```
# Step 1: Import file upload utility
from google.colab import files

# Step 2: Prompt the user to upload the file
uploaded = files.upload()

# Step 3: Read the uploaded file (adjust the filename accordingly)
import pandas as pd

df = pd.read_csv('house_prices.csv')  # Replace with your actual file name
print("Dataset loaded successfully. Shape:", df.shape)

# Optional: Show first few rows
df.head()
```

Choose Files house_prices.csv

- **house_prices.csv**(text/csv) - 106149815 bytes, last modified: 6/4/2025 - 100% done

```
Saving house_prices.csv to house_prices.csv
Dataset loaded successfully. Shape: (187531, 21)
```

| | Index | Title | Description | Amount(in rupees) | Price (in rupees) | location | Carpet Area | Status | Floor | Transaction | ... | facing | overlooking | Society | Bat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 BHK Ready to Occupy Flat for sale in Srushti... | Bhiwandi, Thane has an attractive 1 BHK Flat f... | 42 Lac | 6000.0 | thane | 500 sqft | Ready to Move | 10 out of 11 | Resale | ... | NaN | NaN | Srushti Siddhi Mangal Murti Complex | |
| **1** | 1 | 2 BHK Ready to Occupy Flat for sale in Dosti V... | One can find this stunning 2 BHK flat for sale... | 98 Lac | 13799.0 | thane | 473 sqft | Ready to Move | 3 out of 22 | Resale | ... | East | Garden/Park | Dosti Vihar | |
| **2** | 2 | 2 BHK Ready to Occupy Flat for sale in Sunrise... | Up for immediate sale is a 2 BHK apartment in ... | 1.40 Cr | 17500.0 | thane | 779 sqft | Ready to Move | 10 out of 29 | Resale | ... | East | Garden/Park | Sunrise by Kalpataru | |
| **3** | 3 | 1 BHK Ready to Occupy Flat for sale Kasheli | This beautiful 1 BHK Flat is available for sal... | 25 Lac | NaN | thane | 530 sqft | Ready to Move | 1 out of 3 | Resale | ... | NaN | NaN | NaN | |
| **4** | 4 | 2 BHK Ready to Occupy Flat for sale in TenX Ha... | This lovely 2 BHK Flat in Pokhran Road, Thane ... | 1.60 Cr | 18824.0 | thane | 635 sqft | Ready to Move | 20 out of 42 | Resale | ... | West | Garden/Park, Main Road | TenX Habitat Raymond Realty | |

5 rows × 21 columns

```
# Step 1: Extract numeric Carpet Area
df['Carpet Area'] = df['Carpet Area'].str.extract(r'(\d+)').astype(float)

# Step 2: Estimate house price
df['Amount'] = df['Carpet Area'] * df['PricePerSqFt']

# Step 3: Drop rows with missing values in key columns
df.dropna(subset=['Amount', 'Carpet Area'], inplace=True)

# Step 4: Preview updated dataset
df[['Carpet Area', 'PricePerSqFt', 'Amount']].head()
```

|   | Carpet Area | PricePerSqFt | Amount |
|---|---|---|---|
| **0** | 500.0 | 6000.0 | 3000000.0 |
| **1** | 473.0 | 13799.0 | 6526927.0 |
| **2** | 779.0 | 17500.0 | 13632500.0 |
| **4** | 635.0 | 18824.0 | 11953240.0 |
| **6** | 550.0 | 2538.0 | 1395900.0 |

```
# Step 1: Convert 'Bathroom' and 'Balcony' to numeric
df['Bathroom'] = pd.to_numeric(df['Bathroom'], errors='coerce')
df['Balcony'] = pd.to_numeric(df['Balcony'], errors='coerce')

# Step 2: Extract numeric floor number (e.g., '10 out of 15' → 10)
df['Floor_Level'] = df['Floor'].str.extract(r'(\d+)').astype(float)

# Step 3: Simplify Car Parking values (e.g., '1 Open', '1 Covered' → 1)
df['Car Parking'] = df['Car Parking'].str.extract(r'(\d+)').astype(float)

# Step 4: Drop original Floor column if not needed
df.drop(columns=['Floor'], inplace=True)

# Step 5: Preview cleaned data
df[['Bathroom', 'Balcony', 'Floor_Level', 'Car Parking']].head()
```

|   | Bathroom | Balcony | Floor_Level | Car Parking |
|---|---|---|---|---|
| **0** | 1.0 | 2.0 | 10.0 | NaN |
| **1** | 2.0 | NaN | 3.0 | 1.0 |
| **2** | 2.0 | NaN | 10.0 | 1.0 |
| **4** | 2.0 | NaN | 20.0 | 1.0 |
| **6** | 1.0 | NaN | 4.0 | NaN |

```
# Step 1: Define input features (X) and target variable (y)
X = df[['Carpet Area', 'Status', 'Transaction', 'Furnishing',
        'facing', 'overlooking', 'Society', 'Bathroom', 'Balcony',
        'Car Parking', 'Ownership', 'Floor_Level', 'location']]

y = df['Amount']  # Target: predicted house price

# Step 2: Identify categorical and numerical columns
cat_cols = X.select_dtypes(include='object').columns.tolist()
num_cols = X.select_dtypes(include=['int64', 'float64']).columns.tolist()

# Step 3: Print the feature types for verification
print("Categorical Columns:", cat_cols)
print("Numerical Columns:", num_cols)
```

```
Categorical Columns: ['Status', 'Transaction', 'Furnishing', 'facing', 'overlooking', 'Society', 'Ownership', 'location']
Numerical Columns: ['Carpet Area', 'Bathroom', 'Balcony', 'Car Parking', 'Floor_Level']
```

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

# Step 1: Define imputers
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(drop='first', handle_unknown='ignore'))
])

# Step 2: Full preprocessing pipeline with imputers
```

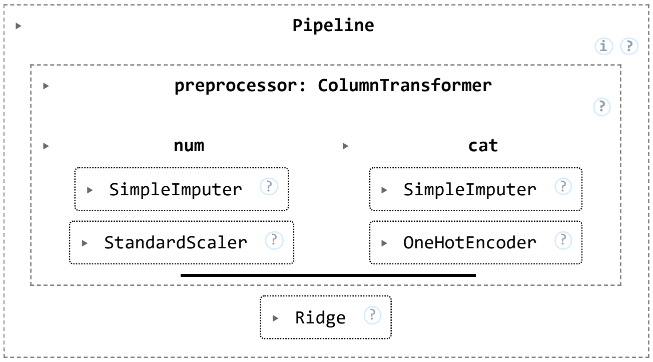
```
preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, num_cols),
    ('cat', categorical_transformer, cat_cols)
])
```

```
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.pipeline import Pipeline

# Linear Regression pipeline
lr_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

# Ridge Regression pipeline
ridge_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', Ridge(alpha=1.0))
])

# Train models
lr_pipeline.fit(X_train, y_train)
ridge_pipeline.fit(X_train, y_train)
```



```
from sklearn.metrics import mean_squared_error
import numpy as np

# Step 1: Predict on the test set
y_pred_lr = lr_pipeline.predict(X_test)
y_pred_ridge = ridge_pipeline.predict(X_test)

# Step 2: Define a reusable evaluation function
def evaluate_model(name, y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    print(f"{name}:\n  MSE: {mse:.2f}\n  RMSE: {rmse:.2f}\n")

# Step 3: Evaluate both models
evaluate_model("Linear Regression", y_test, y_pred_lr)
evaluate_model("Ridge Regression", y_test, y_pred_ridge)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_encoders.py:246: UserWarning: Found unknown categories in columns [5] dur
  warnings.warn(
Linear Regression:
  MSE: 1135513265410168.50
  RMSE: 33697377.72

Ridge Regression:
  MSE: 1129055557167808.25
  RMSE: 33601421.95

/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_encoders.py:246: UserWarning: Found unknown categories in columns [5] dur
  warnings.warn(
```