest accuracy of 97.30%, demonstrating its effectiveness in capturing complex relationships. Overview: The produced model was then exposed through a Flask API, one endpoint takes in a JSON and returns the result of the price classification. Finally, the API has been tested for reliability and accuracy with Postman.