🏡 HOUSE PRICE PREDICTION USING MACHINE LEARNING

Using Kaggle "House Prices — Advanced Regression Techniques" Dataset
Goal: Predict sale prices of residential homes using ML models.

---

1. INTRODUCTION

Real estate pricing depends on multiple factors such as neighborhood, number of rooms, building quality, and age of the property. The goal of this project is to build a **machine learning regression model** that accurately predicts house prices using the **Kaggle House Prices** dataset.
This project includes:
✓ Exploratory data analysis (EDA)
✓ Data preprocessing
✓ Model training
✓ Regularization (LASSO & Ridge)
✓ Evaluation metrics
✓ Feature importance
✓ Deployment with Streamlit

---

2. DATASET DESCRIPTION

Dataset Source: Kaggle — *House Prices: Advanced Regression Techniques*
Files used:

- `train.csv` (1460 rows, 80 features)
- `test.csv` (for optional submission)

Target Variable
`SalePrice` — the house sale price (continuous variable).
Feature Types

Type	Examples
Numeric	LotArea, GrLivArea, OverallQual, YearBuilt

Categorical	Neighborhood, HouseStyle, RoofStyle
Ordinal	OverallCond, ExterQual, KitchenQual

3. IMPORT LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error, r2_score
```

4. LOAD THE DATA

```
df = pd.read_csv("train.csv")
df.head()
df.info()
df.describe()
```

5. EXPLORATORY DATA ANALYSIS (EDA)

5.1 Missing Values

```
df.isnull().sum().sort_values(ascending=False).head(20)
```

5.2 Correlation with SalePrice

```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr()['SalePrice'].sort_values(ascending=Fals
e).to_frame(), annot=True)
```

Most correlated features:

- OverallQual
- GrLivArea
- GarageCars
- TotalBsmtSF
- YearBuilt

5.3 Distribution of SalePrice

```
sns.histplot(df['SalePrice'], kde=True)
```

---

6. DATA PREPROCESSING

6.1 Define Feature Groups

```
y = df['SalePrice']
X = df.drop(columns=['SalePrice'])


num_cols = X.select_dtypes(include=['int64','float64']).column
s
cat_cols = X.select_dtypes(include=['object']).columns
```

6.2 Transformers

```
num_transformer = Pipeline([
```

```
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

cat_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer([
    ('num', num_transformer, num_cols),
    ('cat', cat_transformer, cat_cols)
])
```

---

7. TRAIN-TEST SPLIT

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

---

8. MODEL TRAINING

We build **3 models:**
1️⃣ **Linear Regression** (baseline)
2️⃣ **LASSO Regression** (L1 regularization)
3️⃣ **Ridge Regression** (L2 regularization)

8.1 Linear Regression

```
linreg = Pipeline([
```

```
    ('pre', preprocessor),
    ('model', LinearRegression())
])

linreg.fit(X_train, y_train)
pred_linreg = linreg.predict(X_test)
```

8.2 LASSO Regression

```
lasso = Pipeline([
    ('pre', preprocessor),
    ('model', Lasso(alpha=0.001, max_iter=10000))
])

lasso.fit(X_train, y_train)
pred_lasso = lasso.predict(X_test)
```

8.3 Ridge Regression

```
ridge = Pipeline([
    ('pre', preprocessor),
    ('model', Ridge(alpha=1.0))
])

ridge.fit(X_train, y_train)
pred_ridge = ridge.predict(X_test)
```

---

9. MODEL EVALUATION

```
def evaluate_model(y_test, y_pred, name):
```

```
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    r2 = r2_score(y_test, y_pred)
    print(f"{name} RMSE: {rmse}")
    print(f"{name} R² Score: {r2}")

evaluate_model(y_test, pred_linreg, "Linear Regression")
evaluate_model(y_test, pred_lasso, "LASSO Regression")
evaluate_model(y_test, pred_ridge, "Ridge Regression")
```

✓ **Expected Result**

- Ridge usually performs best
- LASSO is good for feature selection
- Linear Regression provides baseline performance

---

10. FEATURE IMPORTANCE (Ridge / LASSO)

```
model = ridge.named_steps['model']
pre = ridge.named_steps['pre']

feature_names = (
    list(num_cols) +
    list(pre.named_transformers_['cat']['onehot'].get_feature_names_out(cat_cols))
)

coeff = pd.Series(model.coef_, index=feature_names).sort_values(key=abs, ascending=False)
coeff.head(20)
```

---

11. SHAP INTERPRETATION (Optional Advanced)

```
import shap


explainer = shap.Explainer(ridge.named_steps['model'], pre.transform(X_train))
shap_values = explainer(pre.transform(X_train))


shap.summary_plot(shap_values, feature_names=feature_names)
```

---

12. DEPLOYMENT USING STREAMLIT

Create **app.py**

```
import streamlit as st
import pandas as pd
import joblib


model = joblib.load("ridge_model.pkl")


st.title("□ House Price Prediction App")


LotArea = st.number_input("Lot Area", min_value=100, max_value=200000, value=5000)
OverallQual = st.slider("Overall Quality", 1, 10, 5)
YearBuilt = st.number_input("Year Built", 1800, 2025, 1990)


data = pd.DataFrame({
    'LotArea':[LotArea],
    'OverallQual':[OverallQual],
```

```
    'YearBuilt':[YearBuilt]
})


if st.button("Predict Price"):
    price = model.predict(data)[0]
    st.success(f"Estimated Price: ${price:,.2f}")
```

Run:

```
streamlit run app.py
```

---

13. CONCLUSION

In this project, we:

✓ Loaded & analyzed the Kaggle housing dataset
✓ Handled missing values, scaling & encoding
✓ Trained Linear, LASSO, and Ridge models
✓ Identified the best model (usually Ridge)
✓ Interpreted features
✓ Built a Streamlit app for deployment