House Price Prediction Project Description

1. Overview

The House Price Prediction project is a web-based application designed to estimate house prices using a machine learning model trained on the Ames Housing dataset (train.csv). The application integrates a Flask backend, an XGBoost regression model, and a Jinja2-templated frontend to deliver a user-friendly interface. Users can input specific house characteristics, receive accurate price predictions, and view model performance metrics, making it a valuable tool for real estate analysis, educational purposes, or machine learning demonstrations.

2. Purpose

The project aims to showcase the application of machine learning in solving real-world regression problems, specifically predicting house prices based on key property features. By combining data preprocessing, model training, and web development, it provides an end-to-end solution that is both functional and educational. It serves real estate enthusiasts, data scientists, and developers interested in integrating machine learning with web technologies.

3. Features and Fields

The application uses six key features from the Ames Housing dataset, selected for their strong correlation with house prices and interpretability. Below is a detailed description of each field:

1. **OverallQual (Overall Quality):**

- Description: Rates the overall material and finish quality of the house on a scale from 1 (Very Poor) to 10 (Very Excellent).
- Role: Captures the general quality of construction and materials, significantly influencing house value. Higher ratings correlate with higher prices due to better craftsmanship.
- Example: A house with high-end finishes might have OverallQual=8, while a basic home might be 4.
- Data Type: Integer (1–10).
- Preprocessing: No missing values; used as-is.

2. **GrLivArea (Above-Grade Living Area):**

- Description: Measures the above ground living area in square feet, encompassing finished living spaces (e.g., bedrooms, living rooms) excluding basements.
- Role: A key determinant of house size and livability, strongly impacting price as larger spaces command higher values.
- Example: A house with 1500 square feet of living space has GrLivArea=1500.
- Data Type: Integer (square feet).
- Preprocessing: No missing values; scaled using StandardScaler.

3. **GarageCars (Garage Capacity):**

- Description: Indicates the number of cars that can fit in the garage, reflecting garage size.
- Role: Affects parking and storage convenience, with larger garages increasing value.
- Example: A two-car garage has GarageCars=2.
- Data Type: Integer (typically 0–4).
- Preprocessing: Missing values filled with 0 (no garage).

4. **TotalBsmtSF (Total Basement Square Feet):**

- Description: Total area of the basement in square feet, including finished and unfinished portions.
- Role: Contributes to potential living or storage space, influencing value, especially if finished.
- Example: A 1000-square-foot basement has TotalBsmtSF=1000.
- Data Type: Integer (square feet).
- Preprocessing: Missing values filled with 0; scaled using StandardScaler.

5. **FullBath (Number of Full Bathrooms):**

- Description: Counts full bathrooms (with sink, toilet, and shower/tub).
- Role: Essential for functionality and comfort; more bathrooms increase value.
- Example: A house with two full bathrooms has FullBath=2.
- Data Type: Integer (typically 0–4).
- Preprocessing: No missing values; used as-is.

6. **YearBuilt (Year Built):**

- Description: The year the house was originally constructed.
- Role: Newer houses often have higher prices due to modern standards; older houses may need renovations.
- Example: A house built in 2000 has YearBuilt=2000.
- Data Type: Integer (year).
- Preprocessing: No missing values; scaled using StandardScaler.

7. **SalePrice (Target Variable):**

- Description: The sale price of the house in dollars, the target for prediction.
- Role: Reflects the market value predicted by the model.
- Example: A house sold for $200,000 has SalePrice=200000.
- Data Type: Integer (dollars).
- Preprocessing: No missing values; used as the target.

These features capture essential aspects of a house’s quality, size, amenities, and age, driving market value.

4. Technical Implementation

Data Preprocessing:

Loads train.csv using pandas.

Fills missing GarageCars and TotalBsmtSF with 0.

Scales numerical features (GrLivArea, TotalBsmtSF, YearBuilt) using StandardScaler.

Splits data into 80% training and 20% testing sets.

Machine Learning Model:

Uses XGBoost regressor with n_estimators=100, learning_rate=0.1, max_depth=5, objective='reg:squarederror'.

Evaluates with MSE and R-squared metrics.

Saves model and scaler as model.pkl and scaler.pkl.

Backend:

Flask handles routing:

/: Renders index.html with form and metrics.

/predict: Processes form inputs, scales data, predicts price, and handles errors.

Uses joblib for model persistence.

Frontend:

index.html uses Jinja2 to render form, metrics, predictions, and errors.

CSS ensures a responsive, professional design with form container, result/error boxes, and hover effects.

Dependencies:

Python 3.12

pandas, numpy, scikit-learn, xgboost, flask, joblib

Implementation:

Access at http://127.0.0.1:5001.

Input feature values in the form and submit.

View predicted price and model metrics (MSE, R-squared).

Errors are displayed for invalid inputs.

Model Performance:

MSE measures prediction error (lower is better).

R-squared indicates explained variance (closer to 1 is better).

Access:

Visit http://127.0.0.1:5001.

Input values (e.g., OverallQual=8, GrLivArea=2000, GarageCars=3, TotalBsmtSF=1500, FullBath=3, YearBuilt=2010).

The predicted price is displayed.

5. Limitations

Limited to six features.

Local debug model production requires additional setup.

Dataset-specific predictions may not generalize.

6. Conclusion

The project demonstrates machine learning and web development integration, offering accurate house price predictions via an accessible interface. Its modular design supports extensions for enhanced functionality or deployment.