**Diamond Price Prediction Project Steps**

1. **Importing Libraries and Modules:**

* Pandas, Matplotlib, Seaborn, and scikit-learn for data manipulation, visualization, and machine learning.
* Specific regression models: Linear Regression, RandomForestRegressor, GradientBoostingRegressor, etc.
* XGBoost library for XGBoostRegressor.
* Preprocessing modules: OneHotEncoder, OrdinalEncoder, ColumnTransformer, Pipeline, StandardScaler.
* Performance metrics: mean\_squared\_error, r2\_score.
* GridSearchCV for hyperparameter tuning using cross-validation.
* Filtering warnings for enhanced code readability.

1. **Reading Data:**

* Loads the training and test datasets from CSV files into Pandas DataFrames.

1. **Exploratory Data Analysis (EDA):**

* Displays the first few rows of the training dataset to inspect its structure and contents.
* Checks for duplicated rows in the training dataset and computes the sum of duplicates.
* Checks for missing values (NaN) in the training dataset and computes the sum of missing values for each column.
* Provides information about the training dataset including the data types of columns and memory usage.

1. **Data Visualization:**

* Price Distribution: Histogram showing the distribution of diamond prices.
* Carat Distribution: Histogram displaying the distribution of diamond carats.
* Categorical Variables:
* Distribution plots for 'cut', 'color', and 'clarity' categorical variables.
* Price vs. Carat: Scatter plot showing the relationship between diamond price and carat.
* Price vs. Cut/Color/Clarity:
* Box plots illustrating the relationship between diamond price and cut quality, clarity grade.
* Dimension Analysis:
* Scatter plot depicting the relationship between diamond price and length.
* Price vs. Carat with Color Encoding for Cut:
* Scatter plot with carat on the x-axis, price on the y-axis, and color encoded by cut quality.
* Correlation Matrix:
* Heatmap displaying the correlation matrix between numerical features in the dataset.

1. **Linear Regression Model Training and Evaluation:**

* Splits the dataset into features (X) and target variable (y) for training.
* Divides the data into training and testing sets using train\_test\_split with a test size of 20% and a random state of 42.
* Instantiates and trains a Linear Regression model on the training data.
* Makes predictions on the test data.
* Computes the root mean squared error (RMSE) and R-squared (R2) score to evaluate the model's performance on the test set.
* Prints the RMSE and R2 score for assessment.

1. **Model Training and Evaluation with Encoding Techniques:**

* Defines features (X) and target variable (y) from the dataset.
* Splits the data into training and testing sets.
* Defines ColumnTransformers for one-hot encoding and ordinal encoding of categorical columns ('cut', 'color', 'clarity').
* Constructs pipelines for linear regression models with one-hot encoding and ordinal encoding.
* Fits the models to the training data and makes predictions on the test data.
* Computes and prints the mean squared error (MSE) and R-squared (R2) score for both encoding techniques to evaluate model performance.

1. **Cross-Validation for Model Evaluation:**

* Defines features (X) and target variable (y) from the dataset.
* Defines a ColumnTransformer for one-hot encoding categorical columns ('cut', 'color', 'clarity') and standard scaling numerical columns ('carat', 'depth', 'table', 'x', 'y', 'z').
* Constructs a pipeline with preprocessing steps including one-hot encoding and standard scaling followed by linear regression.
* Performs cross-validation with 5 folds using the pipeline and computes the negative root mean squared error (RMSE) scores.
* Prints the cross-validation RMSE scores and the mean RMSE score for model evaluation.

1. **Residual Analysis:**

* Aligns the indices of y\_test and y\_pred\_test to ensure correct mapping.
* Calculates residuals by subtracting predicted values from actual values.
* Visualizes the distribution of residuals using a histogram plot with 40 bins and kernel density estimation (KDE) for smoother visualization.
* The plot helps in understanding the distribution and pattern of errors (residuals) made by the model.

1. **Model Evaluation with Various Regression Models:**

* Defines preprocessing steps including standard scaling for numerical features and one-hot encoding for categorical features.
* Defines a dictionary of regression models to evaluate, including Linear Regression, Ridge Regression, Lasso Regression, ElasticNet Regression, Decision Tree Regression, Random Forest Regression, Gradient Boosting Regression, and k-Nearest Neighbors Regression.
* Constructs pipelines for each model with preprocessing and the respective regression model.
* Performs cross-validation with 5 folds for each model and computes the negative root mean squared error (RMSE) scores.
* Prints the mean RMSE score for each model to evaluate their performance.

1. **Feature Engineering and Model Evaluation:**
   * Calculates a new feature 'volume' by multiplying the dimensions 'x', 'y', and 'z'.
   * Selects numerical features 'carat' and 'volume' along with categorical features 'cut', 'color', and 'clarity' as predictors.
   * Defines the target variable 'price'.
   * Splits the data into training and testing sets.
   * Defines preprocessing steps including standard scaling for numerical features and one-hot encoding for categorical features.
   * Defines evaluation metrics to compute: Mean Squared Error, Root Mean Absolute Error, and R-squared Score.
   * Performs cross-validation for each top model with 5 folds and calculates evaluation metrics using cross\_val\_predict.
   * Prints the evaluation metrics for each model to assess their performance after feature engineering.
2. **Feature Selection and Model Training:**

* Selects features including numerical features 'carat' and 'volume' and categorical features 'cut', 'color', and 'clarity', along with the target variable 'price'.
* Splits the data into training and testing sets.
* Defines preprocessing steps for numerical features (no scaling necessary for tree-based models) and categorical features using OneHotEncoder.
* Bundles preprocessing for both numerical and categorical features using ColumnTransformer.
* Defines models including Linear Regression and Random Forest.
* Appends each model to the preprocessing pipeline and trains it on the training data.
* Predicts on the test set and calculates the root mean squared error (RMSE) for each model to evaluate their performance.

1. **Stacking Regression with Hyperparameter Tuning:**
   * Defines numerical features ('carat', 'volume') and categorical features ('cut', 'color', 'clarity').
   * Sets up preprocessing steps for numerical features (no transformation) and categorical features using OneHotEncoder.
   * Defines base models for stacking including Random Forest and XGBoost.
   * Creates a stacking regressor with a meta-regressor using StackingRegressor.
   * Constructs a pipeline including preprocessing and the stacking model.
   * Defines a grid of hyperparameters for the stacking model including the number of estimators and maximum depth for the final estimator.
   * Performs grid search with cross-validation to find the best hyperparameters for the stacking model.
   * Prints the best hyperparameters found by grid search.
   * Uses the best stacking model obtained from grid search to predict on the test set.
   * Calculates the root mean squared error (RMSE) for the stacking model to evaluate its performance.
2. **Model Prediction and Submission Preparation:**

* Defines features (X\_test) for the test dataset.
* Makes predictions on the test dataset using the trained model (pipeline).
* Prepares the predictions along with their corresponding IDs for submission.
* Saves the submission file as 'submission.csv' without including the index column.

1. **Stacking Regressor Model with Custom Hyperparameters:**
   * Defines numerical features ('carat', 'volume') and categorical features ('cut', 'color', 'clarity').
   * Sets up preprocessing steps for numerical features (no transformation) and categorical features using OneHotEncoder.
   * Defines base models including RandomForestRegressor, GradientBoostingRegressor, and XGBRegressor with custom hyperparameters.
   * Initializes the StackingRegressor with the defined base models and a RidgeCV as the final estimator with 5-fold cross-validation.
   * Creates a pipeline including preprocessing and the stacking regressor model.
   * Fits the model on the training data.
   * Makes predictions on the testing data.
   * Calculates the root mean squared error (RMSE) and R-squared score of the model to evaluate its performance.
2. **Final Stacking Regressor Model:**

* Encodes categorical variables using one-hot encoding.
* Splits the data into training and testing sets.
* Defines base models including RandomForestRegressor, GradientBoostingRegressor, and XGBRegressor with optimized hyperparameters.
* Initializes the StackingRegressor with the defined base models and a RidgeCV as the final estimator with 5-fold cross-validation.
* Fits the model on the training data.
* Makes predictions on the testing data.
* Calculates the root mean squared error (RMSE) and R-squared score of the model to evaluate its performance.
* This model represents the final trained and evaluated model for predicting diamond prices.

1. **Final Submission Preparation:**
   * Creates a copy of the 'Id' column from the test dataset.
   * Preprocesses the test data by calculating the 'volume' feature and applying one-hot encoding.
   * Ensures that the test data has the same columns as the training data.
   * Uses the best model to make predictions on the preprocessed test data.
   * Creates a Dataframe for submission including 'Id' and predicted 'price'.
   * Saves the submission Dataframe as a CSV file named 'final\_submission.csv'.

**Summary**

* The final model is a Stacking Regressor composed of base models including RandomForestRegressor, GradientBoostingRegressor, and XGBRegressor with optimized hyperparameters.
* The model preprocesses the test data by calculating the 'volume' feature and applying one-hot encoding, ensuring consistency with the training data.
* Predictions are made using the trained Stacking Regressor model on the preprocessed test data.
* The 'Id' column is preserved for submission, and the predicted prices are saved along with the corresponding 'Id' in a CSV file named 'final\_submission.csv'.
* This submission file can be used for evaluation in the diamond price prediction task.