**Explanation for House Prices-Advanced Regression Techniques**

This code file is designed to predict house prices using machine learning models (Random Forest & XGBoost). It follows a structured data preprocessing, feature engineering, and model training pipeline.

**1.Import Libraries**

* **Pandas** 🡪 Handles data manipulation.
* **Numpy** 🡪 Used for numerical operations.
* **Train\_test\_split** 🡪 Splits data into training and validation sets.
* **LabelEncoder** 🡪Encodes categorical features into numbers.
* **RandomForestRegressor** 🡪Machine learning model (decision-tree-based)
* **Xgboost** 🡪Another powerful regression model for boosting performance.

**2.Kaggle Dataset Link:**

<https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview>

**3.Load Dataset**

Reads **train.csv** and **test.csv** into Pandas DataFrames.

**4.Remove Outliers**

* **GrLivArea** (Above-groundliving area in square feet)
* Houses with **GrLivArea > 4000** are outliers and may affect model performance.
* Filtering them out improves model accuracy.

**5.Handle Missing Values**

* **Why?** Missing values can cause errors in machine learning models.
* Numerical columns 🡪Filled with the median value.
* Categorical columns 🡪Filled with **“None”** (instead of Nan)
* **Why df.copy?** 🡪Avoids modification warnings in pandas 3.0.

**6.Encode Categorical Data**

* Categorical features (e.g., **Neighborhood, HouseStyle**) must be converted to numbers.
* **LabelEncoder()** 🡪Assigns each category a unique number.
* Unseen categories in the test set are assigned **-1** to prevent errors.

**7.Feature Engineering**

* **Why?** Some houses have basements, others have multiple floors.
* **TotalSF =Total livable sqare footage (basement + 1st floor + 2nd floor).**
* **Why apply(pd.to\_numeric, errors= “coerce”)?** Ensures all values are numeric.

**8.Define Features & Target**

* **SalePrice** (Target variable) is transformed using **np.log1p()**
* **Why?** Helps in reducing the effect of large values (improves model performance).
* Removes **Id** since its just an identifier.
* Extracts the same features for the test set (**x\_test**)

**9.Train Test Split**

* Splits the training data into:
* **x\_train, y\_train** 🡪Used for training.
* **x\_validation, y\_validation** 🡪Used for validation (to measure performance).
* **test\_size=0.2 🡪 20%** of data is used for validation.
* **random\_state=42** 🡪Ensures reproducibility.

**10.Train Random Forest Model**

* **Random Forest Regressor:**
* Uses multiple decision trees for better accuracy.
* **n\_estimators=100 🡪** Uses 100 trees in the forest.

**11.Train XGBoost**

* **XGBoost Regressor** (More powerful than Random Forest).
* **n\_estimators=1000** 🡪Trains 1000 trees.
* **learning\_rate=0.05** 🡪Controls the step size of weight updates.
* **Early Stopping:**
* Stops training if performance on validation set **(x\_validation, y\_validation**) stops improving
* The **try\_except** block ensures compatibility with different versions of **XGBoost**.

**12.Make Predictions**

* Reverses **log1p()** transformation using **np.expm1().**
* Combines predictions from models.
* **Why?** Blending models reduces errors.

**13.Save submission File**

* Create a CSV file with **Id** and **SalePrice** for kaggle submission
* Saves it as **submission.csv.**