# House\_Price\_Prediction

# 🏠 House Price Prediction System

## DESCRIPTION :

This project presents a \*\*House Price Prediction System\*\* using \*\*Linear Regression and Random Forest\*\*, built with Python and scikit-learn. The goal is to predict house prices based on a variety of features such as lot size, number of rooms, year built, and more. The model is trained on a cleaned and preprocessed version of a structured housing dataset.

This project demonstrates the application of supervised machine learning for price estimation tasks and includes visual analysis of model performance. It is ideal for learners or developers building real estate or property valuation platforms.

---

## 📂 Dataset

The dataset used is a CSV file named `house\_data.csv`, which contains the following features:

- `LotArea`: Size of the lot

- `OverallQual`: Overall quality of the material and finish

- `OverallCond`: Overall condition of the house

- `YearBuilt`: Year the house was built

- `GrLivArea`: Above-ground living area (square feet)

- `GarageCars`: Garage capacity

- `GarageArea`: Garage size (square feet)

- `TotalBsmtSF`: Total basement area

- `FullBath`: Number of full bathrooms

- `HalfBath`: Number of half bathrooms

- `BedroomAbvGr`: Bedrooms above ground level

- `TotRmsAbvGrd`: Total rooms above ground level

- `SalePrice`: The target variable (house price)

---

## 🧼 Data Preprocessing

Key steps:

1. \*\*Drop Null Columns\*\*: Removed columns with missing values.

2. \*\*Feature Selection\*\*: Extracted numeric features for training.

3. \*\*Missing Value Imputation\*\*: Filled missing values with mean.

4. \*\*Train-Test Split\*\*: Split the dataset into 80% training and 20% testing.

5. \*\*Standard Scaling\*\*: Applied standardization for Linear Regression.

```python

X = df.select\_dtypes(include=[np.number]).drop("SalePrice", axis=1)

y = df["SalePrice"]

X = X.fillna(X.mean())

```

---

# 🧠 MODEL DEVELOPMENT:

We used two models:

- Linear Regression

- Random Forest Regressor

## Steps:

```python

lr\_model = LinearRegression()

rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

lr\_model.fit(X\_train\_scaled, y\_train)

rf\_model.fit(X\_train, y\_train)

y\_pred\_lr = lr\_model.predict(X\_test\_scaled)

y\_pred\_rf = rf\_model.predict(X\_test)

```

---

# 📊 Evaluation Metrics:

The model is evaluated using the following metrics:

- \*\*R² Score\*\*: Measures how well the predictions approximate actual values

- \*\*MAE\*\*: Mean Absolute Error

- \*\*RMSE\*\*: Root Mean Squared Error

Example Output:

```

Model: LinearRegression

MAE: 17981.29

RMSE: 20963.53

R² Score: 0.8693

Model: RandomForestRegressor

MAE: 11600.00

RMSE: 15624.80

R² Score: 0.9241

```

---

# 💾 Saving the Model :

The model can be saved using joblib to reuse it without retraining:

```python

import joblib

joblib.dump(rf\_model, "models/house\_price\_model.pkl")

```

---

📁 Project Structure

```

House\_Price\_Prediction/

├── house\_data.csv

├── House Price Prediction.py

├── output/

│ ├── code\_preview.png

│ └── prediction\_plot.png

├── models/

│ └── house\_price\_model.pkl

└── README.md

```

---

# How to run :

1. Load the dataset `house\_data.csv`

2. Install required Python packages:

```bash

pip install pandas numpy matplotlib seaborn scikit-learn joblib

```

3. Run the Python script:

```bash

python "House Price Prediction.py"

```

---

# 🚀 Future Improvements:

- Add advanced models like XGBoost, LightGBM

- Feature selection and model tuning

- Serve predictions using Flask or FastAPI

- Add UI to upload CSV and visualize predictions

---

# 🙋 Contact

Author: \*\*Pandi Hemaprasad\*\*

Internship Project

Technology Stack: Python, Scikit-learn, Pandas, Matplotlib

---

# OUTPUT :

The script prints performance metrics to the terminal and displays a plot comparing actual and predicted house prices using matplotlib.

```

MAE: 11600.00

RMSE: 15624.80

R² Score: 0.9241

```



