1.upload the dataset

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

Choose Files housing_pri…dataset.csv

- **housing_price_dataset.csv**(text/csv) - 1661213 bytes, last modified: 5/11/2025 - 100% done

```
Saving housing_price_dataset.csv to housing_price_dataset.csv
```

2.Load the dataset

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

file_path = "housing_price_dataset.csv"  # Updated file_path to the uploaded file name
df = pd.read_csv(file_path)
df.head()
```

	SquareFeet	Bedrooms	Bathrooms	Neighborhood	YearBuilt	Price
0	2126	4	1	Rural	1969	215355.2836
1	2459	3	2	Rural	1980	195014.2216
2	1860	2	1	Suburb	1970	306891.0121
3	2294	2	1	Urban	1996	206786.7872
4	2130	5	2	Suburb	2001	272436.2391

Next steps: Generate code with df View recommended plots New interactive sheet

3.Data exploration

```
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   SquareFeet    50000 non-null  int64  
 1   Bedrooms      50000 non-null  int64  
 2   Bathrooms     50000 non-null  int64  
 3   Neighborhood  50000 non-null  object 
 4   YearBuilt     50000 non-null  int64  
 5   Price         50000 non-null  float64
dtypes: float64(1), int64(4), object(1)
memory usage: 2.3+ MB
None
         SquareFeet      Bedrooms     Bathrooms     YearBuilt          Price
count  50000.000000  50000.000000  50000.000000  50000.000000   50000.000000
mean    2006.374680      3.498700      1.995420   1985.404420  224827.325151
std      575.513241      1.116326      0.815851     20.719377   76141.842966
min     1000.000000      2.000000      1.000000   1950.000000  -36588.165400
25%     1513.000000      3.000000      1.000000   1967.000000  169955.860250
50%     2007.000000      3.000000      2.000000   1985.000000  225052.141150
75%     2506.000000      4.000000      3.000000   2003.000000  279373.630025
max     2999.000000      5.000000      3.000000   2021.000000  492195.260000
```

4.Check for missing values and duplicates

```
print("Missing values:\n", df.isnull().sum())
print("Duplicate rows:", df.duplicated().sum())
```

```
Missing values:
 SquareFeet      0
Bedrooms        0
Bathrooms       0
```

rror FileNotFoundError ModuleNotFound •••

Please explain the error:

```
ModuleNotFoundError: No module named 'gradio'
```

Hypothesis

The error message `ModuleNotFoundError: No module named 'gradio'` clearly indicates that the `gradio` library is not installed in the current environment. The code tries to import it using `import gradio as gr`, but the Python interpreter cannot find the module.

Suggested Changes

```
!pip install gradio
import gradio as gr
```

Use code with caution

Explanation:

1. **Install `gradio`:** The first line `!pip install gradio` uses `pip`, the Python package installer, to download and install the `gradio` library and its dependencies. The `!` prefix is used to run this command in the system shell within the Jupyter notebook environment.
2. **Import `gradio`:** After installation, the second line `import gradio as gr` imports the library, making its functionality accessible in the code. This should resolve the `ModuleNotFoundError`.

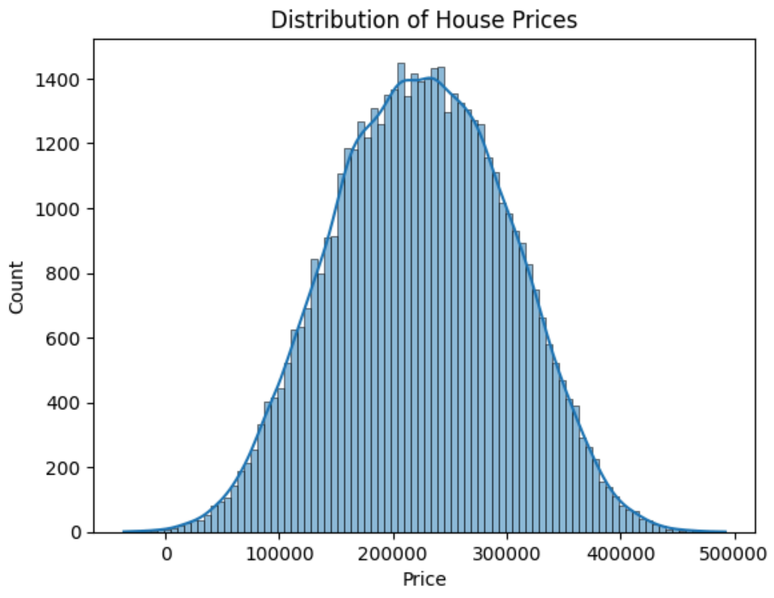
Rate this answer

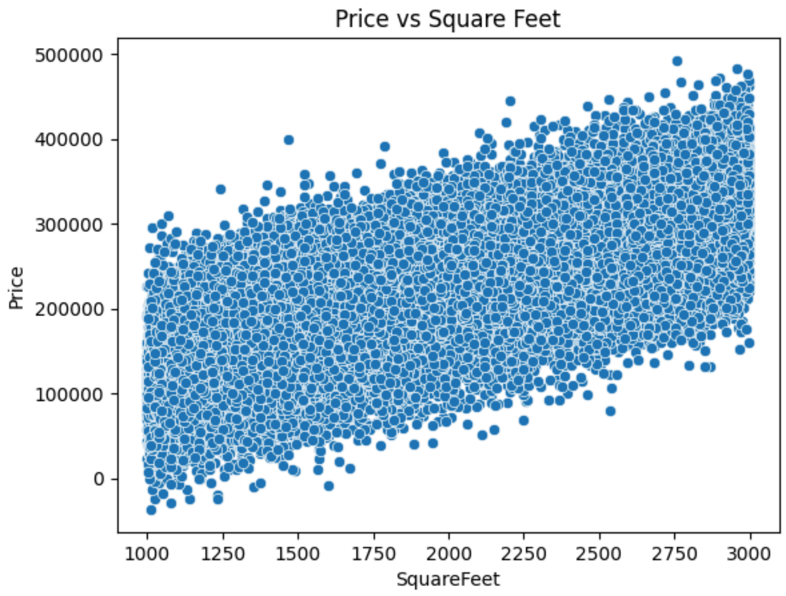
```
Neighborhood    0
YearBuilt       0
Price           0
dtype: int64
Duplicate rows: 0
```

5.Visualize a feww features

```
sns.histplot(df['Price'], kde=True)
plt.title('Distribution of House Prices')
plt.show()

sns.scatterplot(x='SquareFeet', y='Price', data=df)
plt.title('Price vs Square Feet')
plt.show()
```





6.Identify target and features

```
y = df['Price']
X = df.drop(['Price'], axis=1)
```

7.Convert categorical columns to numerical

```
X = pd.get_dummies(X, drop_first=True)
```

8.One-hot encoding

```
X = pd.get_dummies(X, drop_first=True)
```

9.Feature scaling

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

10.Train-test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random
```

11.Model building

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor

models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'Random Forest': RandomForestRegressor(),
    'XGBoost': XGBRegressor()
}
```

12.Evaluation

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n{name} Evaluation:")
    print("MAE:", mean_absolute_error(y_test, y_pred))
    print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
    print("R^2 Score:", r2_score(y_test, y_pred))
```

```
Linear Regression Evaluation:
MAE: 39430.165338362065
RMSE: 49358.37691102903
R^2 Score: 0.5755628630337024

Ridge Regression Evaluation:
MAE: 39430.15757776463
RMSE: 49358.37354139575
R^2 Score: 0.5755629209852617

Lasso Regression Evaluation:
MAE: 39430.15876851178
RMSE: 49358.36495700156
R^2 Score: 0.5755630686211994

Random Forest Evaluation:
MAE: 41829.967768518516
RMSE: 52514.50062214671
R^2 Score: 0.5195478775409099

XGBoost Evaluation:
MAE: 40248.26852179789
RMSE: 50510.4256738507
R^2 Score: 0.5555184962581994
```

13.Make predictions from new input

```
sample_input = X_test[0].reshape(1, -1)
predicted_price = models['XGBoost'].predict(sample_input)
print("\nPredicted Price for Sample Input:", predicted_price[0])
```

```

Predicted Price for Sample Input: 211595.73
```

16.Deployment -building an interactive app

```
!pip install gradio
import gradio as gr
```

```
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dis
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.1
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-pack
Downloading gradio-5.29.0-py3-none-any.whl (54.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.1/54.1 MB 12.0 MB/s eta 0:00:00
Downloading gradio_client-1.10.0-py3-none-any.whl (322 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 322.9/322.9 kB 27.7 MB/s eta 0:00:00
Downloading aiofiles-24.1.0-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.12-py3-none-any.whl (95 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.2/95.2 kB 8.7 MB/s eta 0:00:00
Downloading groovy-0.1.2-py3-none-any.whl (14 kB)
Downloading python_multipart-0.0.20-py3-none-any.whl (24 kB)
Downloading ruff-0.11.9-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 117.6 MB/s eta 0:00:00
Downloading safehttpx-0.1.6-py3-none-any.whl (8.7 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 6.9 MB/s eta 0:00:00
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Downloading uvicorn-0.34.2-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 5.1 MB/s eta 0:00:00
Downloading ffmpy-0.5.0-py3-none-any.whl (6.0 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub, uvicorn, tomlkit, semantic-version, ruff,
Successfully installed aiofiles-24.1.0 fastapi-0.115.12 ffmpy-0.5.0 gradio-5.29.
```

17.Create a prediction function

```
def predict_price(SquareFeet, Bedrooms, Bathrooms, YearBuilt, Neighborhood):
    df_input = pd.DataFrame({
        'SquareFeet': [SquareFeet],
        'Bedrooms': [Bedrooms],
        'Bathrooms': [Bathrooms],
        'YearBuilt': [YearBuilt],
        'Neighborhood': [Neighborhood]
    })
    df_encoded = pd.get_dummies(df_input)
    df_encoded = df_encoded.reindex(columns=X.columns, fill_value=0)
    scaled_input = scaler.transform(df_encoded)
    pred = models['XGBoost'].predict(scaled_input)
    return f"Predicted House Price: ${pred[0]:,.2f}"
```

18.Create the gradio interface

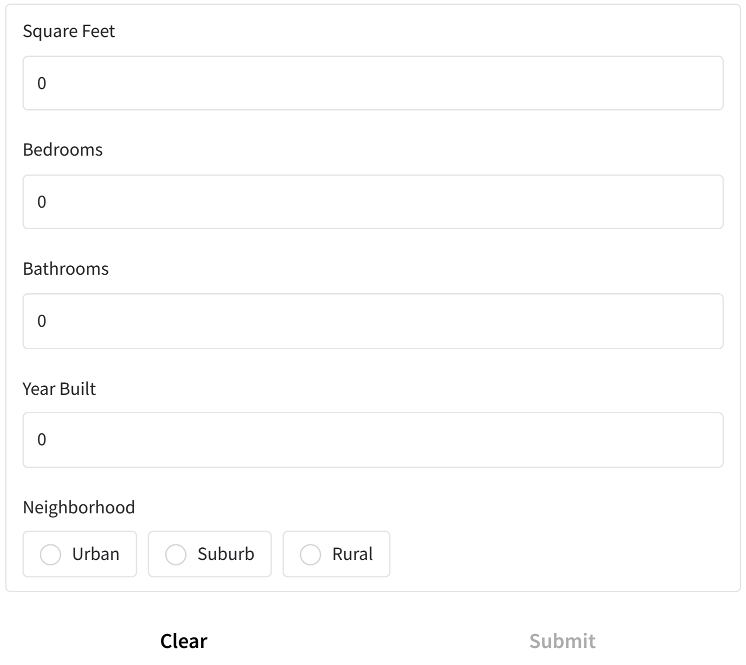
```
interface = gr.Interface(
    fn=predict_price,
    inputs=[
        gr.Number(label="Square Feet"),
        gr.Number(label="Bedrooms"),
        gr.Number(label="Bathrooms"),
        gr.Number(label="Year Built"),
        gr.Radio(["Urban", "Suburb", "Rural"], label="Neighborhood")
    ],
    outputs="text"
)

interface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradi

Colab notebook detected. To show errors in colab notebook, set debug=True in launc
* Running on public URL: https://e8a25a85df1d68b4a1.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, ru
```


Square Feet
0
Bedrooms
0
Bathrooms
0
Year Built
0
Neighborhood
◯ Urban ◯ Suburb ◯ Rural

rewite the ch

13 / 2000

Gemini can make mistakes so double-check responses and use code with caution. Learn more