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

import pandas as pd

# Load dataset
df = pd.read_csv(next(iter(uploaded)))
df.head()
```

Choose Files Housing.csv

- **Housing.csv**(text/csv) - 29981 bytes, last modified: 5/8/2025 - 100% done

```
Saving Housing.csv to Housing (1).csv
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furn
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	

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

```
# Step 2: Data Preprocessing
# Display basic info
df.info()

# Handle missing values: fill with median for numerics
df.fillna(df.median(numeric_only=True), inplace=True)

# Get actual column names
print("Columns:", df.columns)

# Define target column and features
target = df.columns[-1]  # Last column is the target
X = df.drop(columns=[target])
y = df[target]

# Identify categorical columns
categorical_features = X.select_dtypes(include=['object', 'bool']).columns.tolist()
print("Categorical features:", categorical_features)

# Apply One-Hot Encoding to categorical features
X = pd.get_dummies(X, columns=categorical_features, drop_first=True)

# Scale numeric features only
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # Now X contains only numeric data

# Done preprocessing
print("Preprocessing complete. Shape:", X_scaled.shape)
```

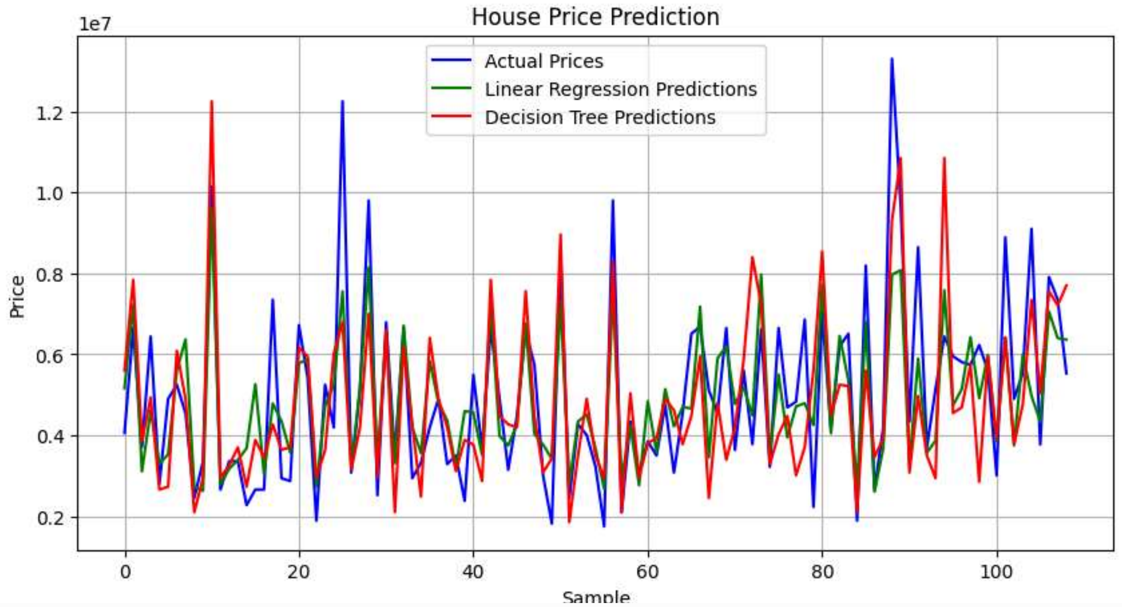
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          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
Columns: Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
Categorical features: ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']
Preprocessing complete. Shape: (545, 12)
```

```
# Step 3: Select Features and Target
target = 'price'  # Correct lowercase column name
X = df.drop(columns=[target])
y = df[target]

# Convert categorical features in X using one-hot encoding
categorical_cols = X.select_dtypes(include=['object', 'bool']).columns.tolist()
X = pd.get_dummies(X, columns=categorical_cols, drop_first=True)

# Scale features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 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_state=42)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)
lr_mse = mean_squared_error(y_test, lr_preds)

# Decision Tree Regressor
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
dt_preds = dt_model.predict(X_test)
dt_mse = mean_squared_error(y_test, dt_preds)

print(f"Linear Regression MSE: {lr_mse:.2f}")
print(f"Decision Tree MSE: {dt_mse:.2f}")
```

```
Linear Regression MSE: 1754318687330.67
Decision Tree MSE: 2642802637614.68
```

```
# Step 5: Plot predictions
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
plt.plot(y_test.values, label='Actual Prices', color='blue')
plt.plot(lr_preds, label='Linear Regression Predictions', color='green')
plt.plot(dt_preds, label='Decision Tree Predictions', color='red')
plt.title('House Price Prediction')
plt.xlabel('Sample')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```


House Price Prediction
1e7
Actual Prices
Linear Regression Predictions
Decision Tree Predictions
Price
1.2
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
Sample