CW1 – Machine Learning Pipeline Report

5CCSAMLF – Machine Learning

K23114605

February 2026

1 Exploratory Data Analysis

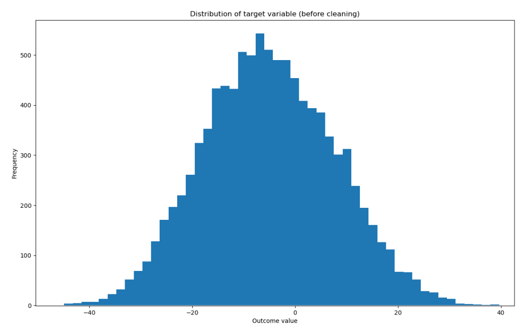
The training set contains 10,000 diamonds with 30 features and a continuous target variable (`outcome`). Features include standard diamond attributes and 20 additional numeric columns (`a1`–`b10`). No missing values were observed.

Target distribution. The outcome variable is approximately normally distributed, with a mean of around -5.0 and a standard deviation of about 12.7 (Fig. 1a). As the distribution is already reasonably well-behaved, no transformation was applied.

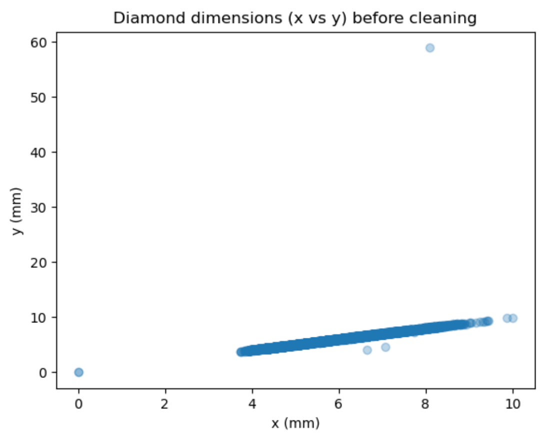
Correlation analysis. A Pearson correlation analysis was used to get a quick idea of which features have a linear relationship with the target. The strongest correlation was for `depth` ($r = -0.41$). Some of the unnamed features also had moderate correlations, such as `b3` ($r = 0.23$), `b1` ($r = 0.17$), and `a1` ($r = 0.15$). In contrast, commonly important diamond attributes such as carat ($r \approx 0.00$) and price ($r = 0.02$) showed very little linear relationship with `outcome`.

The dimensions x, y, and z were found to be highly correlated with carat ($r > 0.97$). This was not explicitly removed, because tree-based models are generally robust to multicollinearity.

Data cleaning. Some rows contained unrealistic measurements (e.g., $x = 0$, $y = 0$, or $z = 0$), which are physically impossible for a diamond. These rows were removed. In addition, conservative bounds were applied to `depth`, `table`, and the physical dimensions to exclude extreme and physically implausible outliers (for example, unusually large x, y, or z values). This removes a small number of problematic cases and improves training stability. After cleaning, 9,995 training samples remained. Figure 1b shows the dimension outliers before cleaning.



(a) Distribution of the target (`outcome`).



(b) Dimensions before cleaning, showing outliers.

Figure 1: Main EDA plots used in this coursework.

2 Model Selection

To choose a good model, I compared several regression methods with different levels of complexity:

1. **Ridge Regression** ($\alpha = 5.0$): a simple linear baseline with regularisation.
2. **Random Forest** (400 trees): an ensemble of decision trees trained on bootstrapped samples.

3. **XGBoost** (500 trees, learning rate = 0.05, depth = 5): a gradient boosting method with extra regularisation.
4. **Gradient Boosting Regressor (GBR)** (500 trees, learning rate = 0.03, depth = 3): boosting where each new tree improves on the previous errors.

Tree-based ensemble models were prioritised as the data showed non-linear relationships, correlated inputs, and mixed feature types, which these models handle well.

3 Model Training and Evaluation

Preprocessing. Categorical features were one-hot encoded. Numerical features were median-imputed and standardised. All preprocessing steps were placed inside an sklearn `Pipeline` to avoid data leakage during cross-validation.

Cross-validation. Each model was evaluated using 5-fold cross-validation and the R^2 score (Table 1). The mean shows average performance across folds, while the standard deviation indicates stability.

Table 1: 5-fold cross-validation results.

Model	Mean R^2	Std
Ridge Regression	0.2853	0.0181
Random Forest	0.4557	0.0095
XGBoost	0.4631	0.0142
Gradient Boosting	**0.4719**	**0.0098**

Gradient Boosting achieved the best average R^2 (0.4719) with low variation across folds (std = 0.0098), which suggests consistent performance. Ridge Regression performed noticeably worse, which suggests that a simple linear model is not enough to capture the patterns in the data. XGBoost was competitive but slightly behind GBR; one possible reason is that the simpler structure of GBR (shallower trees and a smaller learning rate) may reduce overfitting on this dataset.

Hyperparameter tuning. After selecting GBR as the best overall model, `RandomizedSearchCV` was used (20 iterations with 5-fold CV) to tune key parameters that control model complexity and learning behaviour:

Table 2: GBR hyperparameter search space.

Parameter	Values
`n_estimators`	{100, 200, 300, 500}
`learning_rate`	{0.01, 0.05, 0.1}
`max_depth`	{2, 3, 4}
`subsample`	{0.6, 0.8, 1.0}

A smaller learning rate usually needs more estimators to learn the same patterns, while `max_depth` and `subsample` help control overfitting. The best configuration found by the search was refit on the full cleaned training set (9,995 samples), and then used to generate predictions for the 1,000 test examples.

Code Supplement

The full implementation is provided in `train_and_submit.py`. It loads the data, performs cleaning, builds preprocessing pipelines, compares models using cross-validation, tunes the selected model, refits on all training data, and saves the final prediction CSV. The code is available here: `https://github.com/aytenmarab/ml_cw1`.