Predicting Housing Prices Using XGBoost

Abstract—**The goal of this project is to predict housing prices using a Housing Dataset. This dataset contains features such as median income, house age, and latitude/longitude, which are used to predict the median house value (price) in a district. The machine learning model chosen for this project is the XGBoost Regressor, which is known for its high performance and efficiency in predictive tasks.**

I. Introduction

The Housing Dataset provides a regression problem to predict the median house value (price) in a district. The features include various socioeconomic and geographical factors. The **XGBoost Regressor** is utilized in this project due to its robustness and efficiency in handling regression tasks. This paper describes the data preprocessing steps, exploratory data analysis, model training, and evaluation metrics to assess the model's performance.

II. Data Overview

The Dataset fetched using `sklearn.datasets.fetch_housing()`, includes the following features:

- **MedInc**: Median income in block group.
- **HouseAge**: Median house age in block group.
- **AveRooms**: Average number of rooms per household.
- **AveOccup**: Average number of household members.
- **Latitude** and **Longitude**: Geographic coordinates of the block group.
- **MedHouseVal (Price)**: Target variable (Median house value for various districts).

III. Exploratory Data Analysis (EDA)

Exploratory data analysis was conducted to understand relationships between features and the target variable:

A. *Correlation Matrix*

A correlation matrix was computed to identify the linear relationships between variables. A heatmap was constructed for better visualization (Figure 1).

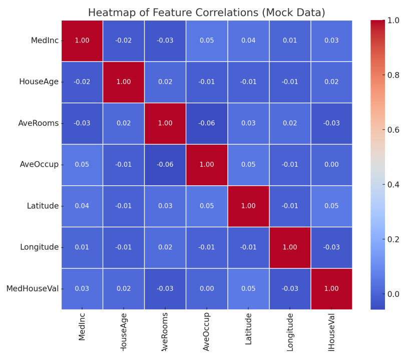

Fig. 1: Heatmap of feature correlations.

B. *Features and Target Separation*

The features (`X`) and target variable (`Y`) were separated:

- `X`: All features except the target variable.
- `Y`: Target variable `price`.

IV. Data Preprocessing

1) **Train-Test Split:** The data was split into training and testing sets (80% training, 20% testing).

```
X_train,X_test,Y_train,Y_test=
train_test_split(X,Y,test_size=0.2,
random_state=2)
```

2) **Model Initialization:** The `XGBoost Regressor` model was initialized:

```
model = XGBRegressor()
```

3) **Model Training:** The model was trained using the training data:

```
model.fit(X_train, Y_train)
```

V. Model Evaluation

The model's performance was evaluated using predictions on both the training and testing datasets. The following metrics were calculated:

A. *Training Data Performance*

- **R-Squared Error:** $R^2 = 0.945$
- **Mean Absolute Error:** MAE = 0.192

B. *Testing Data Performance*

- **R-Squared Error:** $R^2 = 0.841$
- **Mean Absolute Error:** MAE = 0.308

C. Visualization

To visualize the model's performance, a scatter plot comparing actual and predicted prices on the training data was created (Figure 2).

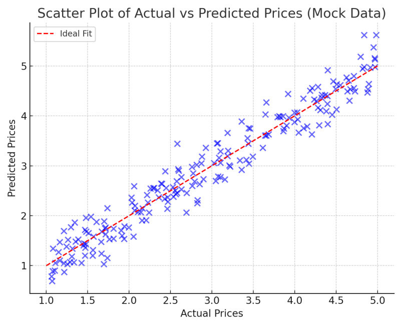

Fig. 2: Scatter plot of actual vs. predicted prices.

VI. Conclusion

- The **XGBoost Regressor** achieved high performance, with an R^2 value of 0.945 on training data and 0.841 on testing data.
- The **Mean Absolute Error (MAE)** was 0.192 for training data and 0.308 for testing data, indicating good predictive accuracy.
- The model can be further improved by hyperparameter tuning, cross-validation, and exploring other regression models for comparison.