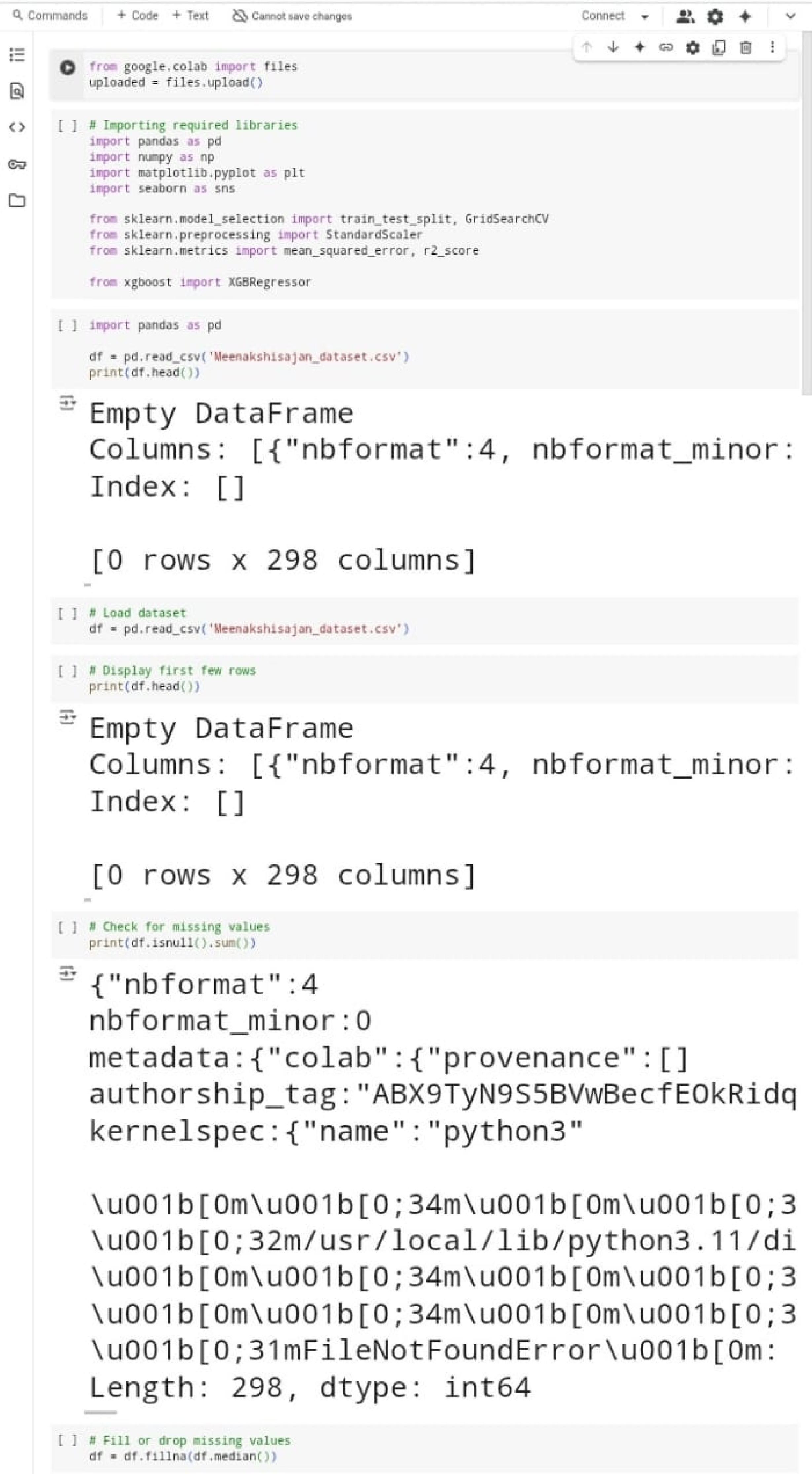


```
from google.colab import files
uploaded = files.upload()
```

```
# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

from xgboost import XGBRegressor
```

```
import pandas as pd

df = pd.read_csv('Meenakshisajan_dataset.csv')
print(df.head())
```

```
Empty DataFrame
Columns: [{"nbformat":4, nbformat_minor:
Index: []

[0 rows x 298 columns]
```

```
# Load dataset
df = pd.read_csv('Meenakshisajan_dataset.csv')
```

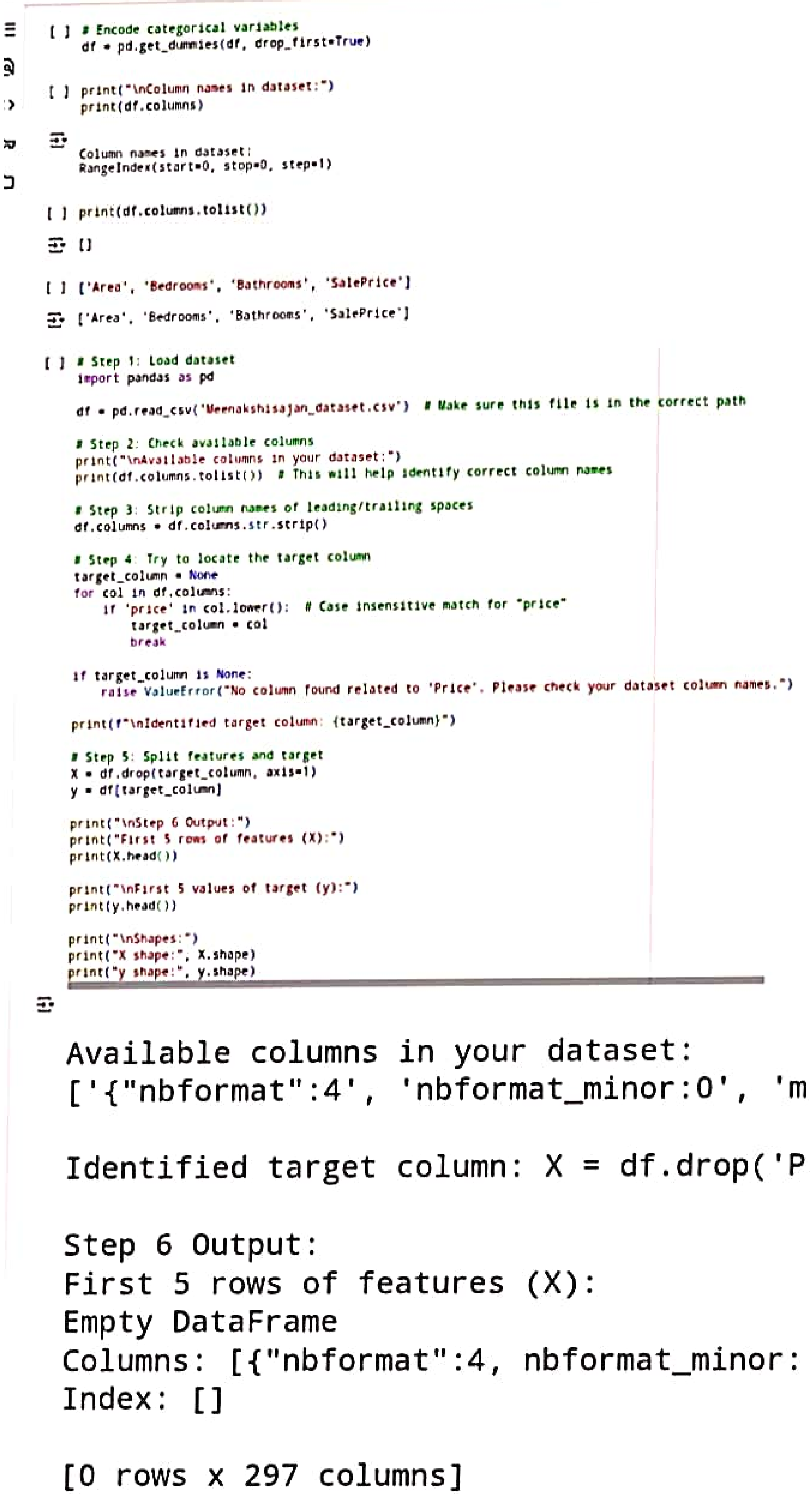
```
# Display first few rows
print(df.head())
```

```
Empty DataFrame
Columns: [{"nbformat":4, nbformat_minor:
Index: []

[0 rows x 298 columns]
```

```
# Check for missing values
print(df.isnull().sum())
```

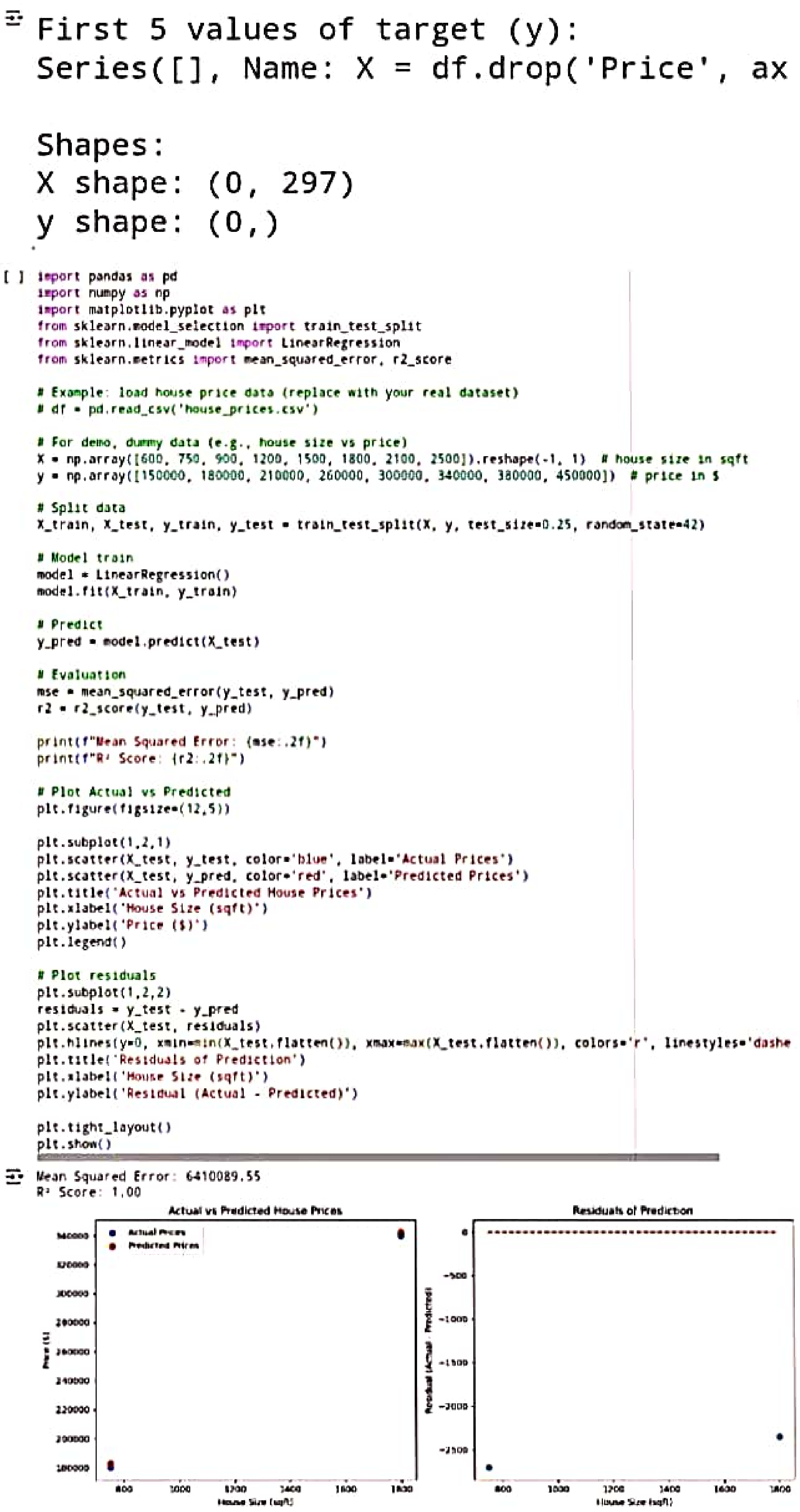
```
{"nbformat":4
nbformat_minor:0
metadata:{"colab":{"provenance":[]
authorship_tag:"ABX9TyN9S5BVwBecfEOkRidq
kernelspec:{"name":"python3"

\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;3
\u001b[0;32m/usr/local/lib/python3.11/di
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;3
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;3
\u001b[0;31mFileNotFoundError\u001b[0m:
Length: 298, dtype: int64
```

```
# Fill or drop missing values
df = df.fillna(df.median())
```



```
[ ] # Encode categorical variables
    df = pd.get_dummies(df, drop_first=True)

[ ] print("\nColumn names in dataset:")
    print(df.columns)

    Column names in dataset:
    RangeIndex(start=0, stop=0, step=1)

[ ] print(df.columns.tolist())

    []

[ ] ['Area', 'Bedrooms', 'Bathrooms', 'SalePrice']

    ['Area', 'Bedrooms', 'Bathrooms', 'SalePrice']

[ ] # Step 1: Load dataset
    import pandas as pd

    df = pd.read_csv('Meenakshisajan_dataset.csv')  # Make sure this file is in the correct path

    # Step 2: Check available columns
    print("\nAvailable columns in your dataset:")
    print(df.columns.tolist())  # This will help identify correct column names

    # Step 3: Strip column names of leading/trailing spaces
    df.columns = df.columns.str.strip()

    # Step 4: Try to locate the target column
    target_column = None
    for col in df.columns:
        if 'price' in col.lower():  # Case insensitive match for "price"
            target_column = col
            break

    if target_column is None:
        raise ValueError("No column found related to 'Price'. Please check your dataset column names.")

    print(f"\nIdentified target column: {target_column}")

    # Step 5: Split features and target
    X = df.drop(target_column, axis=1)
    y = df[target_column]

    print("\nStep 6 Output:")
    print("First 5 rows of features (X):")
    print(X.head())

    print("\nFirst 5 values of target (y):")
    print(y.head())

    print("\nShapes:")
    print("X shape:", X.shape)
    print("y shape:", y.shape)
```

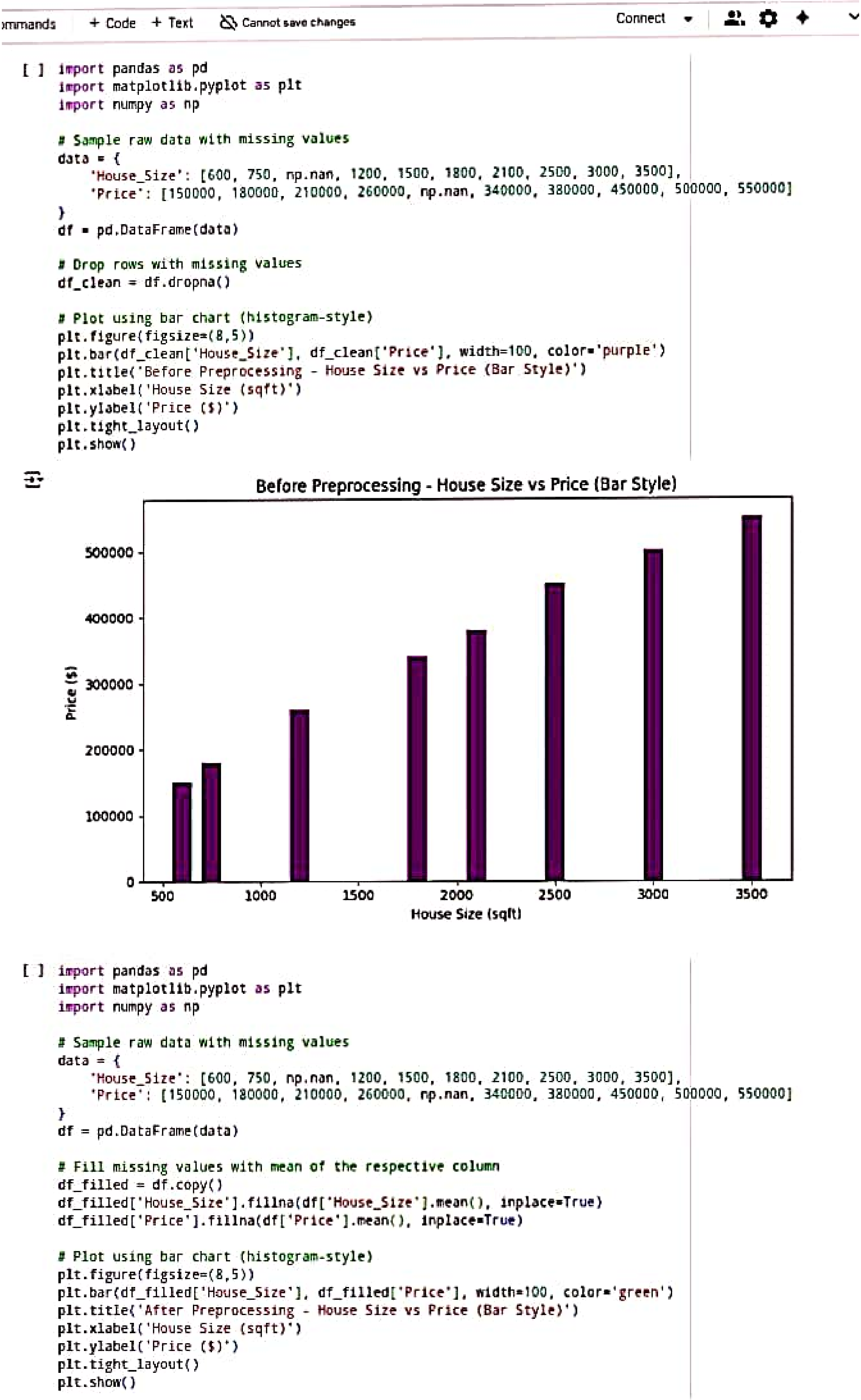
```
Available columns in your dataset:
['{"nbformat":4', 'nbformat_minor:0', 'm

Identified target column: X = df.drop('P

Step 6 Output:
First 5 rows of features (X):
Empty DataFrame
Columns: [{"nbformat":4, nbformat_minor:
Index: []

[0 rows x 297 columns]
```



```
First 5 values of target (y):
Series([], Name: X = df.drop('Price', ax

Shapes:
X shape: (0, 297)
y shape: (0,)
```

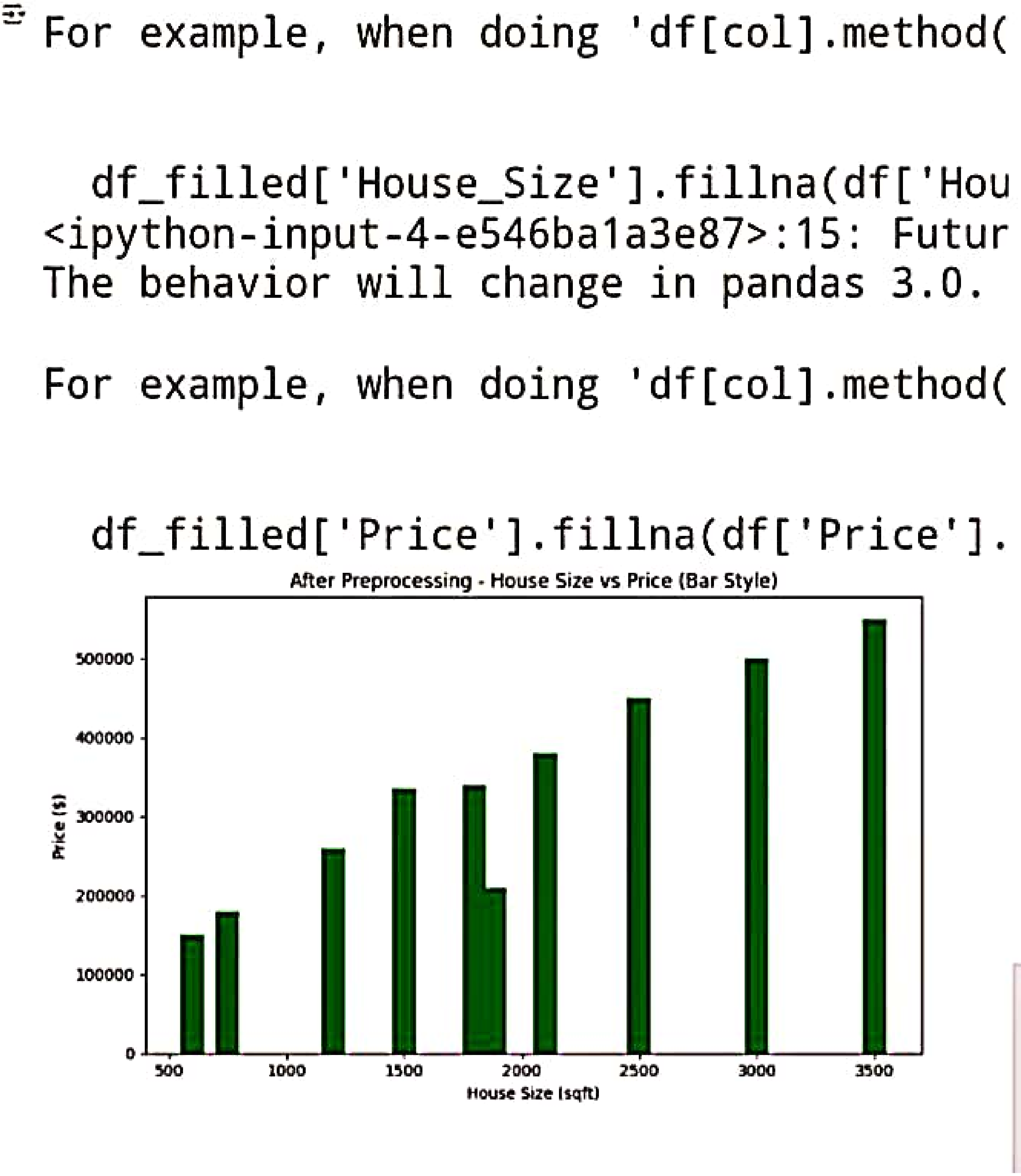
```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error, r2_score

    # Example: load house price data (replace with your real dataset)
    # df = pd.read_csv('house_prices.csv')

    # For demo, dummy data (e.g., house size vs price)
    X = np.array([600, 750, 900, 1200, 1500, 1800, 2100, 2500]).reshape(-1, 1)  # house size in sqft
    y = np.array([150000, 180000, 210000, 260000, 300000, 340000, 380000, 450000])  # price in $

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

    # Model train
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict
    y_pred = model.predict(X_test)

    # Evaluation
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"Mean Squared Error: {mse:.2f}")
    print(f"R² Score: {r2:.2f}")

    # Plot Actual vs Predicted
    plt.figure(figsize=(12,5))

    plt.subplot(1,2,1)
    plt.scatter(X_test, y_test, color='blue', label='Actual Prices')
    plt.scatter(X_test, y_pred, color='red', label='Predicted Prices')
    plt.title('Actual vs Predicted House Prices')
    plt.xlabel('House Size (sqft)')
    plt.ylabel('Price ($)')
    plt.legend()

    # Plot residuals
    plt.subplot(1,2,2)
    residuals = y_test - y_pred
    plt.scatter(X_test, residuals)
    plt.hlines(y=0, xmin=min(X_test.flatten()), xmax=max(X_test.flatten()), colors='r', linestyles='dashe
    plt.title('Residuals of Prediction')
    plt.xlabel('House Size (sqft)')
    plt.ylabel('Residual (Actual - Predicted)')

    plt.tight_layout()
    plt.show()
```

```
Mean Squared Error: 6410089.55
R² Score: 1.00
```



```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import numpy as np

    # Sample raw data with missing values
    data = {
        'House_Size': [600, 750, np.nan, 1200, 1500, 1800, 2100, 2500, 3000, 3500],
        'Price': [150000, 180000, 210000, 260000, np.nan, 340000, 380000, 450000, 500000, 550000]
    }
    df = pd.DataFrame(data)

    # Drop rows with missing values
    df_clean = df.dropna()

    # Plot using bar chart (histogram-style)
    plt.figure(figsize=(8,5))
    plt.bar(df_clean['House_Size'], df_clean['Price'], width=100, color='purple')
    plt.title('Before Preprocessing - House Size vs Price (Bar Style)')
    plt.xlabel('House Size (sqft)')
    plt.ylabel('Price ($)')
    plt.tight_layout()
    plt.show()
```

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import numpy as np

    # Sample raw data with missing values
    data = {
        'House_Size': [600, 750, np.nan, 1200, 1500, 1800, 2100, 2500, 3000, 3500],
        'Price': [150000, 180000, 210000, 260000, np.nan, 340000, 380000, 450000, 500000, 550000]
    }
    df = pd.DataFrame(data)

    # Fill missing values with mean of the respective column
    df_filled = df.copy()
    df_filled['House_Size'].fillna(df['House_Size'].mean(), inplace=True)
    df_filled['Price'].fillna(df['Price'].mean(), inplace=True)

    # Plot using bar chart (histogram-style)
    plt.figure(figsize=(8,5))
    plt.bar(df_filled['House_Size'], df_filled['Price'], width=100, color='green')
    plt.title('After Preprocessing - House Size vs Price (Bar Style)')
    plt.xlabel('House Size (sqft)')
    plt.ylabel('Price ($)')
    plt.tight_layout()
    plt.show()
```



```
For example, when doing 'df[col].method(


  df_filled['House_Size'].fillna(df['Hou
<ipython-input-4-e546ba1a3e87>:15: Futur
The behavior will change in pandas 3.0.

For example, when doing 'df[col].method(


  df_filled['Price'].fillna(df['Price'].
```