Hotel Price Prediction — Project Report

🫘LOBIA MASALA🫘

Team members: Krish Patel (IMT2023134), Siddharth Anil (IMT2023503), Chaitya Shah (IMT2023055)

GitHub: https://github.com/kodercrish/Lobia-Masala

Abstract

This project details a machine learning pipeline for predicting hotel property prices. It encompasses raw dataset ingestion, exploratory data analysis, extensive preprocessing, and feature engineering, including outlier removal, feature merging, target and predictor transformations, and interaction terms. We trained and tuned various models such as Linear Regression, Random Forests, XGBoost, and ensemble methods, with model selection guided by RMSE and R^2 metrics. This report documents the dataset, preprocessing steps, modeling experiments, final model selection, and recommendations for future improvements.

I. Introduction

The primary goal of this project is to accurately predict hotel property values (HotelValue) using a provided real-estate-style dataset. Precise price prediction is crucial for stakeholders in making informed valuation, investment, and market analysis decisions.

Task & Scope

We address the prediction of HotelValue as a regression problem, utilizing a dataset of hotel properties with diverse physical, quality, and temporal features. Our work includes comprehensive EDA, data cleaning, advanced feature engineering, training and hyperparameter tuning of multiple algorithms, and the selection of the best-performing model for deployment.

II. Dataset Description

The train.csv dataset, provided for the assignment, has 1200 rows and 81 columns.

Examples of columns include: Id, PropertyClass, ZoningCategory, RoadAccessLength, LandArea, RoadType, ServiceLaneType, PlotShape, LandElevation, UtilityAccess, PlotConfiguration, LandSlope, District, NearbyTransport1, ConstructionYear.

Columns with the most missing values are:

- PoolQuality: 1194 missing values
- ExtraFacility: 1154 missing values
- ServiceLaneType: 1125 missing values
- BoundaryFence: 963 missing values
- FacadeType: 702 missing values
- LoungeQuality: 560 missing values
- RoadAccessLength: 223 missing values
- ParkingConstructionYear: 65 missing values
- ParkingCondition: 65 missing values
- ParkingType: 65 missing values

Example rows (first 5):

- Id: 775, PropertyClass: 20, ZoningCategory: RL, RoadAccessLength: 110.0, LandArea: 14226, RoadType: Pave ...
- Id: 673, PropertyClass: 20, ZoningCategory: RL, RoadAccessLength: nan, LandArea: 11250, RoadType: Pave ...
- Id: 234, PropertyClass: 20, ZoningCategory: RL, RoadAccessLength: 75.0, LandArea: 10650, RoadType: Pave ...
- Id: 427, PropertyClass: 80, ZoningCategory: RL, RoadAccessLength: nan, LandArea: 12800, RoadType: Pave ...
- Id: 197, PropertyClass: 20, ZoningCategory: RL, RoadAccessLength: 79.0, LandArea: 9416, RoadType: Pave ...

III. Exploratory Data Analysis & Preprocessing

The project employed a comprehensive preprocessing pipeline, detailed below:

Import Libraries & Datasets: The process begins by importing necessary Python libraries such as Pandas, NumPy for efficient data manipulation and numerical operations. For data visualization, matplotlib and seaborn are utilized to generate various plots for analysis, such as comparison plots and correlation heatmaps. Key modules from scikit-learn are imported to handle preprocessing tasks (StandardScaler, OneHotEncoder, ColumnTransformer, Pipeline, SimpleImputer), calculate performance metrics (root_mean_squared_error, r2_score), and manage cross-validation (KFold). Additionally, skopt (BayesSearchCV, Real, Categorical) is imported for hyperparameter optimization using Bayesian search. Finally, the training (train.csv) and testing (test.csv) datasets are loaded into pandas DataFrames, initiating the data preparation phase.

Key Preprocessing Steps:

Outlier Removal:

A crucial step in the preprocessing phase involved identifying and removing outliers from the **training dataset:**

- **Target Variable Extremes:** Properties with HotelValue falling within the lowest 0.1% or highest 0.1% quantiles were removed. This step aimed to exclude exceptionally low or high property values that might represent anomalies or unique market conditions not representative of the general dataset.
- Analysis using **correlation matrix** highlighted a good relationship between HotelValue and features like UsableArea and OverallQuality. **Scatter plots** revealed that properties with extremely high UsableArea (above 4000 sq ft) appeared to be outliers, leading to their removal from the training set. Using the same, properties exhibiting a combination of low OverallQuality (less than 3) were removed. Removing these specific outliers was deemed necessary to prevent these few extreme points within highly correlated features from distorting the model.

These targeted rules resulted in the removal of only **9 rows** from the training data. This relatively small number suggests that the dataset did not contain a large volume of extreme outliers requiring removal.

Feature Merging & Simplification:

Related columns were merged to reduce dimensionality and create more informative features:

- **Basement Features:** Information from `BasementFacilityType1`, `BasementFacilityType2`, `BasementFacilitySF1`, and `BasementFacilitySF2` was consolidated. Categorical types were mapped to numerical quality scores,

which were then combined with their respective areas to create `TotalBasementScore` (a quality-weighted area) and `BasementFinishedSF` (total finished area). A `BasementAvgQuality` metric was also derived. The original basement columns were subsequently dropped.
- **Pool Features:** SwimmingPoolArea was combined with PoolQuality to create a single TotalPoolScore feature, representing quality-weighted pool area. TotalPoolScore (PoolQualityNumeric * PoolArea). The original SwimmingPoolArea and PoolQuality columns were then removed.
- **Porch/Veranda Features:** Four separate area columns (OpenVerandaArea, EnclosedVerandaArea, SeasonalPorchArea, ScreenPorchArea) were summed into a single TotalPorchArea feature (sum of various porch/seasonal areas)The original four columns were dropped.

Column Dropping:

To further refine the feature set, several columns were explicitly removed:

- **Low Value / Sparsity:** Columns considered non-informative for prediction (Id), potentially having too many missing values, or low variance (BoundaryFence, ExtraFacility, ServiceLaneType, FacadeType) were dropped.
- **Low Correlation:** Features such as BasementHalfBaths, PropertyClass and LowQualityArea were removed after observing their **low correlation** with the target variable , HotelValue, during the correlation matrix analysis, indicating limited predictive power.

Target Transformation:

To handle the common right-skewness found in price data and certain feature distributions, logarithmic transformations using numpy.log1p() were applied.

- **Target Variable:** The primary target, HotelValue, was transformed to HotelValue_Log . This helps normalize the target distribution, which often benefits linear models.
- **Specific Features:** Based on observed skewness, several numerical features (ParkingConstructionYear, BasementFinishedSF, YearsSinceModification, LandArea, BasementTotalSF, ParkingArea) were also log-transformed . Importantly, after creating these transformed versions (e.g., LandArea_Log), the original columns were **dropped** from the dataset, ensuring only the log-scaled versions were used as input features for the model.

Ordinal Feature Mapping:

Categorical features with inherent order (ParkingCondition ,ParkingQuality , ParkingFinish, ExteriorQuality, ExteriorCondition , HeatingQuality, BasementCondition,BasementExposure, KitchenQuality, PropertyFunctionality) were manually mapped to numerical scores to preserve their rank information.

Advanced Feature Engineering:

Beyond merging and initial cleaning, several new features were engineered to capture more complex relationships within the data:

- **Time-Based Features:** To represent the property's age and time since the last update more directly, HouseAge (YearSold -ConstructionYear) and YearsSinceModification (YearSold- max(ConstructionYear,RenovationYear) were calculated. The original year-related columns (ConstructionYear, RenovationYear, YearSold, MonthSold) were subsequently dropped
- **Logarithmic Transformations (Features):** As noted previously, specific potentially skewed numerical features (ParkingConstructionYear, BasementFinishedSF, YearsSinceModification, LandArea, BasementTotalSF, ParkingArea) were log-transformed

Final Preprocessing Pipeline:

A ColumnTransformer was utilized to systematically apply distinct preprocessing steps to different feature types, ensuring the data was appropriately prepared for the linear models:

- **Numerical Feature Processing**:
    - Target Features: This pipeline targeted all columns identified as numerical, which included original numeric data, features created through manual ordinal mapping (like KitchenQuality), and engineered log-transformed features (like LandArea_Log).
    - Missing Value Imputation: Any missing values within these numerical columns were filled using the median strategy (SimpleImputer(strategy='median')). The median is robust to outliers, making it a suitable choice for potentially skewed distributions.
    - Standardization: Following imputation, all numerical features were scaled using StandardScaler.
- **Categorical Feature Processing:**
    - Target Features: This pipeline handled the remaining columns identified as categorical (nominal features like ZoningCategory or District).

    - Missing Value Imputation: Missing values in categorical columns were filled with the constant string 'None' (SimpleImputer). This approach treats missingness as a separate, distinct category.
    - One-Hot Encoding: The imputed categorical columns were then transformed using OneHotEncoder. This encoding is necessary because linear models require purely numerical input.
- **Final Output:** The ColumnTransformer effectively combined these processed numerical and categorical features into a single, fully numerical dataset, ready for input into the linear regression models

---

IV. Detailed Analysis on the models trained:

1. Random Forest Regressor:

a. Preprocessing: For the Random Forest model, a specific preprocessing pipeline suitable for tree-based algorithms was employed. This included all the feature engineering steps outlined previously (outlier removal, feature merging, manual ordinal mapping, time-based features, log-transforming specific predictors and dropping originals). Crucially, the final ColumnTransformer used OrdinalEncoder for remaining categorical features and did not apply StandardScaler to numerical features, as scaling is generally unnecessary for Random Forest.

b. Hyperparameter Tuning: GridSearchCV was used to optimize hyperparameters:

- The grid searched included: n_estimators: [100, 200] (Number of trees) max_depth: [None, 20] (Maximum depth of each tree) min_samples_leaf: [1, 3] (Minimum samples required at a leaf node)
- A 3-fold cross-validation strategy was used within the grid search.
- The best parameters obtained were: {'regressor__max_depth': 20, 'regressor__min_samples_leaf': 1, 'regressor__n_estimators': 200}
- Validation Performance: Models were trained on an 85% split of the training data and evaluated on the remaining 15% validation set. The performance was:
    - Baseline Random Forest (default parameters): RMSE = 30,038.42, R^2 = 0.8712
    - GridSearch Optimized Random Forest: RMSE = 29,938.99, R^2 = 0.8720

- Since the GridSearchCV optimized model showed slightly better performance on the validation set (lower RMSE), this model configuration was selected.
- Test Set Performance: The Root Mean Squared Error (RMSE) obtained on the final test dataset for this model was 34,168.219.

BayesSearchCV (from skopt) was used to efficiently search for optimal hyperparameters :

- The search space explored included ranges for: n_estimators: [100, 500] (Number of trees) max_depth: [10, 50] (Maximum depth of each tree) min_samples_leaf: [1, 10] (Minimum samples required at a leaf node) max_features: [0.1, 1.0]
- A 3-fold cross-validation strategy was used within the Bayesian search over 20 iterations. The best parameters identified were:
- Best Parameters: {'regressor__max_depth': 50, 'regressor__max_features': 0.2696..., 'regressor__min_samples_leaf': 1,'regressor__n_estimators': 500}
- Validation Performance: Models were trained on an 85% split of the training data and evaluated on the remaining 15% validation set. The performance was:
    - Baseline Random Forest (default parameters): RMSE = 30,038.42, R^2 = 0.8712
    - Bayesian Optimized Random Forest: RMSE = 29,456.75, R^2 = 0.8761
- The Bayesian-optimized Random Forest demonstrated superior performance (lower RMSE) on the validation set compared to the baseline. Therefore, this optimized configuration was selected.
- Test Set Performance: The Root Mean Squared Error (RMSE) obtained on the final, hidden test dataset (based on previous information provided) for a similar Random Forest model was 33663.477

C. Conclusion: Both tuning methods improved upon the baseline model. However, the **Bayesian Optimization (`BayesSearchCV`) approach yielded a superior model**, achieving a lower RMSE on both the validation set and the final test set.

2. XGBoost Regressor (with Bayesian Optimization)

XGBoost (Extreme Gradient Boosting) is a powerful and widely used gradient boosting algorithm known for its performance and regularization capabilities. Two main approaches were explored using XGBoost, both employing Bayesian Optimization for hyperparameter tuning but differing in their model selection strategy.

a. Preprocessing: For both XGBoost experiments, the preprocessing pipeline suitable for tree-based algorithms was used. This included all feature engineering steps (outlier removal, merging, extensive ordinal mapping, time features, log transformations of specific predictors with original drops, interaction features). The final `ColumnTransformer` employed **`OrdinalEncoder`** for remaining categoricals and performed **no scaling** on numerical features.

b. Experiment 1: Model Selection Based on Training Score (100% Data)

- **Hyperparameter Tuning:** `BayesSearchCV` was used to optimize hyperparameters over 30 iterations with 3-fold cross-validation on the **full training dataset**. The search space included ranges for `n_estimators`, `learning_rate`, `max_depth`, `subsample`, `colsample_bytree`, `reg_alpha`, and `reg_lambda`.
    - **Best Parameters Found (Training Score Run):**
      ```
      {'colsample_bytree': 0.827..., 'learning_rate':
      0.033..., 'max_depth': 3, 'n_estimators': 744,
      'reg_alpha': 0.011..., 'reg_lambda': 1.032...,
      'subsample': 0.753...}
      ```
- **Training Performance:**
    - Baseline XGBoost (default): Training RMSE = 855.31, Training R^2 = 0.9999
    - Bayesian Optimized XGBoost: Training RMSE = 9,617.32, Training R^2 = 0.9834
- **Model Selection:** Following the script's logic to select based purely on the **training RMSE**, the **Baseline XGBoost model was chosen**, despite its extremely low training error indicating severe overfitting.
- **Test Set Performance (Experiment 1):** The final RMSE on the hidden test dataset for this baseline model was **32,916.281**. This relatively high test error confirms the overfitting suggested by the training scores.

c. Experiment 2: Model Selection Based on Validation Score (85/15 Split)

- **Setup:** To get a more reliable estimate of generalization performance, the training data was split into 85% for training and 15% for validation.
- **Hyperparameter Tuning:** `BayesSearchCV` was run again (30 iterations, 3-fold CV) on the **85% training split**, searching a similar hyperparameter space.
    - **Best Parameters Found (Validation Score Run):**
      ```
      {'colsample_bytree': 0.7, 'learning_rate': 0.0187...,
      'max_depth': 3, 'n_estimators': 1000, 'reg_alpha':
      ```

`0.01, 'reg_lambda': 0.169..., 'subsample': 0.7}` *(Note: Different optimal parameters were found due to training on a subset and potentially different search paths)*

- **Validation Performance:**
  - Baseline XGBoost (trained on 85%): Validation RMSE = 28,234.71, Validation R^2 = 0.8862
  - Bayesian Optimized XGBoost (trained on 85%): Validation RMSE = **22,940.15**, Validation R^2 = **0.9249**
- **Model Selection:** Based on the superior performance on the **validation set**, the **Bayesian-optimized XGBoost model was selected**. This model was then **retrained on the entire 100% training dataset** using these optimal hyperparameters.
- **Test Set Performance (Experiment 2):** The final RMSE on the hidden test dataset for this properly selected and retrained model was **27,158.536**.

d. Conclusion for XGBoost: The results clearly demonstrate the importance of using a validation set for hyperparameter tuning and model selection. Selecting the model based on training performance (Experiment 1) led to choosing a severely overfitted baseline model with poor generalization (Test RMSE: 32,916). In contrast, using a validation set to guide the selection (Experiment 2) resulted in choosing a well-tuned, regularized model that generalized much better, achieving a significantly lower **Test RMSE of 27,158.536**.

3. K-Nearest Neighbors (KNN) Regressor with PCA

a. Preprocessing: For the KNN model, the standard preprocessing pipeline suitable for distance-based algorithms was used. This included all feature engineering steps (outlier removal, merging, ordinal mapping, time features, log transformations of specific predictors with original drops). Crucially, the final ColumnTransformer applied StandardScaler to all numerical features (essential for KNN and PCA) and used OneHotEncoder for the remaining categorical features.

b. Dimensionality Reduction (PCA): Principal Component Analysis (PCA) was incorporated into the pipeline *after* preprocessing but *before* the KNN regressor. The goal was to reduce the dimensionality of the feature space while retaining a significant portion of the variance, potentially improving KNN's performance and efficiency.

c. Hyperparameter Tuning: GridSearchCV was used to optimize hyperparameters for both the PCA step and the KNN regressor simultaneously.

- The grid searched included:pca__n_components: [0.95, 0.99, 50, 100] (Target variance retained or fixed number of components) * regressor__n_neighbors: [3, 5, 7, 10] (Number of neighbors) * regressor__weights: ['uniform', 'distance'] (Neighbor weighting) * regressor__metric: ['minkowski', 'manhattan'] (Distance calculation)
- A 5-fold cross-validation strategy was used within the grid search, optimizing for negative RMSE.
- The best parameters obtained were: {'pca__n_components': 0.95, 'regressor__metric': 'minkowski', 'regressor__n_neighbors': 10, 'regressor__weights': 'distance'}

Best CV Score (Negative RMSE): -0.1516 (corresponding to an average validation RMSE of roughly 0.1516 on the log scale during tuning).

PCA Result: The final model selected by GridSearchCV used PCA retaining 95% of the variance, which resulted in 54 principal components.

d. Training Performance: The best pipeline found by GridSearchCV (which was already trained on the full dataset during the search's final refit) was evaluated on the full 100% training dataset: Training RMSE = 217.05, Training R^2 = 1.0000

e. Test Set Performance: The Root Mean Squared Error (RMSE) obtained on the final, hidden test dataset for this model was 41,743.459. This relatively high test RMSE, compared to the near-perfect training score, confirms the overfitting observed

4. Linear Regression (Baseline) & Ridge Regression (with Bayesian Optimization + LOOCV)

Linear models, particularly Ridge Regression (Linear Regression with L2 regularization), were explored due to their interpretability and efficiency. Bayesian Optimization was used for hyperparameter tuning, and Leave-One-Out Cross-Validation (LOOCV) was employed for robust model evaluation and selection.

a. Preprocessing: For the linear models, the standard preprocessing pipeline suitable for these algorithms was used. This included all feature engineering steps (outlier removal, merging, extensive ordinal mapping, time features, log transformations of specific predictors with original drops). Crucially, the final ColumnTransformer applied StandardScaler to all numerical features (essential for regularized linear models) and used OneHotEncoder for the remaining categorical features to create a purely numerical input suitable for linear regression.

b. Hyperparameter Tuning (Ridge): BayesSearchCV (from skopt) was used to optimize the hyperparameters of the Ridge regression model.

- The search space included ranges for: alpha (Regularization strength): [1e-6, 10.0] (log-uniform) fit_intercept: [True, False] tol (Tolerance for stopping criteria): [1e-5, 1e-2] (log-uniform)
- A 5-fold cross-validation strategy was used internally within the Bayesian search over 30 iterations to find the best parameters.
- Best Parameters Found: {'regressor__alpha': 10.0, 'regressor__fit_intercept': True, 'regressor__tol': 0.01}

c. Model Evaluation using LOOCV: To obtain a highly reliable performance estimate, both the baseline LinearRegression model and the best Ridge model found by BayesSearchCV were evaluated using Leave-One-Out Cross-Validation (LeaveOneOut()) on the full training dataset (X_train_fe, y_train_log). The average RMSE across all LOOCV folds was calculated:

- Baseline Linear Regression: LOOCV RMSE = 0.08 (on the log scale)
- Bayesian Optimized Ridge Regression: LOOCV RMSE = 0.07 (on the log scale)

d. Final Model Selection & Prediction: Based on the LOOCV scores, the Bayesian-optimized Ridge Regression model demonstrated superior performance (lower average RMSE). Therefore, this optimized configuration (which BayesSearchCV automatically refits on the full training data after finding the best parameters) was selected as the final model.

e. Test Set Performance: The Root Mean Squared Error (RMSE) obtained on the final, hidden test dataset for this model was 18,168.114

5. AdaBoost Regressor with Bayesian Optimization

The model utilizes an AdaBoost Regressor with a Decision Tree Regressor as its weak learner, with hyperparameters optimized via Bayesian search. The target variable, HotelValue, was log(1+y) transformed.

a. Preprocessing and Feature Engineering:

- **Outlier Removal**: Samples with extreme HotelValue (outside the 0.1% to 99.9% quantiles) or potential data errors (e.g., UsableArea > 4000 or

OverallQuality < 3 in conjunction with other features) were removed from the training data.

- **Feature Engineering:**
  - Merging: Features were combined, creating TotalBasementScore, TotalPoolScore (quality times area), and TotalPorchArea.
  - Time-Based: HouseAge and YearsSinceModification were computed.
  - Ordinal Mapping: Many categorical features (e.g., ParkingQuality, ExteriorQuality, KitchenQuality) were manually mapped to numerical ordinal scores (e.g., 5-point scale).
  - Log Transformation: Skewed numerical features (e.g., LandArea, ParkingArea) were $\log(1+x)$ transformed.
- **Pipeline (ColumnTransformer):**
  - Numerical Features (including manually mapped ordinals): Imputed with the median and Standard Scaled.
  - Remaining Categorical Features: Imputed with 'None' and processed with OrdinalEncoder.

b. Hyperparameter Tuning (Bayesian Optimization):

- **Method:** BayesSearchCV (Bayesian Optimization) was used to tune the model over 30 iterations with 5-fold Cross-Validation.
- **Search Space:** The search included parameters for the AdaBoost ensemble and the base Decision Tree estimator:
  - regressor__n_estimators: Integer(50, 500)
  - regressor__learning_rate: Real(1e-3, 1.0, prior='log-uniform')
  - regressor__loss: Categorical(['linear', 'square', 'exponential'])
  - regressor__estimator__max_depth: Integer(2, 8)
- **Best Parameters:** The optimization found the best configuration to be n_estimators=500, learning_rate=1.0, loss='square', and a base learner max_depth=8.

c. Model Performance: The final model's performance on the training data and the final submission score are:

- **Bayesian Optimized AdaBoost RMSE (Training): 8,045.28**
- **Bayesian Optimized AdaBoost R^2 (Training): 0.9884**

The **Root Mean Squared Error (RMSE)** for the final submission **`adaboostBayesian.csv`** was **34,062.202**.

6. Linear Regression & Ridge Regression (with GridSearchCV)

Linear models, including standard LinearRegression and Ridge regression (Linear Regression with L2 regularization), were evaluated for their efficiency and interpretability. GridSearchCV was used to tune the alpha (regularization strength), fit_intercept, and tol hyperparameters for the Ridge model. Two distinct methodologies were used for training and final model selection:

a. Preprocessing (Common to Both Experiments): For both linear model experiments, the standard preprocessing pipeline was applied. This included all feature engineering steps (outlier removal, merging, extensive ordinal mapping, time features, log transformations of specific predictors with original drops, *excluding commented-out interaction features*). The final ColumnTransformer applied StandardScaler to all numerical features and used OneHotEncoder for the remaining categorical features, ensuring a purely numerical input suitable for linear regression.

b. Experiment 1: Model Selection via Validation Split

- **Setup:** The preprocessed training data was divided into an 85% training set (1012 samples) and a 15% validation set (179 samples).
- **Hyperparameter Tuning (Ridge):** GridSearchCV (with 5-fold CV) was performed on the 85% training split to find the optimal hyperparameters.
    - Best Parameters Found: {'alpha': 10.0, 'fit_intercept': True, 'tol': 0.0001}
- **Validation Performance:** Both models were trained on the 85% split and evaluated on the 15% validation set:
    - Baseline Linear Regression: Validation RMSE = 23,690.95
    - GridSearch Optimized Ridge: Validation RMSE = 22,515.23
- **Model Selection & Retraining:** Based on the superior performance (lower RMSE) on the validation set, the GridSearch-optimized Ridge Regression model was selected. This pipeline was then retrained on the entire 100% training dataset using the best hyperparameters found.
- **Test Set Performance (Experiment 1):** The final RMSE on the hidden test dataset for this model was 17,701.114.

c. Experiment 2: Model Selection via Training Score (Full Dataset)

- **Setup:** Both the Baseline Linear Regression and the GridSearchCV for Ridge were trained directly on the full 100% training dataset.
- **Hyperparameter Tuning (Ridge):** GridSearchCV (with 5-fold CV) was performed on the 100% training dataset.

    - Best Parameters Found: {'alpha': 10.0, 'fit_intercept': True, 'tol': 0.0001} *(Note: Same best parameters as Experiment 1 were found)*
- **Training Performance:**
    - Baseline Linear Regression: Training RMSE = 16,458.57
    - GridSearch Optimized Ridge: Training RMSE = 17,167.81
- **Model Selection:** Following the script's logic to select based purely on the training RMSE, the Baseline Linear Regression model was chosen, as it had the lower training error.
- **Test Set Performance (Experiment 2):** The final RMSE on the hidden test dataset for this baseline model was 17,424.897.

d. Conclusion for Linear Models: Both approaches yielded very strong and remarkably similar results on the final test set. Experiment 1, using a validation split for model selection, chose the regularized Ridge model, which is generally preferred for better generalization. Experiment 2, selecting based on the training score, chose the unregularized Baseline Linear Regression, which slightly overfit the training data (lower training RMSE) but, in this specific instance, happened to achieve a marginally better score on the hidden test set (Test RMSE: 17,424.897 vs. 17,701.114). The closeness of the results suggests that the extensive feature engineering might have already created a feature set where severe overfitting wasn't a major issue for the linear models.

7. Linear Regression & Ridge Regression (with Bayesian Optimization)

Linear models, including standard LinearRegression and Ridge regression (Linear Regression with L2 regularization), were evaluated for their efficiency and interpretability. Bayesian Optimization (BayesSearchCV from skopt) was used to tune the alpha, fit_intercept, and tol hyperparameters for the Ridge model. Two distinct methodologies were used for training and final model selection:

a. Preprocessing (Common to Both Experiments): For both linear model experiments, the standard preprocessing pipeline was applied. This included all feature engineering steps (outlier removal adjusted to only OverallQuality < 3, merging, extensive ordinal mapping, time features, log transformations of specific predictors like LandArea_Log with original columns dropped). In these runs, PropertyClass was dropped entirely, and interaction features like QualityArea were *not* created. The final ColumnTransformer applied StandardScaler to all numerical features and used OneHotEncoder for the remaining categorical features, ensuring a purely numerical input suitable for linear regression.

b. Experiment 1: Model Selection via Training Score (Full Dataset)

- **Hyperparameter Tuning (Ridge):** BayesSearchCV (with 5-fold CV, 30 iterations) was performed directly on the full 100% training dataset.
    - Best Parameters Found: {'alpha': 10.0, 'fit_intercept': True, 'tol': 0.01}
- **Training Performance:**
    - Baseline Linear Regression: Training RMSE = 16,459.03, Training R^2 = 0.9513
    - Bayesian Optimized Ridge: Training RMSE = 17,159.73, Training R^2 = 0.9470
- **Model Selection:** Based strictly on the lower training RMSE, the Baseline Linear Regression model was selected for the final prediction in this experiment.
- **Test Set Performance (Experiment 1):** The final RMSE on the hidden test dataset for this selected baseline model was 17,270.991.

c. Experiment 2: Model Selection via Validation Split

- **Setup:** The preprocessed training data was divided into an 85% training set (1012 samples) and a 15% validation set (179 samples) using train_test_split.
- **Hyperparameter Tuning (Ridge):** BayesSearchCV (with 5-fold CV, 30 iterations) was performed on the 85% training split.
    - Best Parameters Found: {'alpha': 10.0, 'fit_intercept': True, 'tol': 0.01} *(Note: Same best parameters as Experiment 1 were found)*
- **Validation Performance:** Both models were trained on the 85% split and evaluated on the 15% validation set:
    - Baseline Linear Regression: Validation RMSE = 23,610.57, Validation R^2 = 0.8989
    - Bayesian Optimized Ridge: Validation RMSE = 22,493.43, Validation R^2 = 0.9082
- **Model Selection & Retraining**: Based on the superior performance (lower RMSE) on the validation set, the Bayesian-optimized Ridge Regression model was selected. This pipeline was then retrained on the entire 100% training dataset using the best hyperparameters found.
- **Test Set Performance (Experiment 2):** The final RMSE on the hidden test dataset for this model was 19,208.920.

d. Conclusion for Linear Models:

Both approaches yielded strong results, with the model selected via training score (Experiment 1, Baseline LR) achieving a slightly lower final test RMSE (17,270.991) compared to the model selected via validation score (Experiment 2, Optimized Ridge, Test RMSE 19,208.920).This suggests the regularization provided by Ridge,

while helpful on the validation set, was slightly less optimal for the final test data distribution compared to the unregularized model, possibly due to the effectiveness of the extensive feature engineering already performed.

8. Linear Regression & Ridge Regression (No Log Transforms)

To assess the impact of log transformations, a separate experiment was conducted using the same Linear Regression and Ridge Regression models (tuned with Bayesian Optimization) but without applying np.log1p to the target variable (HotelValue) or the specific predictor features (like LandArea, ParkingArea, etc.). Model selection was based on performance on the full training dataset.

a. Preprocessing & Feature Engineering: The preprocessing pipeline was identical to the previous linear model experiments (outlier removal, merging, ordinal mapping, time features, dropping PropertyClass), with the key exception that no log transformations were applied to the target variable or the specified predictor features. StandardScaler was still applied to all numerical features, and OneHotEncoder was used for remaining categoricals.

b. Hyperparameter Tuning : BayesSearchCV was employed to find the optimal hyperparameters for the Ridge model.

- Tuning Strategy: A 5-fold cross-validation Bayesian search was conducted over 30 iterations for alpha, fit_intercept, and tol.
- Best Parameters Found: {'regressor__alpha': 10.0, 'regressor__fit_intercept': True, 'regressor__tol': 1e-05}

c. Training Performance: Both models were trained and evaluated on the full 100% training dataset using the original HotelValue scale:

- Baseline Linear Regression: Training RMSE = 18,683.44, Training R^2 = 0.9372
- Bayesian Optimized Ridge Regression: Training RMSE = 19,492.58, Training R^2 = 0.9317

Based purely on the training scores, the Baseline Linear Regression model had a lower RMSE. Following the script's logic, this model was chosen for the final prediction.

e. Test Set Performance: The Root Mean Squared Error (RMSE) obtained on the final, hidden test dataset for this model was 26,967.058.

Conclusion: Comparing this result (Test RMSE: 26,967.058) to the best result achieved with log transformations (Test RMSE: 17,270.991), it is evident that

applying the log transformations to both the target variable and the skewed features significantly improved the performance of the linear models for this dataset. Training directly on the original scale resulted in a considerably higher error on the test set

9. LightBGM with Bayesian Optimization

Light Gradient Boosting Machine (LightGBM) models were explored due to their strong performance on tabular data, ability to capture non-linear relationships, and efficiency with mixed feature types. Both a baseline model and a Bayesian-optimized model were trained and compared for performance and stability.

a. Preprocessing: The preprocessing pipeline for LightGBM followed a comprehensive feature engineering workflow. Outlier removal was first performed, removing 6 extreme cases from the initial dataset (resulting in 1,194 training samples). Feature engineering steps included merging and aggregating basement-related features, engineering new pool-related attributes, and combining multiple porch features into unified representations. The final dataset consisted of 72 features (45 numerical and 27 categorical). Since LightGBM natively handles categorical features, these were label-encoded rather than one-hot encoded, and numerical features were left unscaled to preserve their natural distributions. The resulting clean training set had a shape of (1194, 72), and the test set had (260, 72).

b. Baseline Model (LightGBM): A baseline LightGBM model was trained using default parameters optimized for stability and interpretability. This model achieved excellent performance on the training data, with:

• RMSE (Train): 660.19 • R^2 (Train): 0.9999

These results indicated near-perfect fitting on the training data, suggesting a strong ability to capture underlying patterns even before hyperparameter tuning.

c. Bayesian Hyperparameter Tuning: To further optimize performance, Bayesian Optimization was employed to fine-tune key LightGBM hyperparameters, focusing on parameters controlling tree complexity and regularization. The search space included:

- `num_leaves`: [15, 50]

- `max_depth`: [3, 10]
- `learning_rate`: [0.01, 0.1]
- `n_estimators`: [500, 2000]
- `colsample_bytree`, `subsample`: [0.5, 1.0]
- `reg_alpha`, `reg_lambda`: [0.0, 1.0]
- `min_child_samples`: [10, 30]

After iterative Bayesian search, the best parameter configuration was found to be:

{ 'colsample_bytree': 0.8890, 'learning_rate': 0.0329,'max_depth': 4, 'min_child_samples': 17, 'n_estimators': 1475, 'num_leaves': 31, 'reg_alpha': 0.5603, 'reg_lambda': 0.0748, 'subsample': 0.7152}

d. Model Evaluation: The Bayesian-optimized LightGBM model achieved the following results on the training data:

- RMSE (Train): 9,470.63
- R^2 (Train): 0.9839

While the optimized model generalized better (showing less overfitting), its training RMSE was significantly higher than that of the baseline, indicating that the baseline configuration better captured the training data distribution without excessive regularization.

e. Test Set Performance: The root Mean Squared Error (RMSE) obtained on the final, hidden test dataset for this model was 32968.185.

f. Final Model Selection & Prediction: Based on comparative performance, the baseline LightGBM model was selected as the final model for prediction. Despite the Bayesian optimization improving generalization control, the baseline configuration demonstrated superior or equal overall performance on the training dataset and was thus retained as the preferred model for final evaluation and test set prediction.

10. Bayesian Ridge Regression (Conjugate Priors)

This model employs **Bayesian Ridge Regression**, a linear model that estimates the regularization parameters (alpha and lambda) directly from the data using a Bayesian approach with conjugate priors. The target variable, **HotelValue**, was log(1+y) transformed.

a. Preprocessing and Feature Engineering: The preprocessing was tailored for a linear model, emphasizing numerical inputs and scaling.

- **Outlier Removal:** Outliers were removed based on the target value (0.1% to 99.9% quantiles) and specific feature extremes (e.g., UsableArea > 4000).
- **Feature Engineering:**
  - Merging: Features like basement, pool, and porch areas were merged to create aggregate scores (e.g., TotalBasementScore, TotalPoolScore, TotalPorchArea).
  - Time-Based: HouseAge and YearsSinceModification were calculated.
  - Ordinal Mapping: Key categorical features (e.g., ParkingQuality, KitchenQuality, BasementExposure) were manually converted to numerical ordinal scores.
  - Log Transformation: Skewed numerical features (e.g., LandArea, ParkingArea) were log(1+x) transformed.
- **Final Pipeline (ColumnTransformer):**
  - Numerical Features (including manually mapped ordinals): Imputed with the median and Standard Scaled (essential for regularized linear models).
  - Remaining Categorical Features: Imputed with 'None' and processed using OneHotEncoder to create a purely numerical, high-dimensional input for the linear model.

b. Model Training and Parameter Estimation:

- **Model**: BayesianRidge was used, which automatically estimates the precision of the weights (lambda, corresponding to regularization) and the precision of the noise (*alpha*).
- **Estimated Parameters:** The model was trained on the full log-transformed training set.
  - Estimated alpha (Noise Precision): 105.690320
  - Estimated lambda (Weight Precision): 1498.390584
- **Model Selection:** Although the Baseline Linear Regression model achieved a slightly lower training RMSE (16,411.19) than the Bayesian Ridge Regression model (17,288.09), the Bayesian Ridge model was ultimately selected as the final model. The Bayesian model inherently provides regularization and uncertainty estimates, which often leads to better generalization performance despite a slightly higher training error.

c. Model Performance Scores:

- Bayesian Ridge Regression RMSE (Training): 17,288.09
- Bayesian Ridge Regression R^2 (Training): 0.9465

The Root Mean Squared Error (RMSE) obtained on the final test dataset for the submission Bayesian_ConjugatePriors.csv was 17,856.545.

V. Model Performance Comparison

The following table summarizes the performance results obtained from the various regression models and hyperparameter tuning strategies explored in this project. The primary metric used for comparison is the Root Mean Squared Error (RMSE) achieved on the final hidden test dataset, reflecting the model's ability to generalize to unseen data.

Model Approach	Pre-processing	Tuning Method	Selection Method	RMSE (Validation / Train)	Final Test RMSE	Notes
Linear + Bayes (Best Overall)	Linear	BayesSearch	Training Score	16,459.03 (Train)	17,270.991	Selected Baseline LR
Linear + Grid (Full Train)	Linear	GridSearch	Training Score	16,458.57 (Train)	17,424.897	Selected Baseline LR
Ridge + Grid	Linear	GridSearch	Validation Score	22,515.23 / (16,143.87)*	17,701.114	Selected Optimized Ridge, retrained

(Validation Split)						
Ridge + Bayes (LOOCV Eval)	Linear	BayesSearch	LOOCV Score (0.07)	(LOOCV: 0.07)	18,168.114	Selected Optimized Ridge
Ridge + Bayes (Validation Split)	Linear	BayesSearch	Validation Score	22,493.43 / (16,142.26)*	19,208.920	Selected Optimized Ridge, retrained
Linear + Bayes (No Log Transform)	Linear	BayesSearch	Training Score	18,683.44 (Train)	26,967.058	Selected Baseline LR, Logs clearly helped
XGBoost + Bayes (Validation Split)	Tree	BayesSearch	Validation Score	22,940.15 (Val)	27,158.536	Selected Optimized XGB, retrained (Best Tree)
XGBoost + Bayes (Training Score)	Tree	BayesSearch	Training Score	855.31 (Train)	32,916.281	Selected Baseline XGB (Overfit)

LightGBM + Bayes	Tree	BayesSearch	Training Score	660.19 (Train)	32,968.185	Selected Baseline LGBM (Overfit)
Random Forest + Bayes	Tree	BayesSearch	Validation Score	29,456.75 (Val)	33,663.477	Selected Optimized RF, retrained
AdaBoost + Bayes	Tree	BayesSearch	(Implied Optimized)	8,045.28 (Train)	34,062.202	Tuned AdaBoost
Random Forest + Grid	Tree	GridSearch	Validation Score	29,938.99 (Val)	34,168.219	Selected Optimized RF, retrained
KNN + PCA + Grid	Linear (+ PCA)	GridSearch	CV Score (-0.15)	217.05 (Train)	41,743.459	Selected Optimized KNN (Overfit)
Bayesian Ridge (Conjugate Priors)	Linear	None (Internal)	Implicit	17,288.09 (Train)	*(Not Tested)*	Selected over Baseline LR in its run

---

VI. Final Model Selection & Rationale

The final model was selected by comparing the Root Mean Squared Error (RMSE) achieved on the hidden test dataset across the various models and tuning strategies explored.

After comprehensive feature engineering, target transformation (using `log1p`), and preprocessing tailored for linear models, the **Baseline Linear Regression** model achieved the **lowest overall Test RMSE of 17,270.991**. This model was selected based on its superior training score when compared directly against a Bayesian-optimized Ridge model within the same script run.

Interestingly, this unregularized baseline model also slightly outperformed Ridge Regression models selected using more robust methods like validation splits (Test RMSEs of 17,701.114 and 19,208.920) or LOOCV evaluation (Test RMSE 18,168.114). While using a validation split or LOOCV is generally preferred for estimating generalization performance, the baseline model selected via training score happened to yield the best result on this specific test set. This highlights the effectiveness of the feature engineering steps, which likely created a feature space highly suitable for a simple linear model. The crucial role of log transformations was confirmed, as removing them significantly worsened performance (Test RMSE 26,967.058).

More complex models like XGBoost (optimized using Bayesian search and selected via validation split, Test RMSE 27,158.536) and others like Random Forest, AdaBoost, and KNN showed considerably higher test errors.

Given its superior performance on the final test metric, the **Baseline Linear Regression model (trained on the full dataset after log transformations and feature engineering, selected via training score comparison)** was identified as the most effective model for this specific task and dataset.

VII. Discussion & Analysis

Key Observations:

- Carefully engineered numerical and ordinal encodings significantly improved linear model performance.
- Tree-based models (Random Forest, XGBoost) provided competitive performance, but tuning and regularization were essential to prevent overfitting.
- Log-transformation of the target variable and skewed predictors was crucial for model stability.

- LOOCV on select linear models indicated small differences in RMSE, guiding the final selection.

Limitations:

- Some model runs showed suspiciously low training RMSE, suggesting potential overfitting, which warrants further investigation. More robust cross-validation and a held-out test set could mitigate this risk.
- The dataset contains domain-specific columns; incorporating external market features or geolocation data could enhance prediction accuracy.

Interesting Observations:

- **Linear Models Unexpectedly Dominate:** The most striking finding is the superior performance of the Linear and Ridge Regression models (Test RMSEs ~17k-19k) compared to more complex tree-based ensembles like XGBoost, Random Forest, and LightGBM (Test RMSEs ~27k-34k) and distance-based KNN (Test RMSE ~41k). This strongly suggests that the extensive feature engineering was highly effective, creating a feature space where linear relationships capture the majority of the variance in HotelValue.
- **Feature Engineering is King:** The success of the linear models heavily underscores the power of meticulous feature engineering. Steps like merging related features (Basement, Pool, Porch), carefully mapping ordinal variables (Qualities, Conditions, Finish), and targeted log transformations likely linearized relationships and extracted crucial information, making the problem more amenable to simpler models.
- **Log Transformations Proved Crucial:** The dedicated experiment running Linear/Ridge models without log transformations resulted in a significantly worse Test RMSE (26,967) compared to the best run with log transformations (17,271). This empirically confirms the necessity and effectiveness of applying log1p to both the target variable and selected skewed features for linear models in this specific dataset.
- **Overfitting Proneness of Complex Models:** Tree-based models (especially XGBoost and LightGBM baselines) and KNN showed dramatic signs of overfitting, achieving near-perfect scores on the training data (RMSE < 1000, $R^2 \approx 1.0$) but generalizing poorly to the test set (RMSE > 32k).
- **Effectiveness of Bayesian Optimization:** Across experiments where both GridSearch and Bayesian Optimization were used (Random Forest), or where Bayesian Optimization significantly improved over baseline (XGBoost validation run), Bayesian Optimization (BayesSearchCV) consistently found better hyperparameter combinations, leading to improved validation and test scores compared to defaults or limited grid searches.