1.1 Summary of Dataset

The dataset consists of 6 continuous variables (carat, depth, table, x, y, z) and 3 categorical variables (cut, color, clarity). The dependent variable we are trying to predict is price. The length of the dataset is approximately 54,000 (rows).

1.2 Baseline Model

Since we are considering a regression problem, we first build a baseline model using multiple linear regression to obtain a basic understanding of the dataset. We split the training data into a 75-25 train-test split in order to evaluate our model performance. The fitted multiple linear regression model achieves an R-squared value of 0.92 which indicates a good model fit. However, there is a caveat as we have many features (26 features after adding dummy variables) and R-squared tends to inflate as the number of features grows. Testing the linear model on the test set in the split, we obtain a MSE of 1,348,035. This model achieves a score of 1102.14 which is worse than the benchmark submission's 709.92.

1.3 Issues with the Baseline Model

The statsmodels library reports "The smallest eigenvalue is 1.73e-24. This might indicate that there are strong multicollinearity problems or that the design matrix is singular." Checking the variance inflation factors verifies the conjecture. In particular, the VIF of feature x is 87.80, indicating the presence of strong multicollinearity. Furthermore, removing features results in poor model for this dataset. Ridge regression might mitigate the multicollinearity issue, but we will skip it and use boosting instead.

1.4 XGBoost (eXtreme Gradient Boosting) Model

The XGBoost Python library provides implementation for decision tree-based gradient boosting framework. Feature selection is handled by the package automatically and therefore we only need to tune the hyperparameters of the model. In particular, we used 10-fold cross validation and grid search (on n_estimators, max_depth, eta) to find an optimal set of parameters (n_estimators = 330, max_depth = 9, eta = 0.05, subsample = 0.7, colsample_bytree = 0.8). With this set of parameters, we achieved a MSE of 288,771 on the 25% test split which is significantly better than the MSE of 1,348,035 achieved by the linear model. The prediction from this XGBoost model scored 522.55.

1.5 Future Improvements

XGBoost is a powerful library that handles a lot of feature selection/engineering work for us. However, this also means that there are little space to improve by tuning an XGBoost model. To achieve a higher score, we could try to construct new features from the dataset based on field knowledge or observations. For example, we could add xyz = x * y * z as a new feature and try to use a linear model with the new feature which adds nonlinearity to the model.