**Combine train and test sets**

this code is preparing the data for machine learning modeling by extracting the necessary columns, dropping unnecessary columns, and concatenating the training and test data together into a single DataFrame for further processing.

**Cleaning**

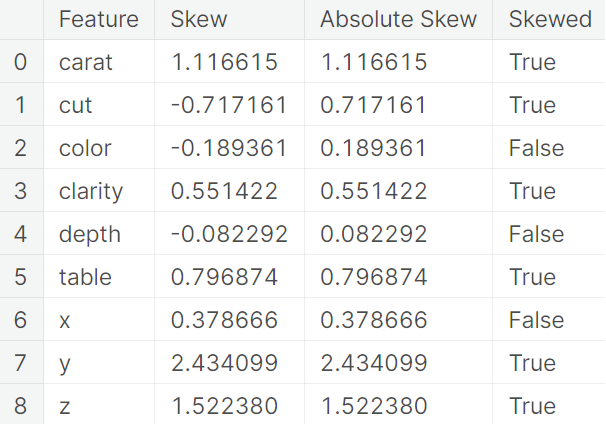
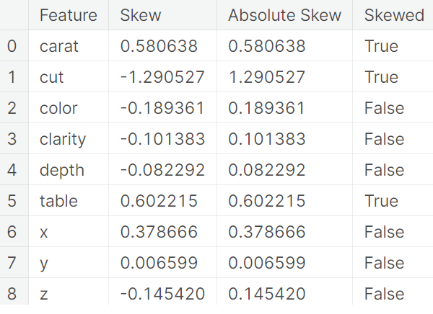
By replacing the original categorical values with numeric values, making the data more suitable for use in machine learning models.

**Feature Transformations**

calculating the skewness of the numeric features in the data1 DataFrame and storing the results in a new DataFrame called skew\_df using np.log1p function because there are zeros in the data.

applies a log transformation to each numeric feature in data1 that has an absolute skewness value greater than or equal to 0.5.

By applying this transformation, the code attempts to reduce the skewness of the skewed features in the data.

**Encode**

By creating dummy variables, the code transforms the categorical data into a format that can be used more easily by machine learning models.

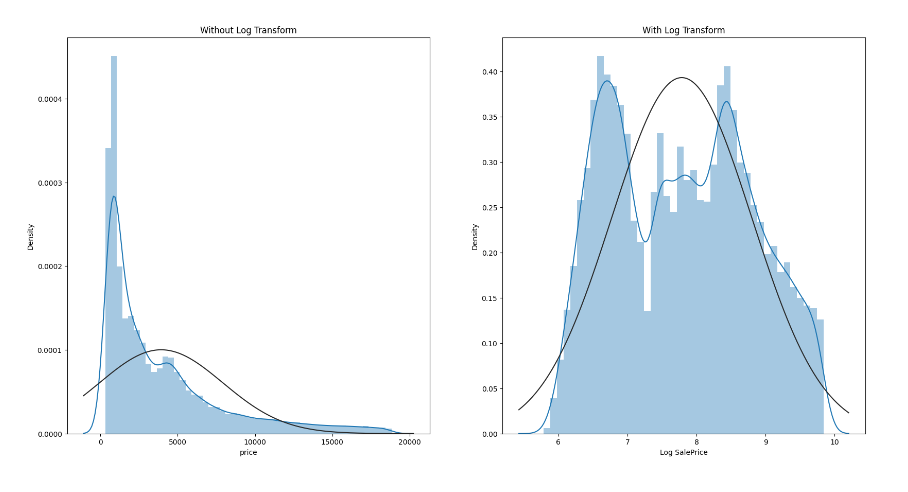
**Scaling**

ensures that each feature has a mean of 0 and a standard deviation of 1, which can help improve the performance of some machine learning models.

**Target Transformation**

compare the distribution of the target variable (target) before and after applying a logarithmic transformation.

The target variable is transformed using np.log() to take the logarithm of each value.



**Split Data**

Splits the combined and preprocessed dataset into two separate datasets, train\_final and test\_final. The train\_final dataset contains the same rows as the original training dataset train0, while the test\_final dataset contains the remaining rows from data4, which correspond to the original test dataset test0.

**Model Selection**

Using the PyCaret machine learning library. It is using the setup() and compare\_models() functions to prepare the data for modeling and compare the performance of different regression models.

this code is setting up and comparing the performance of different regression models for the given dataset using the PyCaret library.

**Bagging Ensemble**

This code defines a dictionary models which contains three different regression models:

CatBoostRegressor: a gradient boosting model that uses decision trees as base learners and has been designed to handle categorical features efficiently.

XGBRegressor: a gradient boosting model based on decision trees that are optimized for speed and performance.

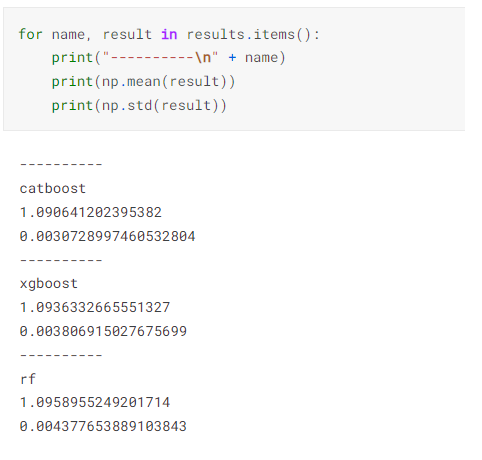
RandomForestRegressor: an ensemble learning model based on decision trees that fits multiple decision trees on different subsets of the data and averages their predictions to reduce overfitting.

This code fits each of the regression models in the models dictionary on the training data (train\_final) and target variable transformed using the log function (log\_target). The loop iterates over the items in the models dictionary, where name represents the name of the model and model represents the corresponding model object.

**Evaluate**

This code performs k-fold cross-validation using the KFold function from scikit-learn with 10 splits to evaluate the performance of three different regression models (CatBoost, XGBoost, and RandomForest) on the training data (train\_final) and target variable (log\_target).

calculates the root mean squared error (RMSE) using cross-validation for each model in the models dictionary, and then prints out the mean and standard deviation of the RMSE for each model. The purpose is to compare the performance of the different models on the dataset.



**Combine Predictions**

making final predictions on the test set using a weighted average of the predictions from three models (CatBoostRegressor, XGBRegressor, and RandomForestRegressor) that were trained on the training set. The first line of the code defines the weighted average formula where the final prediction is equal to 0.5 from the CatBoostRegressor model, 0.2 of the predictions from the XGBRegressor model, and 0.3 of the predictions from the RandomForestRegressor model. The resulting predictions are stored in the final\_predictions variable.

**Make Submission**

creates a submission file for the competition by concatenating the test IDs and the final predictions of the ensemble model.