# House Price Prediction - Final Report & Documentation

This report provides an overview of the process followed in building and deploying a machine learning model for **house price prediction**. The model leverages various regression techniques and is optimized for performance before deployment as an API. The following sections detail data preprocessing, model selection, and deployment strategy.

.

**Part 1: Data Preprocessing:**

* Load the dataset and perform exploratory data analysis (EDA).
* Handle missing values appropriately.
* Perform feature engineering (scaling, encoding categorical variables, feature selection).
* Visualize correlations between features and the target variable.

**Dataset Used:**

* The dataset used in this project is the California Housing Dataset from sklearn.datasets.
* This dataset provides **median house values** for various districts in California.
* To enhance interpretability and ensure consistency in regression modeling, the house price values were converted to actual scale by multiplying by 100,000.

**Why Convert Price:**

* Regression models perform better when target values are within a reasonable numerical scale.
* Keeping small target values, such as **2.5**, can lead to unstable training behaviour.
* By converting the values to actual dollars, the model training process becomes more robust and results are easier to interpret.
* **Exploratory Data Analysis (EDA)** was performed to understand data distribution and relationships between variables.
* Key insights were obtained through visualization techniques such as correlation matrices, scatter plots, and histograms.

**Handling Missing Values:**

* The dataset was checked for missing values using **.isnull().sum(),**
* No missing values in California Housing Dataset; otherwise, we can handle them using:

**housing\_data.fillna(housing\_data.median(), inplace=True)**

**Feature Scaling & Encoding:**

* Feature Scaling: Since house prices and other attributes had varying ranges, StandardScaler was applied.
* Encoding Categorical Variables: If the dataset contained categorical variables, they were encoded using One-Hot Encoding.

**Correlation Analysis:**

Correlation analysis was conducted to evaluate the relationships between features in the dataset. The Pearson correlation coefficient quantifies the strength and direction of linear relationships between variables, expressed in a unitless standardized form.

The correlation coefficient provides insights into feature relationships:

* Positive Correlation (0 to +1): As one variable increases, the other also increases.
* Negative Correlation (0 to -1): As one variable increases, the other decreases.
* No Correlation (0): No linear relationship between the variables.

**Key Observations:**

* Most influential predictor: Median Income (**MedInc**) has a strong correlation (**0.69**) with house prices, making it the most significant feature for prediction.
* Weakly related features: **AveRooms** (**0.15**) and **HouseAge** (**0.11**) exhibit low correlation with house prices, indicating limited predictive power.
* Least important features: Population, AveOccup, and Longitude show negligible correlation with house prices and may be less useful for modeling.
* Multicollinearity issue: AveRooms and AveBedrms are highly correlated (**0.85**), which may introduce redundancy if both features are retained.

**Part 2: Model Training & Evaluation**

**Train-Test Split**

* Data was split into **80% training and 20% testing**.

**Train a Regression Model**

* Multiple regression models were tested:
* **Linear Regression**: Baseline model.
* **Decision Tree Regressor**: Captures non-linear relationships.
* **Random Forest Regressor**: Handles feature interactions better.
* **XGBoost Regressor**: Provides high accuracy and robustness.

**Evaluation**

* Models were evaluated using **Mean Squared Error (MSE), R-squared (R²), and Mean Absolute Error (MAE).**
* XGBoost provided the best results with the lowest error metrics

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE | RMSE | R² Score |
| Linear Regression | 53,320.01 | 74,558.14 | 0.58 |
| Decision Tree | 45,444.61 | 70,584.19 | 0.62 |
| Random Forest | 32,833.38 | 50,740.91 | 0.80 |
| XGBoost | 31,545.10 | 47,672.03 | 0.83 |

**Hyperparameter Tuning**

* **RandomizedSearchCV** used for hyperparameter optimization.
* The best parameters were selected to maximize prediction accuracy and minimize overfitting.

**Save Model**

* The trained model was saved using **pickle** for efficient loading.

**Part 3: Model Deployment**

**Deployment Approach:**

* Flask was used to deploy the model as a REST API.
* The trained model was serialized using **pickle** for efficient loading.
* The API was hosted on a server for real-time prediction.
* Hosting: Runs locally on <http://127.0.0.1:8000>
* API Endpoints:
  + / → Home
  + /predict → Takes house features in JSON & returns price prediction

**API Usage Instructions:**

* **Run FastAPI Server:** uvicorn app:app --host 0.0.0.0 --port 8000 –reload
* **Expected Output in Terminal:** INFO: Uvicorn running on <http://127.0.0.1:8000>

**Test API with Postman**

* Open **Postman**.
* **POST Request** to: <http://127.0.0.1:8000/predict>
* **Body → raw → JSON**, enter

{

"MedInc": 8.0,

"HouseAge": 25,

"AveRooms": 6.5,

"AveBedrms": 1.5,

"Population": 2000,

"AveOccup": 3.0,

"Latitude": 37.5,

"Longitude": -122.0

}

* Click **Send** → Get price prediction.

As per the requirements, the **FastAPI**-based house price prediction model is now deployed and can be tested using **CURL** in PowerShell or Linux/macOS terminal.

**Test API using CURL**

Use the appropriate command based on your system:

* **For Windows (PowerShell Command):**

Invoke-WebRequest -Uri "http://127.0.0.1:8000/predict" `

-Method Post `

-Headers @{"Content-Type"="application/json"} `

-Body '{"MedInc": 8.0, "HouseAge": 25, "AveRooms": 6.5, "AveBedrms": 1.5, "Population": 2000, "AveOccup": 3.0, "Latitude": 37.5, "Longitude": -122.0}' `

-UseBasicParsing

* **For Linux/Mac/Git Bash Users:**

curl -X POST "http://127.0.0.1:8000/predict" -H "Content-Type: application/json" -d '{"MedInc": 8.0, "HouseAge": 25, "AveRooms": 6.5, "AveBedrms": 1.5, "Population": 2000, "AveOccup": 3.0, "Latitude": 37.5, "Longitude": -122.0}'

* **Expected JSON Response Example:**

{

"predicted\_price": 452345.89

}

**Note: Please run the API using uvicorn in VS Code before testing**

**Overview:**

* We built a regression model using Random Forest and XGBoost for predicting house prices.
* Data preprocessing included scaling, correlation analysis, and feature selection.
* The best model was optimized using GridSearchCV.
* The model was deployed using FastAPI.

**Conclusion**

This project successfully developed a machine learning model to predict house prices with high accuracy. The deployment strategy ensures easy accessibility via an API, allowing real-time predictions. This structured workflow enables further refinements and scalability for large-scale applications.