ML Project Report: Hotel Property Value Prediction

Team Members

1. M S Dheeraj Murthy - imt2023552
2. Ayush Tiwari - imt2023524
3. Mathew Joseph - imt2023008

Github Link: https://github.com/Dheeraj-Murthy/ML-project

Task

The primary task of this project is to build a machine learning model to accurately predict the value of hotel properties. This is a regression problem where the target variable is the `HotelValue`. The project involves exploring the dataset, performing necessary preprocessing, training various regression models, and evaluating their performance to find the best model for this prediction task.

Dataset and Features Description

The dataset contains various features related to hotel properties. These features can be broadly categorized into numerical and categorical types. The dataset is split into `train.csv` and `test.csv` files. The training data contains 1200 rows and 81 columns, including the target variable `HotelValue`. The test data is used for final evaluation. The features include information about the property's size, location, quality, condition, and various amenities.

EDA and pre-processing

The following section outlines the necessary steps for conducting essential Exploratory Data Analysis (EDA) to gain insights, detect potential issues, and prepare data for model training.

1. Import Libraries and Load Dataset

The initial step in the EDA process involves importing essential libraries such as pandas, numpy, matplotlib, seaborn, and relevant modules from scikit-learn for data preprocessing. The dataset is loaded into a pandas DataFrame for further analysis and manipulation.

2. Data Overview

The dataset is examined by displaying the first few rows, checking data types, and identifying any missing values. This step helps in understanding the structure and initial quality of the data. A statistical summary of numerical columns is also obtained to review the central tendency, spread, and potential outliers.

3. Handling Missing Values

The dataset was examined for missing values, and it was found that several columns have a significant number of missing entries. Columns with a high percentage of missing values (e.g., `PoolQuality`, `ExtraFacility`, `ServiceLaneType`) were dropped. For other columns with fewer missing values, imputation was performed using the median for numerical features and a constant value ('None') for categorical features.

4. Handling Duplicate Values

The dataset was checked for duplicate rows, and none were found. Therefore, no action was required for handling duplicates.

5. Exploratory Data Analysis (EDA)

Various plots were generated to analyse the data distribution, detect outliers, and explore relationships among features. These visualisations provide critical insights that inform subsequent preprocessing and modelling decisions.

Distribution Analysis of Numerical Features:

Histograms and Q-Q plots were used to analyze the distribution of numerical features. It was observed that many features are skewed. The target variable, `HotelValue`, is also right-skewed, so a log transformation (`np.log1p`) was applied to make its distribution more normal.

Outlier Detection:

Boxplots were used to identify outliers in numerical features. Outliers were found in several columns, including `LandArea` and `UsableArea`. A function `remove_outliers` was implemented to handle these outliers by removing data points based on domain knowledge (e.g., large land area with low hotel value) and the Interquartile Range (IQR) method.

Correlation Analysis:

A correlation matrix heatmap was generated for numerical features to understand their relationships. It was found that `OverallQuality` has a high positive correlation with `HotelValue` (0.79). Other features like `UsableArea`, `BasementTotalSF`, and `ParkingArea` also show a significant positive correlation with the target variable. Some features were also found to be highly correlated with each other, such as `ParkingCapacity` and `ParkingArea` (0.88).

6. Data Preprocessing

In this stage, the data is prepared for modelling through the following steps:

- **Feature Engineering**: New features were created to capture more information.

New Feature	Description
`TotalOutdoorArea`	Sum of terrace, veranda, and porch areas
`TotalSF`	Sum of ground floor, upper floor, parking, and outdoor areas
`TotalBaths`	Sum of full and half baths (including basement)
`OverallScore`	Average of `OverallQuality` and `OverallCondition`
`Age`	`YearSold - ConstructionYear`
`YearsSinceRemodel`	`YearSold - RenovationYear`

- **Encoding Categorical Variables**: Ordinal categorical features were encoded using a custom ordinal encoder, and nominal categorical features were one-hot encoded.
- **Scaling Numerical Features**: Numerical features were scaled using `StandardScaler` to bring them to a similar scale.
- **Target Transformation**: The target variable `HotelValue` was log-transformed.

A `ColumnTransformer` pipeline was created to apply these preprocessing steps consistently to the training and test data. The fitted preprocessor was saved to a `fitted_preprocessor.joblib` file.

7. Splitting the Dataset

The dataset is split into training and testing sets to evaluate the model's performance on unseen data.

Models Used For Training

Several regression models were trained to predict the hotel property value. A brief explanation of each model and its performance is provided below.

Linear Models

These models assume a linear relationship between the features and the target variable.

- **Linear Regression:** This is the most basic regression model and serves as a good baseline to compare other models against.
- **Ridge, Lasso, and ElasticNet Regression:** These are all types of regularized linear

regression. They are used to prevent overfitting by adding a penalty term to the loss function. Ridge uses L2 regularization, Lasso uses L1 regularization, and ElasticNet is a combination of both. These were tried to see if regularization would improve the performance of the basic linear model.
- **Bayesian Ridge Regression:** This is a probabilistic model that includes regularization. Unlike the standard Ridge regression, it can determine the regularization parameter by itself. It's a good alternative to Ridge when we are not sure about the best regularization strength.

Non-Linear Models

These models can capture more complex, non-linear relationships in the data.

- **Polynomial Regression:** This model can capture non-linear relationships between features and the target by fitting a polynomial equation to the data. It was tried to see if a non-linear model would perform better than the linear ones.
- **Gaussian Process Regressor:** This is a non-parametric, Bayesian approach to regression. It can provide uncertainty estimates for the predictions, which can be useful. It was tried as a different type of non-linear model.

Ensemble Models

These models combine the predictions of multiple individual models to produce a more accurate and robust prediction.

- **Random Forest Regressor:** This is an ensemble model that uses multiple decision trees to make a prediction. It is a powerful model that can capture complex non-linear relationships and is robust to overfitting.
- **AdaBoost Regressor:** This is another ensemble model that combines multiple weak learners (typically decision trees) to create a strong learner. It was tried to see if a different type of boosting algorithm would perform well.
- **XGBoost, CatBoost, and LightGBM Regressors:** These are all gradient boosting models, which are known for their high performance and efficiency. They build trees one at a time, where each new tree helps to correct errors made by previously trained trees. They were tried as they are often the state-of-the-art for tabular data.

Model Performance Summary

The performance of each model was evaluated using the Root Mean Squared Error (RMSE) on a validation set. The results are summarized in the table below, from best to worst performing.

Rank	Model	RMSE
1	Ridge Regression	20071.41
2	Bayesian Ridge Regression	20425.49

Rank	Model	RMSE
3	Lasso Regression	21315.94
4	Linear Regression	21327.46
5	CatBoost Regressor	26503.93
6	ElasticNet	26671.33
7	LGBM Regressor	27240.31
8	Gaussian Process Regressor	28094.36
9	Random Forest Regressor	29172.59
10	XGBoost Regressor	29296.00
11	Polynomial Regression	30305.02
12	AdaBoost Regressor	33193.79

Discussion on the Performance of Different Approaches

The results show that the regularized linear models, particularly **Ridge Regression**, performed the best on this dataset. Ridge Regression achieved the lowest RMSE of **20071.41**, outperforming more complex ensemble methods like XGBoost and Random Forest.

This suggests that the relationships between the features and the target variable in this dataset may be largely linear, and the regularization provided by the Ridge model was effective in preventing overfitting and improving generalization.

The more complex models like XGBoost and CatBoost did not perform as well, which could be due to several factors. It's possible that with this particular dataset, the additional complexity of these models led to overfitting, or that the hyperparameter tuning was not sufficient to find an optimal set of parameters for these models. The performance of the XGBoost model from `xgboost_model.py` (RMSE: 29296.00) was not competitive with the simpler linear models.

This project highlights an important lesson in machine learning: more complex models are not always better. For this particular problem, a well-regularized linear model proved to be the most effective solution.

Interesting Observations

- **Linear Models Outperforming Ensemble Models:** Perhaps the most interesting observation is that the regularized linear models, particularly Ridge Regression, outperformed the more complex ensemble models like XGBoost, CatBoost, and Random Forest. This is a somewhat

surprising result, as ensemble models are often expected to perform better on complex, high-dimensional datasets. This suggests that for this particular problem, a simpler, well-regularized linear model was more effective at capturing the underlying patterns without overfitting.

- **The Importance of Regularization:** Within the linear models, the regularized versions (Ridge and Lasso) performed better than the basic Linear Regression. This highlights the importance of regularization in preventing overfitting and improving the generalization of the model.
- **Feature Engineering:** The creation of new features like `TotalSF` and `Age` likely provided all the models with more meaningful information, contributing to better predictions across the board.

References

- XGBoost Documentation
- Scikit-learn Documentation
- Pandas Documentation