Data Science Challenge

Abdullah al mu'adh

March 6, 2025

1 Introduction

This report outlines an end-to-end pipeline for predicting the continuous target variable `outcome` from a diamond dataset. The training file includes numeric (e.g., dimensions, synthetic features) and categorical (cut, colour, clarity) columns plus the target `outcome`. A separate test file omits **outcome**. Our goal is to train a model that generalises well, measured by R^2 and to choose the best model to compare to the test sets **outcome**.

2 Exploratory Data Analysis

Data Overview. The training set has $\sim$ 10,000 rows and 31 columns (including `outcome`). The test set has $\sim$ 1,000 rows with 30 columns (excluding **Outcome**. Key features include `carat`, `depth`, `table`, `price`, `x`, `y`, `z`, and synthetic columns `a#`, `b#`, plus categorical `cut`, `colour`, `clarity`.

Correlation Analysis. A correlation matrix (Fig.1) showed `price` correlates strongly with `carat`, `x`, `y`, `z` but has little correlation with `outcome`. Meanwhile, `depth` is moderately negatively correlated (≈ -0.4) with `outcome`. Also, `x`, `y`, `z` are strongly intercorrelated (> 0.8).

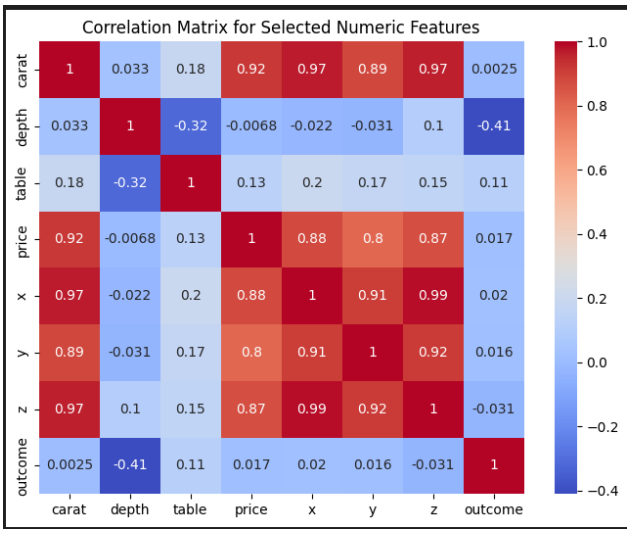

Figure 1: Sample correlation heatmap of numeric features.

Data Cleaning. We removed rows where any dimension was zero (`x`=0 or `y`=0 or `z`=0) and dropped `price` to reduce multicollinearity.

Feature Transformations. We created `volume = x * y * z` to capture overall size, then dropped `x`, `y`, `z`. We also applied one-hot encoding to `cut`, `color`, `clarity`, dropping the first category to avoid the dummy-variable trap.

3 Model Selection

Candidate Algorithms. We tested: (1) Linear Regression, (2) Random Forest, (3) XGBoost, and (4) a Neural Network (MLP). The tree-based methods (RF, XGBoost) often excel at capturing nonlinear interactions. The MLP can learn complex functions but may require extensive tuning. Linear Regression serves as a baseline.

Rationale. Random Forest is an ensemble of decision trees (bagging), robust to outliers. XGBoost uses boosting with regularization and often outperforms other models on tabular data. The MLP can approximate nonlinearities given sufficient data and tuning.

4 Model Training and Evaluation

Preprocessing. After removing invalid rows and dropping `price`, we computed `volume`, dropped `x`, `y`, `z`, encoded categorical features, and scaled numeric columns (e.g., `carat`, `depth`, `table`, `volume`).

Cross-Validation. We performed 5-fold CV with R^2. Each model trained on 4 folds, validated on the remaining fold, repeated 5 times, then averaged.

Hyperparameters. Random Forest used `n_estimators=200, max_depth=12`, XGBoost used `n_estimators=200, max_depth=6, learning_rate=0.01`, the MLP had hidden layers of `(100,50)`, and Linear Regression was standard OLS. The way these values were decided was via trial and error with the default as the inital reference. More rigorous tuning of the hyperparameters may potentially yield better results These values

Results. Table 1 shows approximate 10-fold CV R^2 means.

Model	R^2 Range	Time Taken (s)
Linear Regression	0.28–0.30	1.0
Random Forest	0.43–0.47	23.5
XGBoost	0.43–0.47	2.2
Neural Network	0.35–0.40	2.7

Table 1: Approximate 10-fold CV performance and corresponding evaluation time.

In our experiments, Random Forest or XGBoost typically outperformed the others in both R^2 and time for execution. The MLP exceeded the linear baseline but did not surpass tree ensembles under initial settings.

Final Model. We retrained the best performer (e.g., Random Forest) on all training data and generated predictions on the **test set** as a single-column CSV ($\hat{y}$).

5 Conclusion

We executed a full workflow: removing invalid rows, dropping `price`, encoding categorical variables, computing `volume`, scaling numeric columns, and comparing four models. A tree-based ensemble (RF or XGBoost) generally gave the highest $R^2 \approx 0.45$–0.48. Future improvements could involve deeper hyperparameter tuning, feature selection of synthetic `a#`, `b#` variables, or ensembling multiple strong models or using grid search on XGboost to find the best hyperparameters.

Bibligraphy

1. Scikit-learn: `https://scikit-learn.org/stable/`
2. XGBoost: `https://xgboost.readthedocs.io/`
3. Neural Network (MLP): `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`
4. Random Forest: Breiman, L. *Random Forests*, Machine Learning, 45(1), 5–32, 2001.

Code Repository: `https://github.com/Abdullah21200/data_science_challenge`