## Assignment – Kaggle Competition: Regression with a Tabular Gemstone Price Dataset

### Part 1: Problem Description

As the title of my report indicates, I chose to participate in the Regression with a Tabular Gemstone Price Dataset Playground Competition. This was a regression problem, where the aim was to predict the price of a given gemstone given various features including the carat, cut, colour, clarity and depth. Most features were numeric, but a couple (cut, colour and clarity) were categorical. The dataset was synthetically generated by a deep learning model trained on the original Gemstone Price Prediction dataset, and there was an option to use the original dataset to explore how it differed from the synthetically-created dataset as well as whether inclusion in model training helped model performance. Results were evaluated on root mean squared error. The training and test sets (the latter without the price variable) were provided. In total, there were 194,000 observations in the training set and 129,000 observations in the test set.

### Part 2: Analysis Approach

Before modelling, I conducted an exploratory analysis on the training data to determine if there was anything I should be aware of that would impact on the modelling stage; for example missing data, skewed data etc. I also wanted to know if there were any strong relationships between the features and/or between the features and the target variable. I ignored the test set as it lacked the price variable and I was treating it as new gemstones that didn’t yet exist.

I trained a simple linear regression model as a measure of baseline performance and then a gradient boosted tree as my actual model.

### Part 3: Initial Solution

I chose to undertake my analysis in Python as it has better native support for larger datasets than R.

The exploratory data analysis was conducted primarily using the pandas library, and consisted of visualising a seaborn pairplot and then constructing a visualising a correlation heatmap. Neither of my algorithms can natively handle categorical variables, so I used a mapping function to construct a numeric representation of cut, clarity and colour as the data dictionary revealed a natural ordering to the labels. I also found that the price variable was very skewed so transformed it using numpy’s log function.

I split the data using sklearn’s train\_test\_split function with a test size of 20% and set the seed to 23.

I trained a linear regression using statsmodels’ OLS function as I wanted the model diagnostics, which aren’t easily available using sklearn’s LinearRegression function and a baseline LGBMRegressor from the lightgbm package.

I fit the model to the training set and made predictions on both the training and validation set to check for model overfitting in addition to comparing the performance of the two algorithms. Although I was primarily interested in RMSE, I also used the sklearn package to output R-squared, mean absolute error, median absolute error and mean absolute percentage error metrics for both the training and validation sets.

### Part 4: Initial Solution Analysis

The baseline linear model resulted in a RMSE score of around 832 on the training data and 820 on the validation data. The booted tree model did better with a training RMSE of 555 and a validation RMSE of 574. However, the linear model diagnostic results unsurprisingly warned me of multicollinearity.

### Part 5: Revised Solution and Analysis

I wanted to see if I could improve on the baseline results for both algorithms. I first removed outliers using sklearn’s Local Outlier Factor algorithm with default parameters. This helped the linear regression but had a negligible impact on the boosted tree. Next, I tuned the hyperparameters of the boosted tree using Optuna. I optimised on RMSE and ran 30 trials. I found that the hyperparameter helped the performance on the training data (RMSE dropped to 543), but actually worsened on the validation data.