**Bangalore House Price Predictor Model**

**TABLE OF CONTENTS**

1. Introduction and Overview
2. Data Preprocessing and Feature Engineering
   * 2.1 Loading the Dataset
   * 2.2 Initial Data Analysis
   * 2.3 Exploratory Data Analysis
   * 2.4 Feature Engineering
   * 2.5 Handling Missing Values
   * 2.6 Feature Selection
   * 2.7 Encoding and Scaling
3. Model Selection and Optimization Approach
   * 3.1 Model Selection
   * 3.2 Pipeline Construction
   * 3.3 Train-Test Split
   * 3.4 Evaluation
   * 3.5 Optimization
4. Deployment Strategy and API Usage Guide
   * 4.1 Model Serialization
   * 4.2 Deployment Strategy
   * 4.3 API Usage Guide
5. Conclusion

**PROJECT REPORT**

**1. Introduction and Overview**

**Introduction**

This project develops a machine learning model to estimate house prices in Bangalore based on key factors like location, size, and amenities. Using historical data (Bengaluru\_House\_Data.csv), the model helps buyers and sellers make informed decisions.

**Project Overview**

1. **Data Preprocessing**: Cleaned missing values, converted size to bhk, and standardized total\_sqft.
2. **Feature Engineering**: Selected key features (location, total\_sqft, bath, bhk), applied one-hot encoding, and scaled numerical data.
3. **Model Selection**: Ridge regression (alpha=1) was chosen for its balance between accuracy and generalization, achieving R² = 0.914.
4. **Deployment**: The trained model was saved (RidgeModel.pkl) and prepared for API deployment via Flask.

**Significance**

With strong predictive performance, this project provides a practical tool for house price estimation in Bangalore, making real estate decisions more data-driven and efficient.

**2. Steps Taken for Data Preprocessing and Feature Engineering**

The data preprocessing and feature engineering steps were critical to preparing the Bangalore house dataset (Bengaluru\_House\_Data.csv) for modeling. The following steps were undertaken:

**2.1 Loading the Dataset**

* The dataset was loaded using pandas (pd.read\_csv), containing 13,320 rows and 9 columns: area\_type, availability, location, size, society, total\_sqft, bath, balcony, and price.

**2.2 Initial Data Analysis**

* The dataset’s shape, structure (df.info()), missing values (df.isnull().sum()), and basic statistics (df.describe()) were analyzed.
* Missing values were identified: location (1), size (16), society (5,502), bath (73), and balcony (609).
* Duplicates (529 rows) and categorical feature distributions (e.g., area\_type.value\_counts()) were checked.

**2.3 Exploratory Data Analysis (EDA)**

* Bar plots visualized the distribution of area\_type, revealing "Super built-up Area" as the most frequent type.
* This step likely guided feature selection by identifying key variables.

**2.4 Feature Engineering**

* **Size Conversion**: The size column (e.g., "2 BHK") was processed to extract the numeric bedroom count (bhk), creating a new feature.
* **Total Square Footage**: The total\_sqft column, originally an object type with mixed formats (e.g., ranges like "1000-1200"), was cleaned and converted to numeric values (likely by averaging ranges or handling exceptions).

**2.5 Handling Missing Values**

* Although not explicitly shown, libraries like SimpleImputer were imported, suggesting imputation strategies (e.g., mean/median for bath and balcony, mode for categorical columns like location).

**2.6 Feature Selection**

* The final features used for modeling were location, total\_sqft, bath, and bhk, dropping less predictive columns like society (due to high missing values), availability, area\_type, and balcony.

**2.7 Encoding and Scaling**

* location (categorical, 1,305 unique values) was one-hot encoded using OneHotEncoder to create binary features.
* Numerical features (total\_sqft, bath, bhk) were standardized using StandardScaler to normalize their scales.

**3. Model Selection and Optimization Approach**

**3.1 Model Selection**

* Multiple regression algorithms were imported, including LinearRegression, Lasso, Ridge, RandomForestRegressor, and XGBRegressor.
* The final model chosen was Ridge regression (L2 regularization), as evidenced by the pipeline (Ridge(alpha=1)).

**3.2 Pipeline Construction**

* A Pipeline was created using make\_pipeline and ColumnTransformer:
  + OneHotEncoder for location.
  + StandardScaler for total\_sqft, bath, and bhk.
  + Ridge regression (alpha=1).

**3.3 Train-Test Split**

* The data was split into training (X\_train, Y\_train) and testing (X\_test, Y\_test) sets using train\_test\_split.

**3.4 Evaluation**

* The model was evaluated using MAE, RMSE, and R²:
  + MAE = 15.87, RMSE = 24.53, R² = 0.914.

**3.5 Optimization**

* GridSearchCV and RandomizedSearchCV were imported, but alpha=1 for Ridge suggests either manual selection or reliance on default parameters after initial experimentation.

**4. Deployment Strategy and API Usage Guide**

**4.1 Model Serialization**

* The trained pipeline was saved using pickle (RidgeModel.pkl).
* Feature metadata (columns.json) was saved for input validation.

**4.2 Deployment Strategy**

* **Local Deployment**: The model can be loaded in a Python environment.
* **Web Deployment**: Flask or FastAPI can serve the model as an API.

**4.3 API Usage Guide**

* **Endpoint**: POST /predict
* **Request Format**:
* {"location": "string", "total\_sqft": float, "bath": int, "bhk": int}
* **Response Format**:
* {"predicted\_price": float}

**5. Conclusion**

The project successfully preprocessed the Bangalore house dataset, engineered relevant features, and trained a Ridge regression model with strong performance (R² = 0.914). The deployment strategy enables practical use via a serialized model and a potential API, making it accessible for real-world price predictions.