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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Housing.csv to Housing.csv

```
import pandas as pd
df = pd.read_csv('Housing.csv')
df.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no

```
print("Shape:", df.shape)
print("Columns:", df.columns.tolist())
df.info()
df.describe()
```

```
Shape: (545, 13)
Columns: ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditior
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

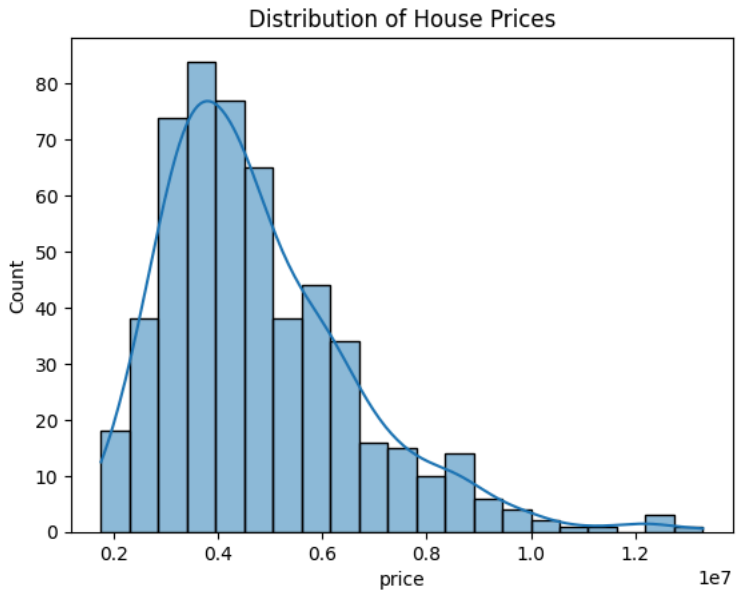
	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

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

```
Missing Values:
 price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
```

```
prefarea            0
furnishingstatus    0
dtype: int64
Duplicate Rows: 0
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.histplot(df['price'], kde=True)
plt.title('Distribution of House Prices')
plt.show()
sns.scatterplot(x='area', y='price', data=df)
plt.title('Area vs Price')
plt.show()
```




```
categorical_cols = df.select_dtypes(include=['object']).columns
df_encoded = pd.get_dummies(df, drop_first=True)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded.drop('price', axis=1))
y = df_encoded['price']
```

```
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_state=42)
```

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
MSE: 1754318687330.6675
R² Score: 0.6529242642153177
```

```
# Replace with actual column names from your dataset
new_house = {
    'area': 3000,
    'bedrooms': 3,
    'bathrooms': 2,
    'stories': 2,
    'mainroad': 'yes',
    'guestroom': 'no',
    'basement': 'yes',
    'hotwaterheating': 'no',
    'airconditioning': 'yes',
    'parking': 1,
    'prefarea': 'no',
    'furnishingstatus': 'furnished'
}
new_df = pd.DataFrame([new_house])
df_temp = pd.concat([df.drop('price', axis=1), new_df], ignore_index=True)
df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)
df_temp_encoded = df_temp_encoded.reindex(columns=df_encoded.drop('price', axis=1).columns, fill_value=0)
new_input_scaled = scaler.transform(df_temp_encoded.tail(1))
predicted_price = model.predict(new_input_scaled)
print("💰 Predicted House Price:", round(predicted_price[0], 2))
```

```
💰 Predicted House Price: 5976557.41
```

```
!pip install gradio
```

```
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 gradio-5.30.0-py3-none-any.whl (54.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.2/54.2 MB 16.4 MB/s eta 0:00:00
Downloading gradio_client-1.10.1-py3-none-any.whl (323 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.1/323.1 kB 18.5 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 5.2 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-0.11.10-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 73.5 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 4.2 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 5.8 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, python-multipart, groovy, ffmpy, aiofiles, starlet
Successfully installed aiofiles-24.1.0 fastapi-0.115.12 ffmpy-0.5.0 gradio-5.30.0 gradio-client-1.10.1 groovy-0.1.2 pydub-0.25.1 p
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
import gradio as gr

# Load the data
df = pd.read_csv("/content/drive/MyDrive/Housing.csv")

# One-hot encoding
df_encoded = pd.get_dummies(df, drop_first=True)

# Features and target
X = df_encoded.drop("price", axis=1)
y = df_encoded["price"]

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Save column structure
columns = X.columns

# Define prediction function
def predict_price(area, bedrooms, bathrooms, stories, mainroad, guestroom, basement,
                  hotwaterheating, airconditioning, parking, prefarea, furnishingstatus):

    # Form input dict
    input_dict = {
        'area': area,
        'bedrooms': bedrooms,
        'bathrooms': bathrooms,
        'stories': stories,
        'mainroad': mainroad,
        'guestroom': guestroom,
        'basement': basement,
        'hotwaterheating': hotwaterheating,
        'airconditioning': airconditioning,
        'parking': parking,
        'prefarea': prefarea,
        'furnishingstatus': furnishingstatus
    }

    # Convert to DataFrame and encode
    input_df = pd.DataFrame([input_dict])
    temp_df = pd.concat([df.drop("price", axis=1), input_df], ignore_index=True)
    temp_encoded = pd.get_dummies(temp_df, drop_first=True)
    temp_encoded = temp_encoded.reindex(columns=columns, fill_value=0)

    # Scale and predict
    input_scaled = scaler.transform(temp_encoded.tail(1))
    prediction = model.predict(input_scaled)[0]

    return round(prediction, 2)
```

```
# Define interface inputs
inputs = [
    gr.Number(label="Area (sq ft)"),
    gr.Number(label="Bedrooms"),
    gr.Number(label="Bathrooms"),
    gr.Number(label="Stories"),
    gr.Dropdown(["yes", "no"], label="Main Road Access"),
    gr.Dropdown(["yes", "no"], label="Guest Room"),
    gr.Dropdown(["yes", "no"], label="Basement"),
    gr.Dropdown(["yes", "no"], label="Hot Water Heating"),
    gr.Dropdown(["yes", "no"], label="Air Conditioning"),
    gr.Number(label="Parking Spaces"),
    gr.Dropdown(["yes", "no"], label="Preferred Area"),
    gr.Dropdown(["furnished", "semi-furnished", "unfurnished"], label="Furnishing Status")
]

# Create the interface
gr.Interface(
    fn=predict_price,
    inputs=inputs,
    outputs=gr.Number(label="Predicted House Price"),
    title="🏠 House Price Prediction App",
    description="Enter the house features to estimate its price."
).launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatica

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://025ee1efbdcda63a76.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working
```

Start coding or generate with AI.