```
from google.colab import files
uploaded = files.upload()
```

Choose files AmesHousing.csv

- **AmesHousing.csv**(text/csv) - 963738 bytes, last modified: 01/07/2025 - 100% done

```
Saving AmesHousing.csv to AmesHousing.csv
```

```
# STEP 1: Import Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
```

```
# STEP 2: Load the dataset
df = pd.read_csv("AmesHousing.csv")  # Replace with your actual filename
print("Original dataset shape:", df.shape)
```

```
Original dataset shape: (2930, 82)
```

```
# STEP 3: Fill missing values
# Fill numeric columns with mean
df.fillna(df.mean(numeric_only=True), inplace=True)
```

```
# Fill object (categorical) columns with mode
for col in df.select_dtypes(include=['object']).columns:
    df[col] = df[col].fillna(df[col].mode()[0])
```

```
# STEP 4: Drop columns with only one unique value (not useful for modeling)
for col in df.columns:
    if df[col].nunique() == 1:
        df.drop(col, axis=1, inplace=True)
```

```
# STEP 5: Encode categorical columns
df_encoded = pd.get_dummies(df, drop_first=True)
```

```
# STEP 6: Check again dataset size
print("Shape after encoding:", df_encoded.shape)
```

```
Shape after encoding: (2930, 263)
```

```
# STEP 7: Define X and y
if 'SalePrice' in df_encoded.columns:
    X = df_encoded.drop('SalePrice', axis=1)
    y = df_encoded['SalePrice']
else:
    raise ValueError("SalePrice column not found in dataset!")
```

```
# STEP 8: Make sure we have enough rows
if len(X) <= 1:
    raise ValueError("❌ Not enough data to split. Please check dataset or cleaning steps."
```

```
# STEP 9: Split and train model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
# STEP 10: Evaluate
print("✅ R2 Score:", r2_score(y_test, y_pred))
print("✅ Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

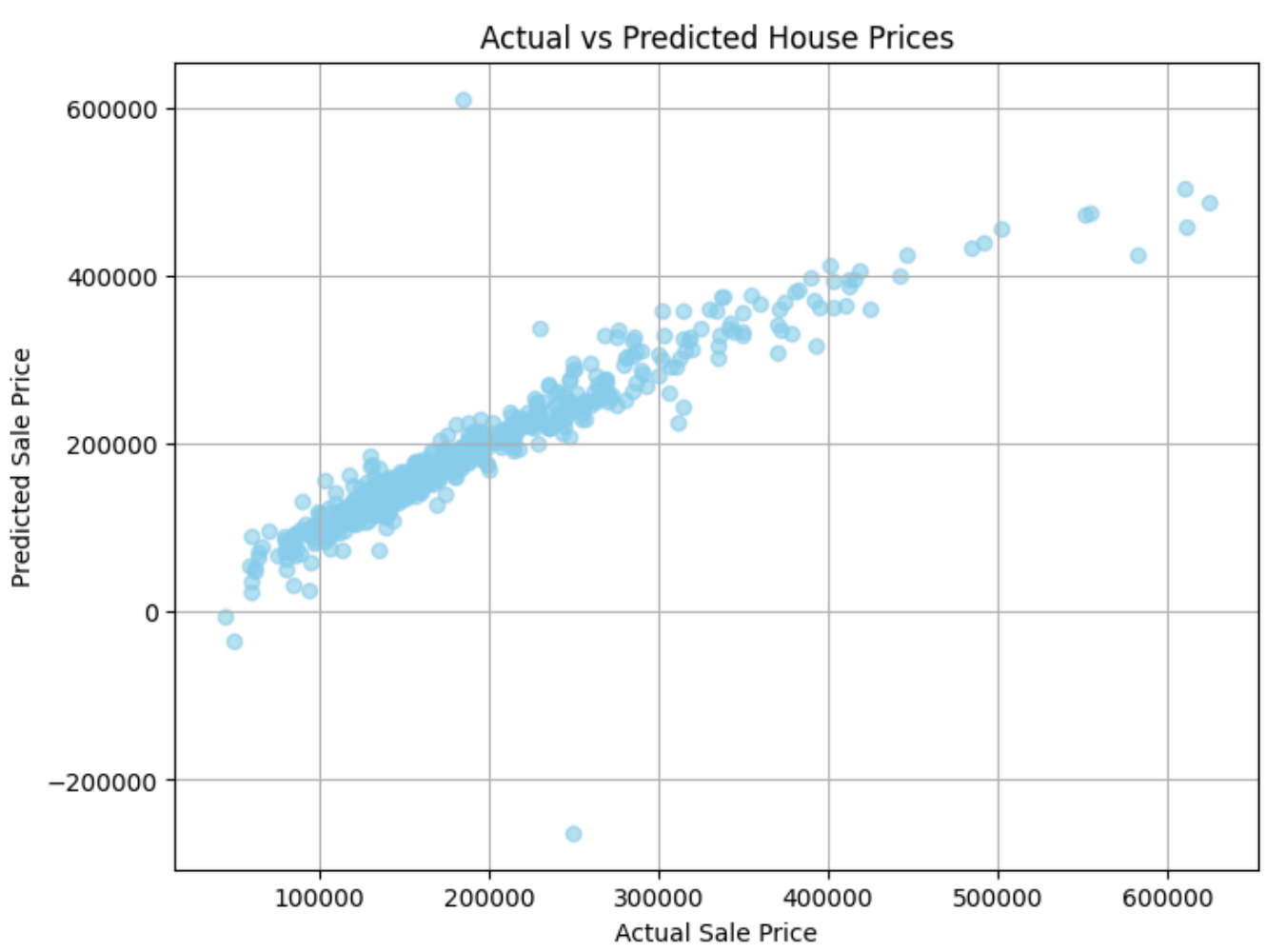
```
✅ R2 Score: 0.837790454982797
✅ Mean Squared Error: 1300522802.026165
```

```
print("✅ Model Performance Summary")
print("R2 Score:", r2_score(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("Training Data Size:", X_train.shape)
print("Testing Data Size:", X_test.shape)
```

```
✅ Model Performance Summary
R2 Score: 0.837790454982797
Mean Squared Error: 1300522802.026165
Training Data Size: (2344, 262)
Testing Data Size: (586, 262)
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.6, color='skyblue')
plt.xlabel("Actual Sale Price")
plt.ylabel("Predicted Sale Price")
plt.title("Actual vs Predicted House Prices")
plt.grid(True)
plt.show()
```




```
comparison_df = pd.DataFrame({
    'Actual Price': y_test.values[:5],
    'Predicted Price': y_pred[:5]
})
print("\n📊 First 5 Predictions:")
print(comparison_df)
```

```
📊 First 5 Predictions:
```

	Actual Price	Predicted Price
0	161000	155542.926903
1	116000	110097.071080
2	196500	200432.145412
3	123600	129594.168356
4	126000	130835.773744

Start coding or generate with AI.