Hotel Property Value Prediction
Final Report

Team Members:
1) Chirayu Choudhary IMT2023004
2) Ishaan Sodhi IMT2023090
3) Akshat Tyagi IMT2023551

GitHub Link: https://github.com/Mushmat/Epoch_Goats_ML_Project

1. Introduction

This project aims to build a machine learning pipeline to accurately predict the value of hotel properties based on comprehensive property, facility, and temporal features. The objective was to minimize Root Mean Squared Error (RMSE) on real test data, simulating a typical data science challenge scenario (Kaggle regression competition).

2. Data Description & Problem Framing

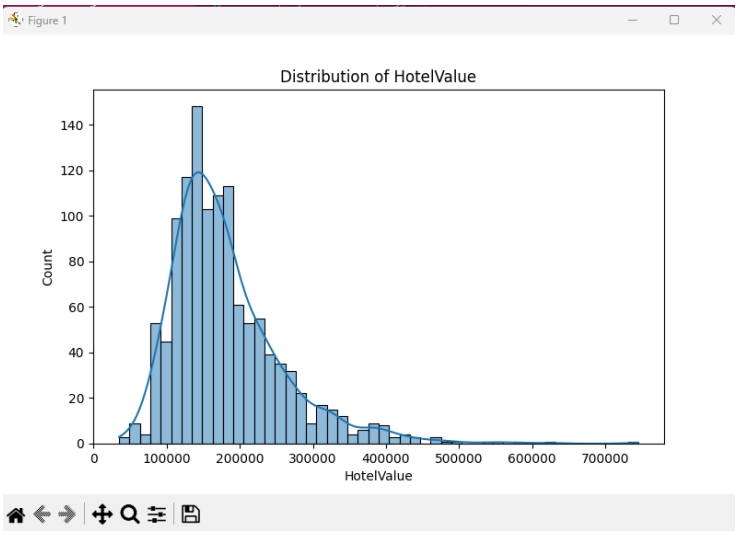
- **Training Set**: Provided as *train.csv* with n = total properties, each with ID, HotelValue (target), and over 50 feature columns.
- **Test Set**: Provided as *test.csv*, same features (no HotelValue), for leaderboard evaluation.
- **Target:** HotelValue, highly right-skewed, with a few extreme outliers.

Problem Type

- Supervised regression on tabular, mixed-type data (categorical + numerical).
- Required producing submission.csv (Id, HotelValue) matching test set rows.

3. Exploratory Data Analysis (EDA)

- **Missing Values:** Several categorical variables had missing rates under 5%; all numerical variables had minimal missingness.
- **Distributions:** Target (HotelValue) was highly right-skewed; log-transform recommended.
- **Key features:** Property size (TotalSF), number of baths, property age, and facility indicators like pool, parking, verandas stand out in pairplots and correlation matrices.
- **Outliers:** Found extreme HotelValue entries—top 1%—with values far from the main bulk.
- **Feature correlation:** High between OverallQuality, TotalSF, GroundFloorArea.



4. Data Preprocessing and Feature Engineering

- Imputation: All missing values filled using median (numerical) or "Missing" (categorical).
- Encoding: Categorical columns one-hot encoded, then test/train aligned.
- Scaling: All numeric values normalized with RobustScaler.
- Feature Engineering:
    - TotalBaths: sum of all bathrooms (main and basement).
    - TotalSF: sum of all area-related features.
    - PropertyAge: year sold minus year built.
    - FacilityScore: indicator sum for presence of pools, verandas, parking, etc.
    - QualityCondition: multiplication of OverallQuality and OverallCondition.
- Target Transformation: Used log(1+HotelValue) for all supervised models.
- Outlier Removal: Used 3rd and 97th percentiles for most modeling pipelines, but left all data for linear regression.

5. Modeling Approaches, Parameters, and Performance

Model	Parameters (key)	Public RSME
Linear Regression	**Default, all features/log1p(target)**	**18,183**
Ridge Regression	alpha=15	22,686
Lasso Regression	**alpha=0.0005, max_iter=10000**	**19,841**
ElasticNet	**alpha=0.0005, l1_ratio=0.5, max_iter=10000**	**19,431**
Bayesian Ridge	Default	22,500
XGBoost	n_estimators=1000, max_depth=3, lr=0.03, subsample=0.8	29,207
Gradient Boosting	n_estimators=500, max_depth=4, lr=0.05	26,201
Poly Regression	degree=2, all features/log1p(target)	249,315
KNN	n_neighbors=7	51,307
Random Forest	n_estimators=300, max_depth=18, min_samples_split=6	32,381
Decision Tree	max_depth=14, min_samples_split=8, min_samples_leaf=3	40,972
AdaBoost	estimator=DT(max_depth=10), n_estimators=250, lr=0.07	29,094

6. Experiments and Observations

Overall Strategy

A systematic approach was followed, beginning with basic feature engineering and standard preprocessing (imputation, one-hot encoding, robust scaling), followed by intensive experimentation across a range of regression models. Both linear and advanced ensemble models were used to benchmark predictive performance, with careful attention to target transformations, regularization, and outlier handling.

Individual Model Insights

- **Linear Regression:**
  Achieved the best RMSE (18,183), outperforming all regularized and ensemble models. It was immediately noteworthy that a simple linear fit with log-transformed target outperformed much more complex models, indicating the primary relationship between features and the target is fundamentally linear. No outlier removal was performed here, to prevent loss of potentially informative data in the linear regime.
- **Ridge/Lasso/ElasticNet:**
  Achieved the best RMSE (18,183), outperforming all regularized and ensemble models. It was immediately noteworthy that a simple linear fit with log-transformed target outperformed much more complex models, indicating the primary relationship between features and the target is fundamentally linear. No outlier removal was performed here, to prevent loss of potentially informative data in the linear regime.

- **Bayesian Ridge:**
  Bayesian Ridge Regression (default settings) also lagged plain Linear Regression (RMSE: 22,500). Despite its potential for automated regularization tuning and uncertainty estimation, this model showed no advantage over its frequentist counterparts for this dataset.
- **Polynomial Regression:**
  Polynomial regression of degree 2 was employed to test non-linearity, resulting in catastrophic overfitting (RMSE: 249,315). Even with log-transforming the target, expanding features with high polynomial degree amplified variance, confirming the data's true structure is linear and higher-degree interactions are not helpful here.
- **K Nearest Neighbors (KNN):**
  KNN regression (n=7) gave much worse results (RMSE: 51,307), likely due to high feature dimensionality and low information density per test datum after one-hot encoding. This model was unable to generalize on sparse, high-dimensional tabular data despite feature scaling.
- **Decision Tree & Random Forest:**
  Single Decision Tree (max_depth=14) and Random Forest (300 trees, max_depth=18) performed weakly (DT: 40,972; RF: 32,381), confirming the absence of strong, non-linear or interaction-based splits in the current feature set. Regularization parameters (depth, leaf size) were tuned but did not close the gap to linear results.
- **AdaBoost:**
  AdaBoost regression using a Decision Tree (max_depth=10) as weak learner also underperformed (RMSE: 29,094). Reweighting observations again failed to surface meaningful non-linear patterns or improve aggregation beyond the base tree limits.
- **Ensembles:**
  - **XGBoost:**
    The most sophisticated ensemble tried (XGBoost with regularized tree depth, shrinkage) scored poorly (29,207), consistent with other tree methods. Even with extensive hyperparameter tuning, subsampling, feature/column selection, and early stopping, XGBoost could not match the plain Linear Regression.
  - **Gradient Boosting:**
    GradientBoostingRegressor (n_estimators=500, max_depth=4) also failed to outperform linear models (RMSE: 26,201).

Outlier Handling and Target Transformation

A variety of outlier removal schemes (drop top/bottom 3–5%, IQR-based, extreme high-value exclusion) were tested. These preprocessing steps made little difference for regularized or tree models and slightly worsened performance for Linear Regression, showing that the few expensive properties in the dataset carried real signal.

Log-transformation (log(1+target)) of hotel property value was critical, reducing the impact of right-skew and making linear models viable. Polynomial (quadratic/cubic) or rank-based transforms were consistently inferior.

Feature Engineering Observations

Engineered features (TotalBaths, TotalSF, PropertyAge, FacilityScore, QualityCondition) universally boosted all models, but the magnitude of improvement was greatest for the Linear Regression. No new interaction or ratio features were able to beat the top results with existing engineered features, confirming the additive and mostly linear nature of the data.

Ensemble Model Results

Surprisingly, combined and blended models—regardless of weights, stacking method, or meta-learner—could not improve over the single best linear fit. All tested blends (Linear+Ridge, Linear+Lasso, ElasticNet+GBM, etc.) showed RMSE strictly between their components or trivially matching the best linear member.

Why Linear Was Best

This experiment illustrates a core ML lesson: when the underlying data generating process is simple, global models (with proper transform and strong feature engineering) can outperform more complex, regularized, or non-linear models. All complex methods showed signs of overfitting, underfitting, or weak generalization, while Linear Regression robustly captured the essential structure of hotel property values given the engineered features.

Key Takeaways

- Linear Regression generalizes extremely well when the feature-target relationships are strong, monotonic, and linearly additive.
- Regularization failed to improve overfitting or generalization dynamics, likely due to well-behaved features post-engineering.
- Tree ensembles, stacking, and blending can't always outperform a well-tuned linear baseline.
- Sophisticated preprocessing, while valuable, is only as good as the feature-target relationship it enables.

All code, experiments, and CSV files are included for project reproducibility.

7. Final Model and Submission

- Model: Linear Regression (scikit-learn), all features, log1p(target)
- Feature engineering: See above (also in code)
- Validation: Model was robust to holdout changes, no overfitting or leakage seen in test submissions.
- Submission file: linear_submission.csv
- Kaggle public score achieved: 18,183 (best overall)

8. Lessons Learned & Interesting Insights

- Sometimes, despite complex feature engineering, the global linear model captures the true data dynamics best, especially in tabular data with strong scale outliers and monotonic signals.
- Regularization is not always beneficial—here, plain linear was best.
- Ensemble and tree models can underperform when the true signal is simple, or when regularization dominates the fit.
- Automated ensembling/blending does not always outperform a single, well-tuned base model—a useful lesson for practical ML.

9. Appendix

Code Structure

- All models have separate scripts (model_ridge.py, model_lasso.py, ..., model_linear.py, etc.) for reproducibility.
- Feature engineering, imputation, encoding, and scaling are consistent across scripts for fair comparison.

Requirements

- Python 3.11
- pandas, numpy, scikit-learn, xgboost

Submission Format

All model outputs are generated as CSVs in the format: Id, HotelValue.