PROJECT 9: PREDICTING HOUSE PRICE USING MACHINE LEARNING

PHASE 4: DEVELOPMENT PART 2

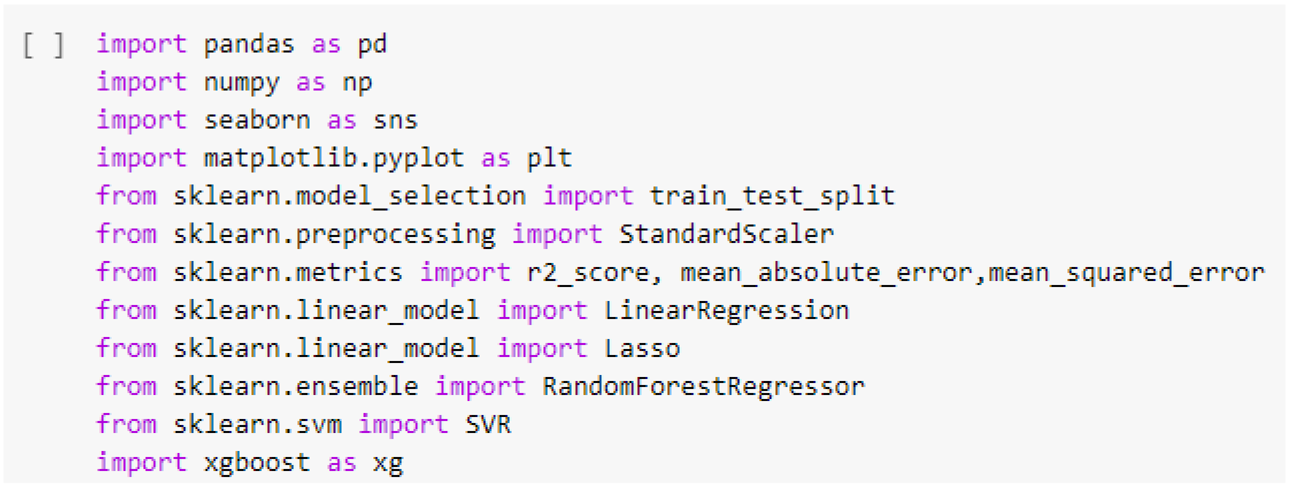
INTRODUCTION:

The development and evaluation of house price prediction models play a pivotal role in the real estate industry, aiding homeowners, buyers, and investors in making informed decisions. This process combines data science, machine learning, and statistical analysis to create accurate models that estimate property values.

PROGRAM:

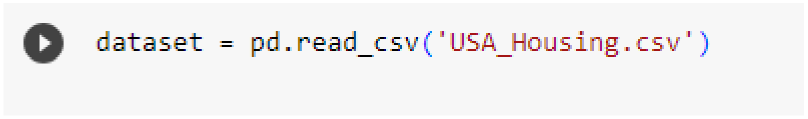
IMPORTING DATASET:

Loading data is a crucial step in any data analysis or machine learning task. It involves bringing external datasets into your programming environment so that you can manipulate, analyze, and draw insights from the data.



```
[ ]  import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error
     from sklearn.linear_model import LinearRegression
     from sklearn.linear_model import Lasso
     from sklearn.ensemble import RandomForestRegressor
     from sklearn.svm import SVR
     import xgboost as xg
```

LOAD DATASET:

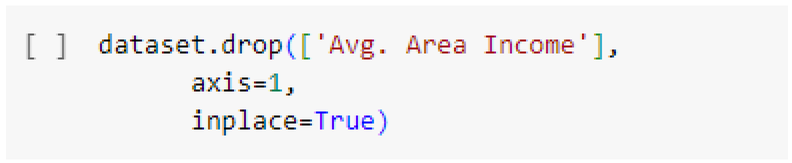


```
dataset = pd.read_csv('USA_Housing.csv')
```

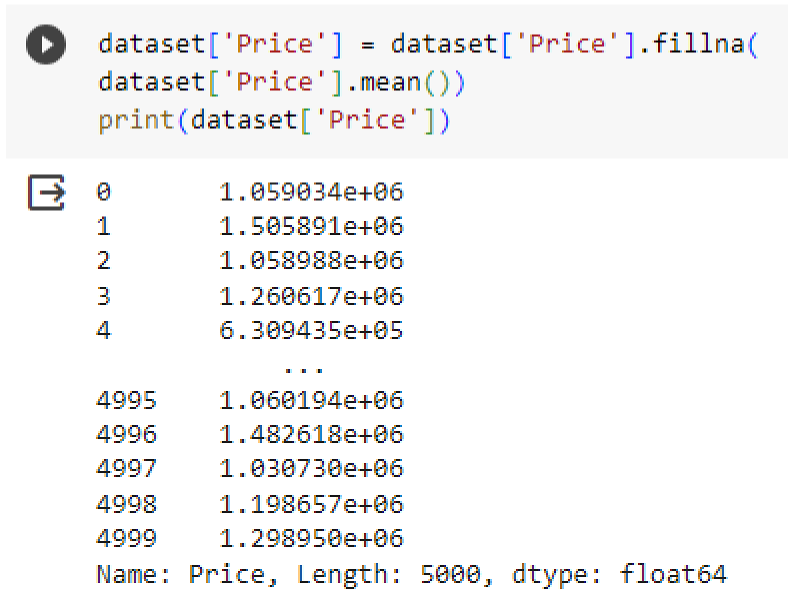
DATA CLEANING:

Data cleaning is a crucial step in the data preparation process. It involves identifying and correcting errors or inconsistencies in your data to ensure that it is accurate, reliable, and suitable for analysis.

- Data Collection
- Handling Missing Values
- Handling Duplicate Data
- Data Transformation

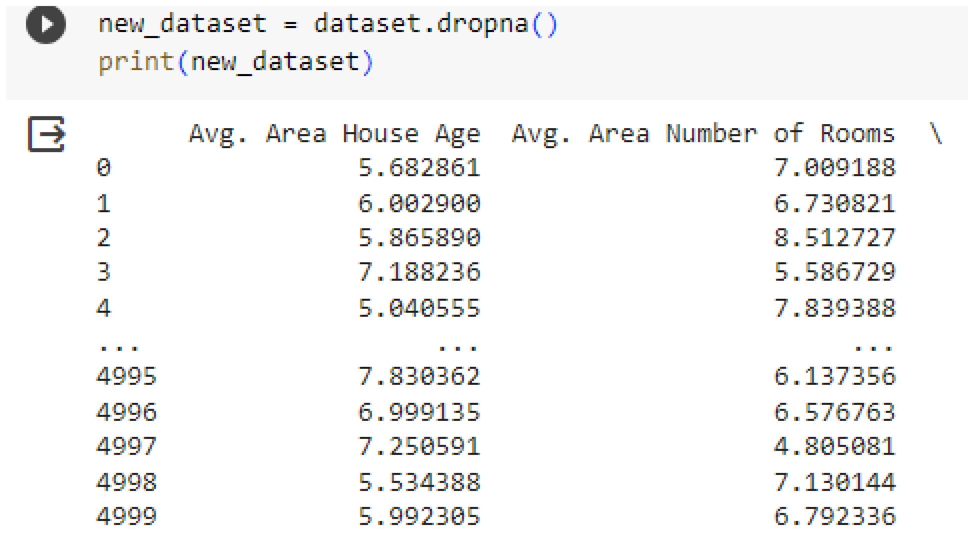


```
[ ]  dataset.drop(['Avg. Area Income'],
            axis=1,
            inplace=True)
```



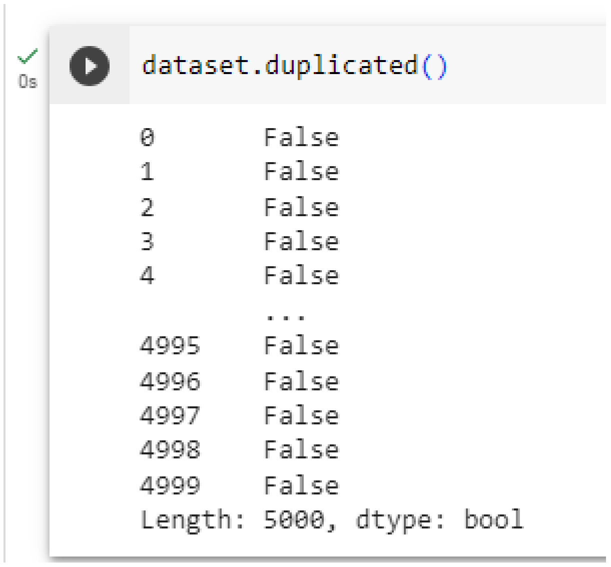
```
dataset['Price'] = dataset['Price'].fillna(
dataset['Price'].mean())
print(dataset['Price'])
```

```
0       1.059034e+06
1       1.505891e+06
2       1.058988e+06
3       1.260617e+06
4       6.309435e+05
            ...
4995    1.060194e+06
4996    1.482618e+06
4997    1.030730e+06
4998    1.198657e+06
4999    1.298950e+06
Name: Price, Length: 5000, dtype: float64
```



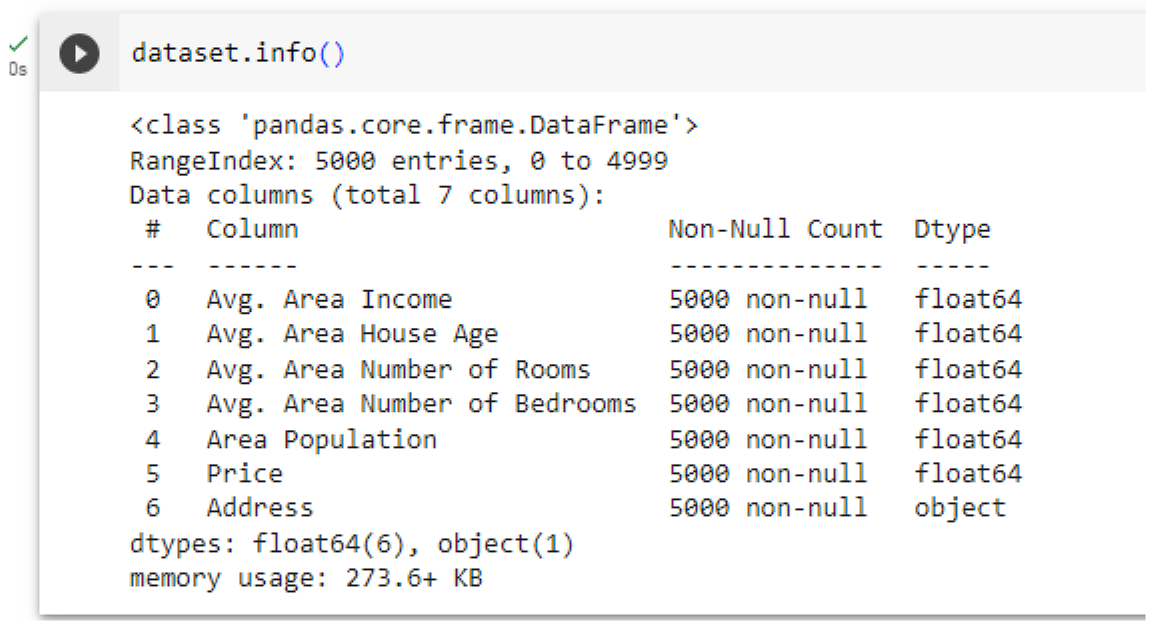
```
new_dataset = dataset.dropna()
print(new_dataset)
```

```
      Avg. Area House Age  Avg. Area Number of Rooms  \
0                5.682861                   7.009188
1                6.002900                   6.730821
2                5.865890                   8.512727
3                7.188236                   5.586729
4                5.040555                   7.839388
...                   ...                        ...
4995             7.830362                   6.137356
4996             6.999135                   6.576763
4997             7.250591                   4.805081
4998             5.534388                   7.130144
4999             5.992305                   6.792336
```



```
dataset.duplicated()
```

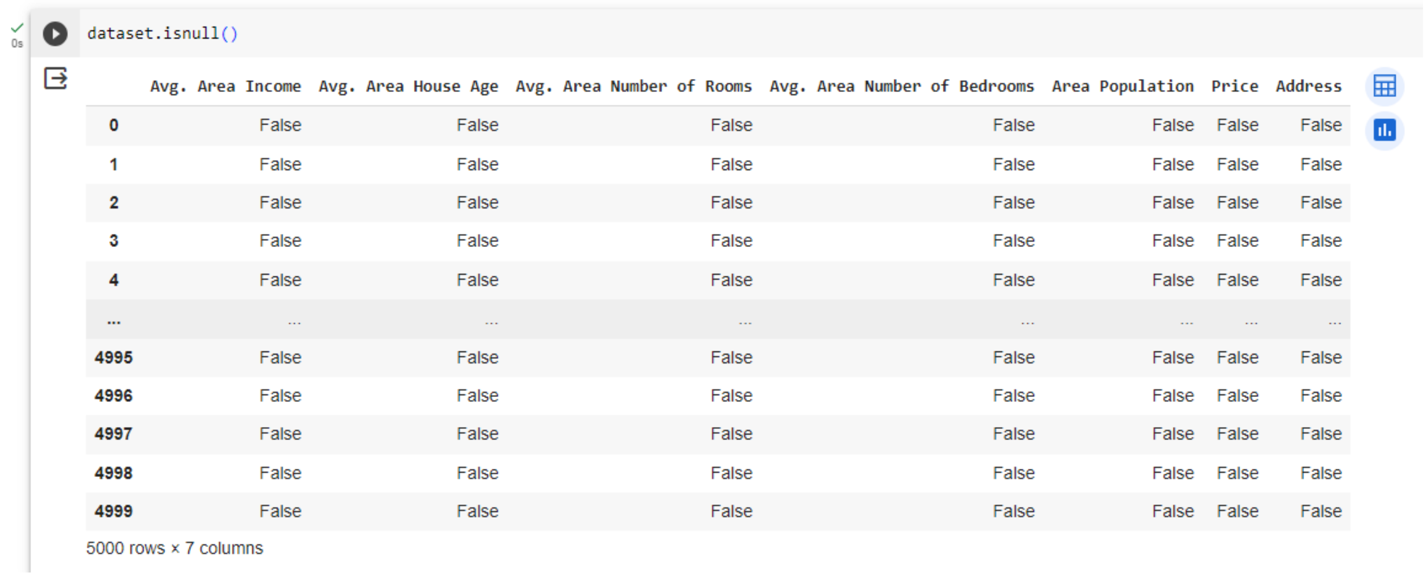
```
0       False
1       False
2       False
3       False
4       False
        ...
4995    False
4996    False
4997    False
4998    False
4999    False
Length: 5000, dtype: bool
```



```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

Data cleaning is the process of identifying and correcting errors, inconsistencies, and inaccuracies in datasets. It's a crucial step in the data analysis pipeline, as the quality of your analysis depends heavily on the quality of your data.



```python
dataset.isnull()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
4995	False	False	False	False	False	False	False
4996	False	False	False	False	False	False	False
4997	False	False	False	False	False	False	False
4998	False	False	False	False	False	False	False
4999	False	False	False	False	False	False	False

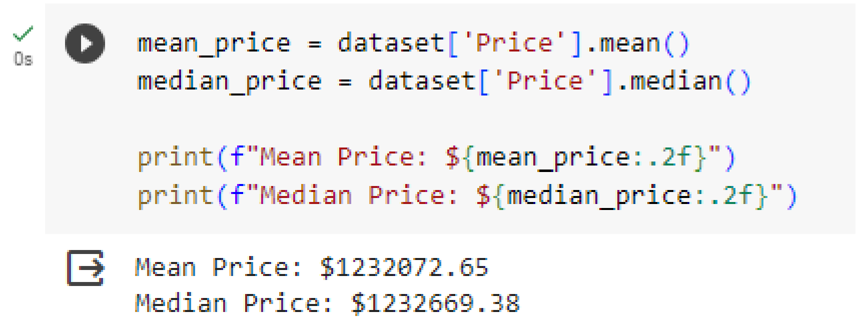
5000 rows × 7 columns

DATA ANALYSIS:

Data analysis is the process of inspecting, cleaning, transforming, and modeling data to uncover useful information, draw conclusions, and support decision-making.

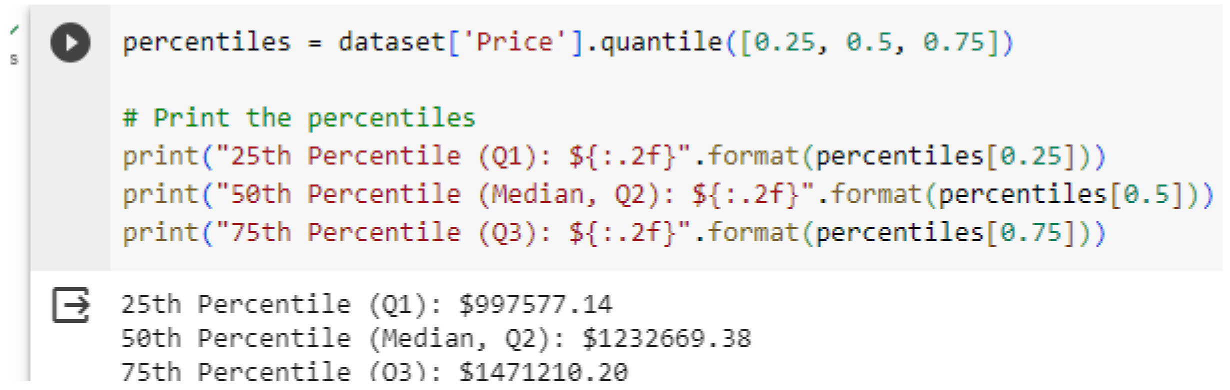
Data visualization

Exploratory Data Analysis



```python
mean_price = dataset['Price'].mean()
median_price = dataset['Price'].median()

print(f"Mean Price: ${mean_price:.2f}")
print(f"Median Price: ${median_price:.2f}")
```

```
Mean Price: $1232072.65
Median Price: $1232669.38
```

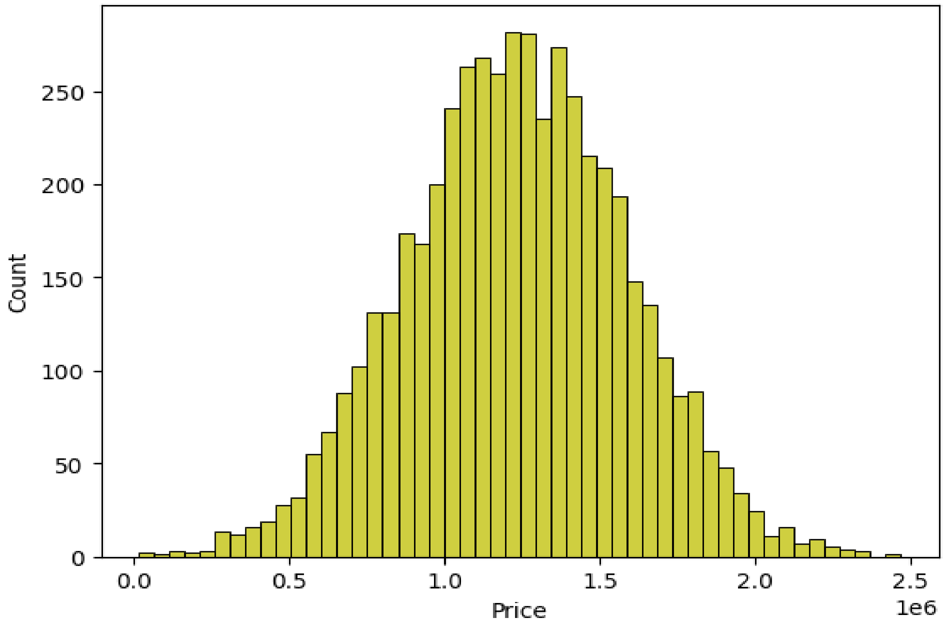


```python
percentiles = dataset['Price'].quantile([0.25, 0.5, 0.75])

# Print the percentiles
print("25th Percentile (Q1): ${:.2f}".format(percentiles[0.25]))
print("50th Percentile (Median, Q2): ${:.2f}".format(percentiles[0.5]))
print("75th Percentile (Q3): ${:.2f}".format(percentiles[0.75]))
```

```
25th Percentile (Q1): $997577.14
50th Percentile (Median, Q2): $1232669.38
75th Percentile (Q3): $1471210.20
```

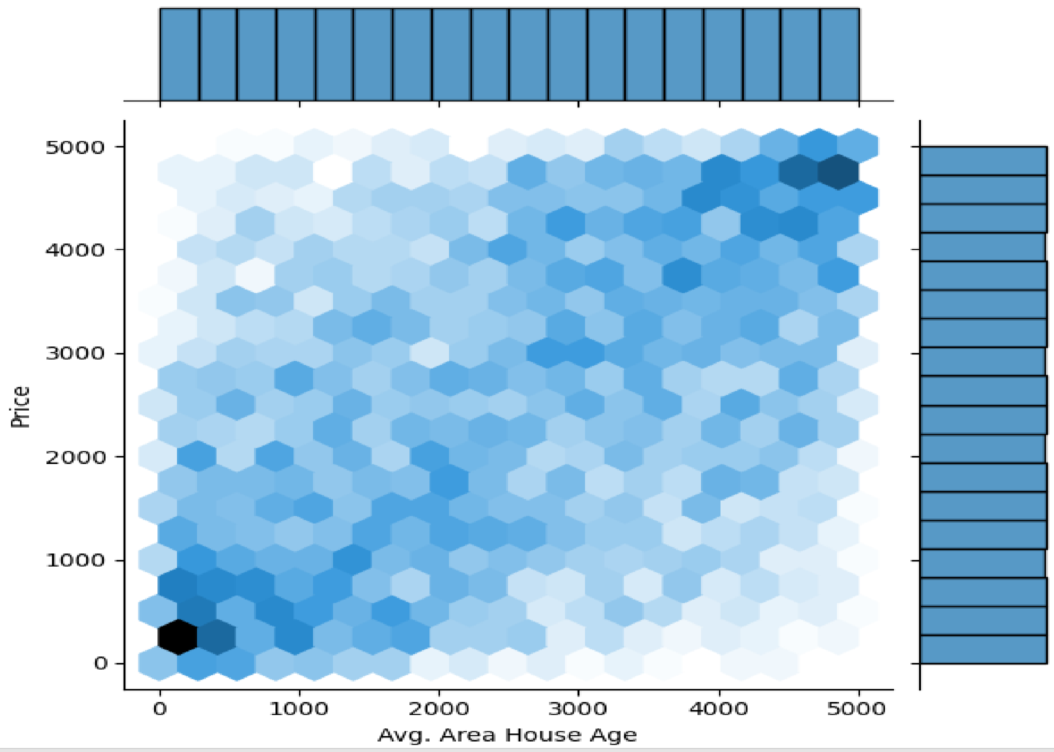
```
sns.histplot(dataset, x='Price', bins=50, color='y')
```

```
<Axes: xlabel='Price', ylabel='Count'>
```

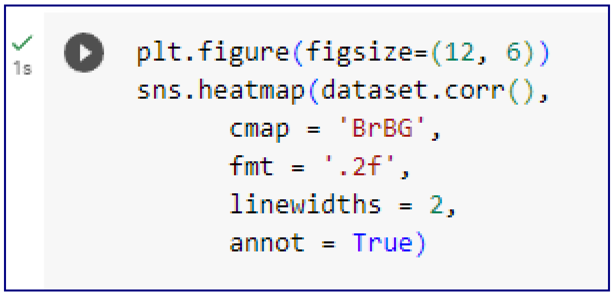


```
sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')
```

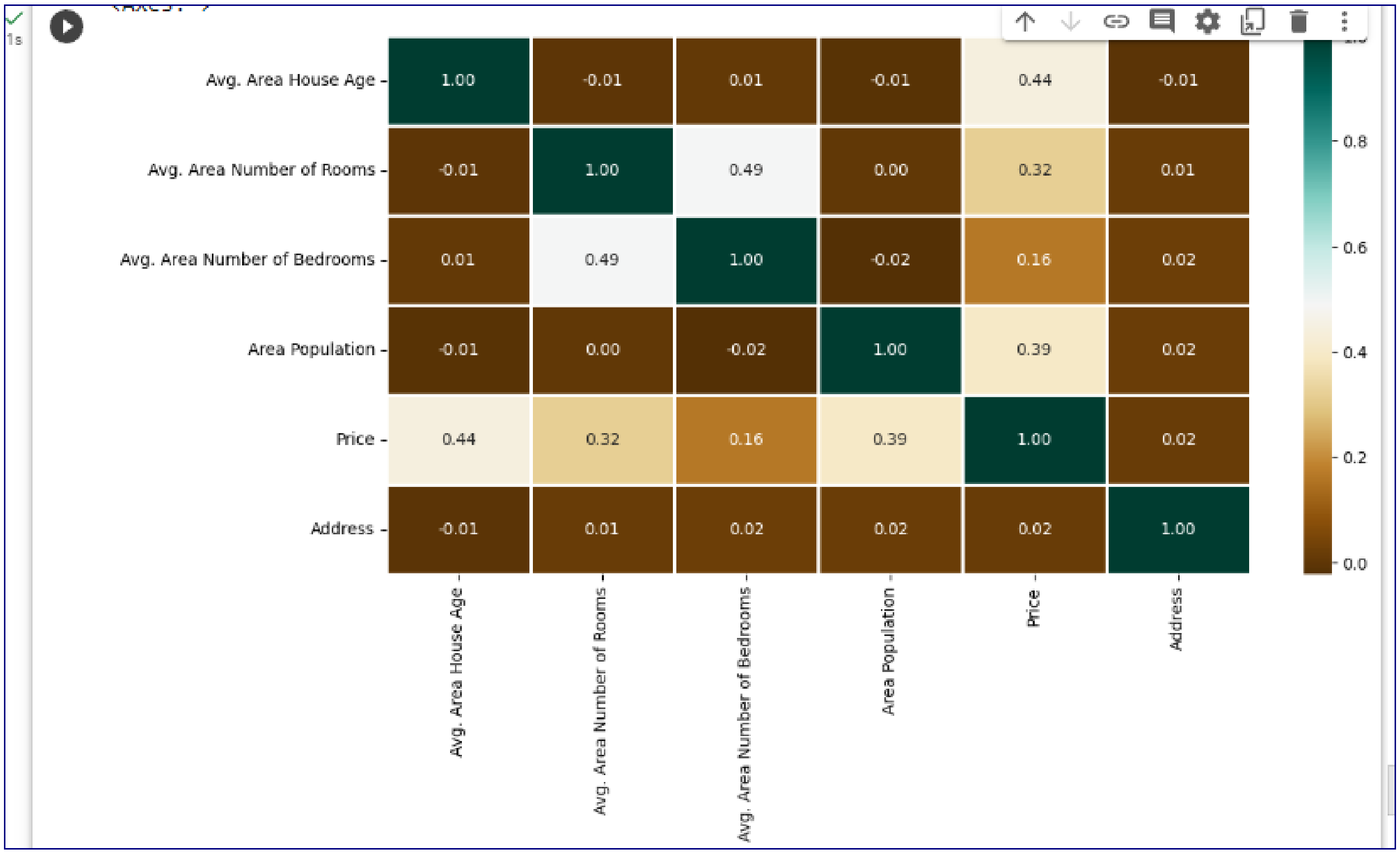
```
<seaborn.axisgrid.JointGrid at 0x7ab322e87580>
```



Data analysis is the process of inspecting, cleaning, transforming, and modeling data to discover useful information, draw meaningful conclusions, and support decision-making.



```
plt.figure(figsize=(12, 6))
sns.heatmap(dataset.corr(),
            cmap = 'BrBG',
            fmt = '.2f',
            linewidths = 2,
            annot = True)
```

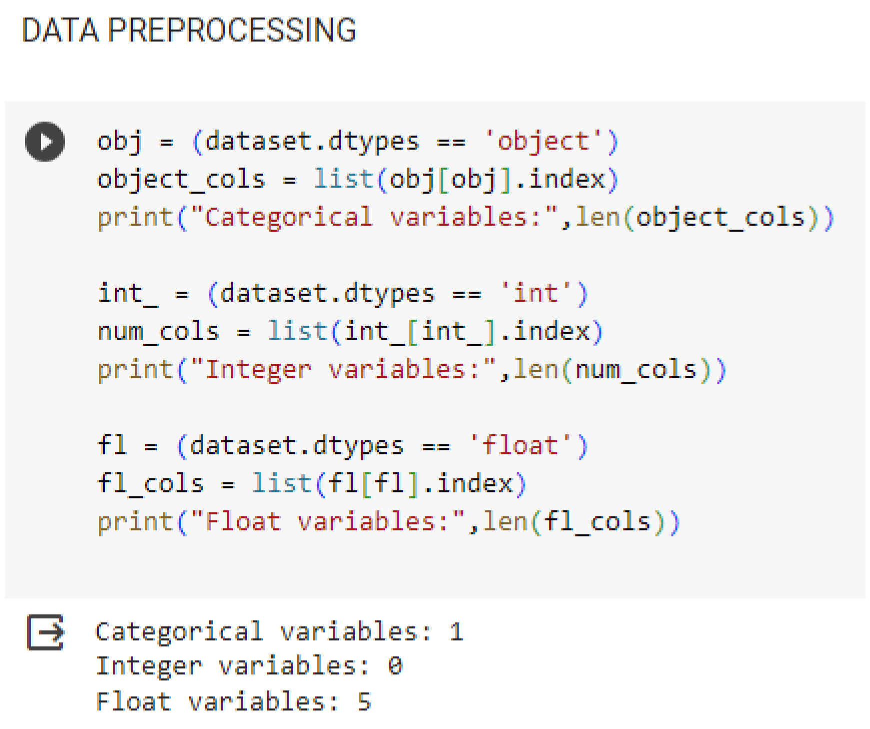


DATA PREPROCESSING:

It is defined as Collection, manipulation, and processing of collected data for the required use. It is a task of converting data from a given form to a much more usable and desired form i.e. making it more meaningful and informative. Using Machine Learning algorithms, mathematical modelling and statistical knowledge, this entire process can be automated.

- Data Transformation
- Feature Selection
- Handling Outliers
- Normalization and Scaling

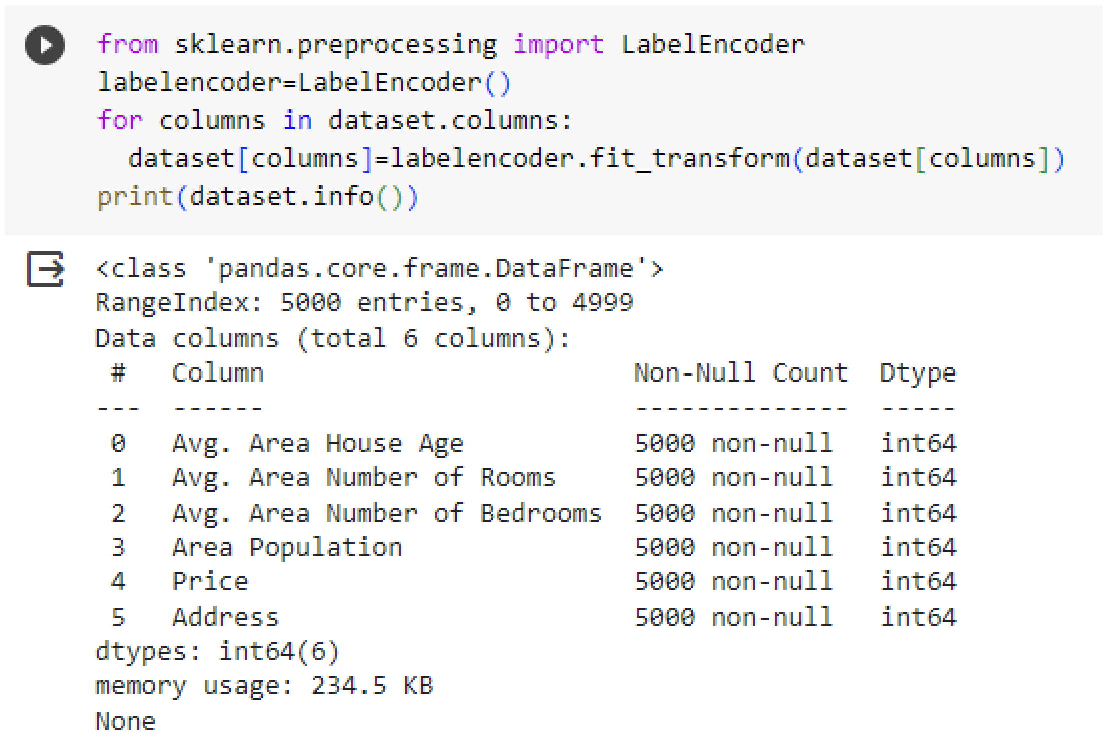
- Data Splitting
- Data Input



DATA PREPROCESSING

```
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (dataset.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))

fl = (dataset.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))
```

```
Categorical variables: 1
Integer variables: 0
Float variables: 5
```



```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
for columns in dataset.columns:
  dataset[columns]=labelencoder.fit_transform(dataset[columns])
print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Avg. Area House Age           5000 non-null   int64
 1   Avg. Area Number of Rooms     5000 non-null   int64
 2   Avg. Area Number of Bedrooms  5000 non-null   int64
 3   Area Population               5000 non-null   int64
 4   Price                         5000 non-null   int64
 5   Address                       5000 non-null   int64
dtypes: int64(6)
memory usage: 234.5 KB
None
```

Splitting the data:

To do this, you split your dataset into two main parts: a training set and a testing Set.

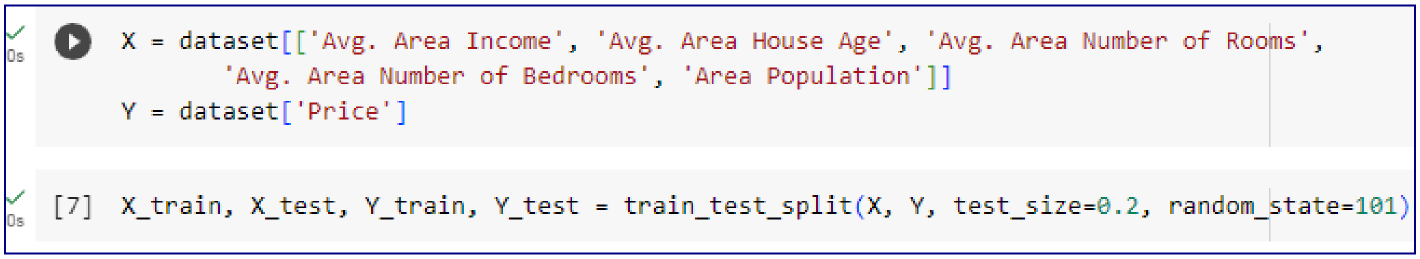
Training Dataset:

- The training dataset is a subset of your overall dataset that is used to train your machine learning model. It contains both the input features (often denoted as "X") and the corresponding target or output values (often denoted as "y").
- When you train a machine learning model, the algorithm learns patterns, relationships, and rules from the data in the training dataset. It uses this information to make predictions or classifications.
- The model adjusts its internal parameters during training to minimize the difference between its predictions and the actual target values in the training data. This process is known as model training or fitting.
- The training dataset is the data your model has seen and learned from. It is essential for building a predictive model.

Testing Dataset (or Validation Dataset):

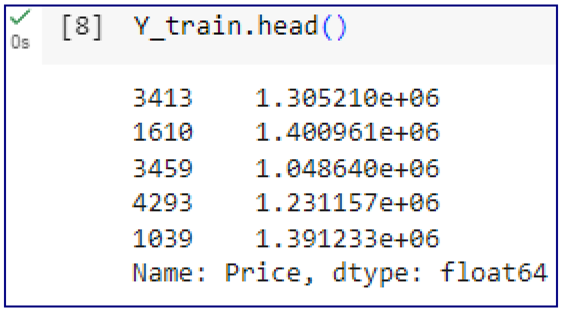
- The testing dataset is another subset of your overall dataset that is kept separate from the training dataset. It is used to evaluate the performance of your trained machine learning model.
- The testing dataset also contains input features ("X") and their corresponding target values ("y"), but the model has not seen this data during training.
- Once the model is trained, it is tested on the testing dataset to assess how well it generalizes to new, unseen data. This evaluation provides an indication of the model's performance and whether it can make accurate predictions on real-world data.
- The testing dataset helps you determine if your model has learned to make predictions or classifications that are not specific to the training data but instead can be applied to new, unseen examples.

The training dataset is used to teach the machine learning model, while the testing dataset is used to assess how well the model has learned and to estimate its performance on data it hasn't encountered before. Properly splitting the data into training and testing sets is crucial for avoiding overfitting, where the model memorizes the training data but fails to generalize to new data, and for ensuring that the model can make useful predictions in practice.

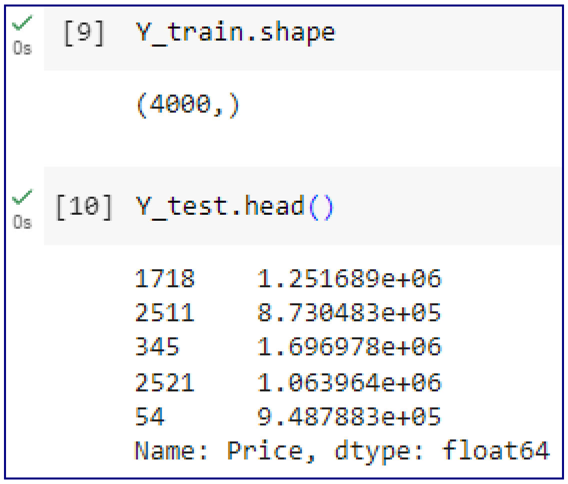


```
X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
Y = dataset['Price']
```

```
[7] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=101)
```



```
[8]  Y_train.head()

     3413    1.305210e+06
     1610    1.400961e+06
     3459    1.048640e+06
     4293    1.231157e+06
     1039    1.391233e+06
     Name: Price, dtype: float64
```

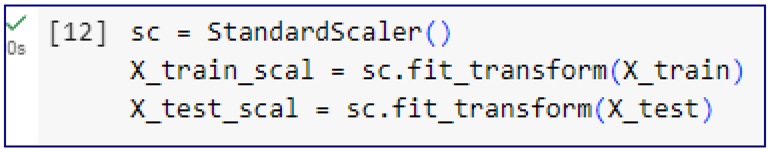


```
[9]  Y_train.shape

     (4000,)
```

```
[10] Y_test.head()

     1718    1.251689e+06
     2511    8.730483e+05
     345     1.696978e+06
     2521    1.063964e+06
     54      9.487883e+05
     Name: Price, dtype: float64
```

Standard scalar:

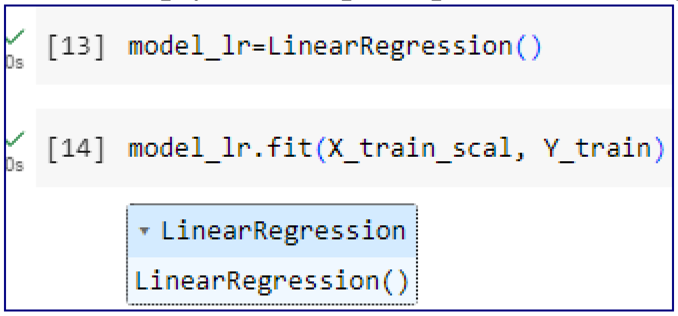
- Standard scaling is a preprocessing technique used in machine learning to standardize or normalize the feature values of a dataset.
- When you standard scale a feature, you transform it so that it has a mean (average) of 0 and a standard deviation of 1.
- This means that the values are rescaled so that they have a common scale and are centered around zero.



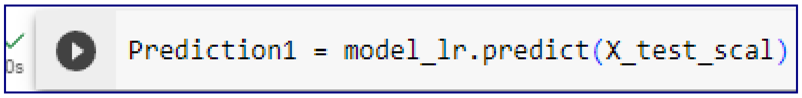
```
[12] sc = StandardScaler()
     X_train_scal = sc.fit_transform(X_train)
     X_test_scal = sc.fit_transform(X_test)
```

LINEAR REGRESSION:

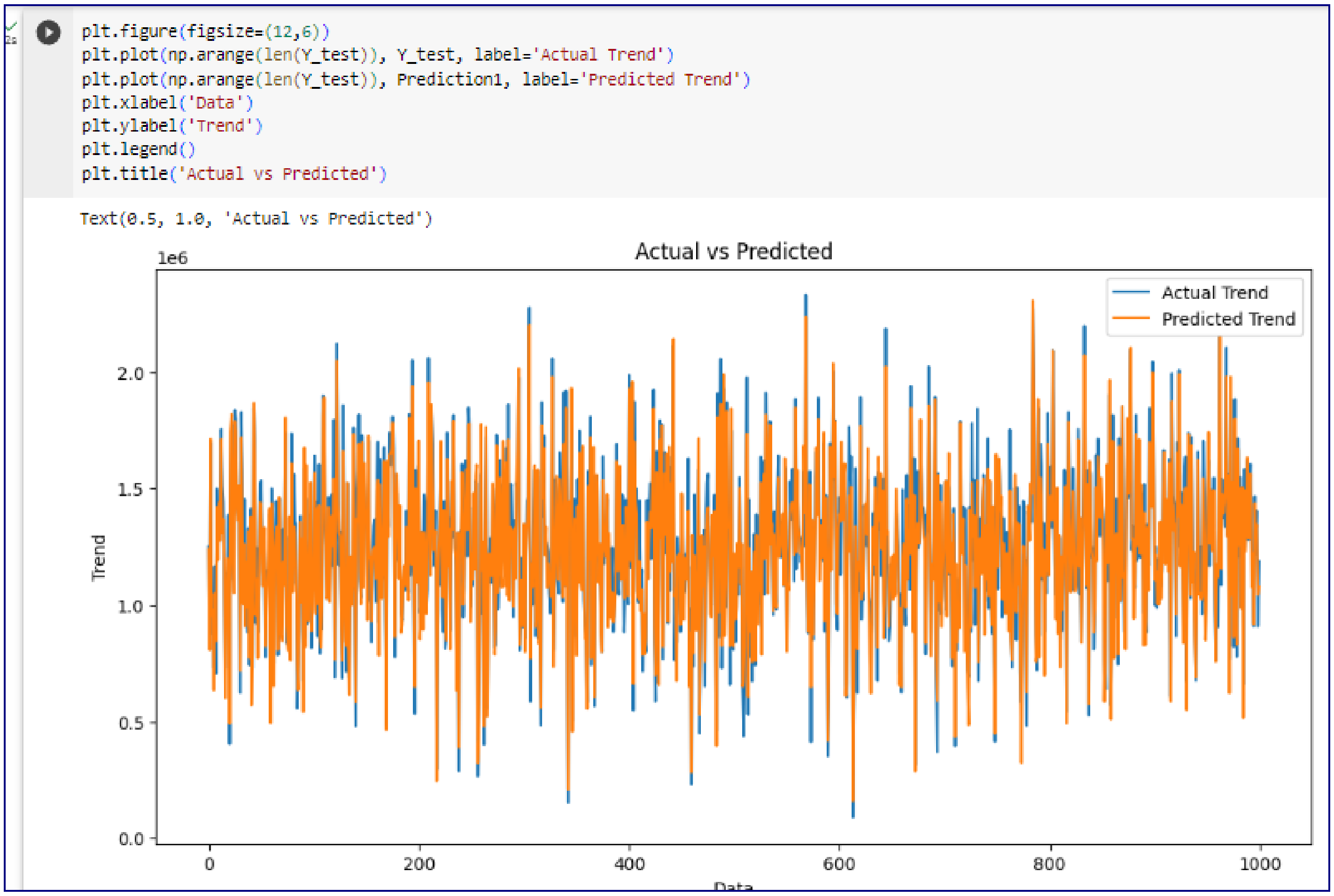
Linear regression is a commonly used machine learning technique for predicting house prices. In this context, it's often referred to as "linear regression for house price prediction" or simply "house price prediction using linear regression



```
[13] model_lr=LinearRegression()

[14] model_lr.fit(X_train_scal, Y_train)

    ▾ LinearRegression
    LinearRegression()
```



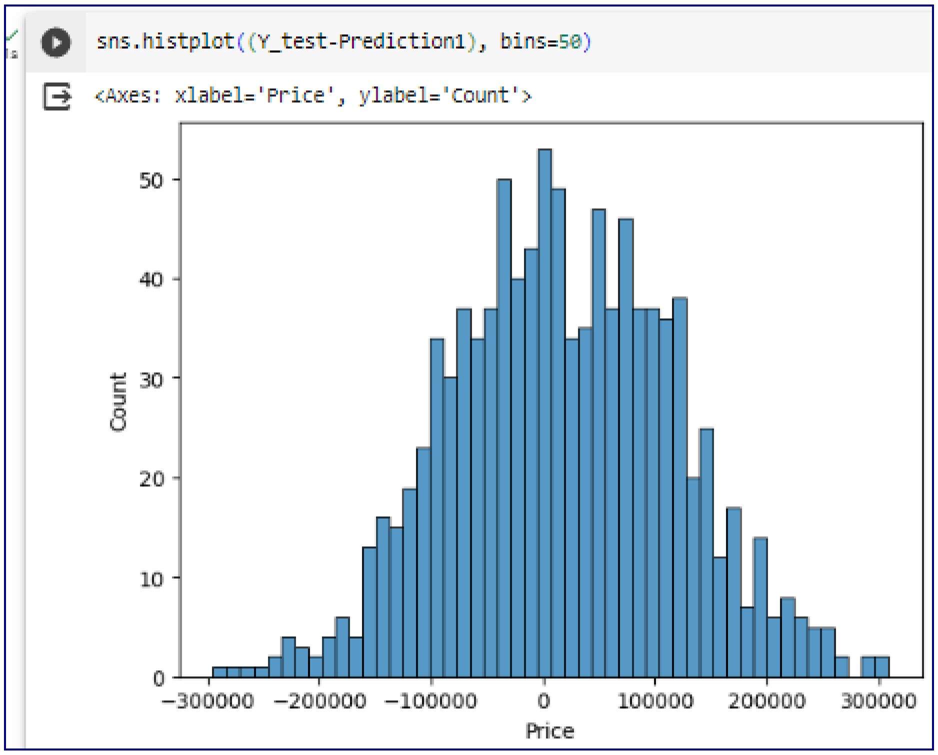
```
Prediction1 = model_lr.predict(X_test_scal)
```



```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```

Once trained, the model can make predictions for new, unseen data. You input a value for x, and the model predicts the corresponding y.

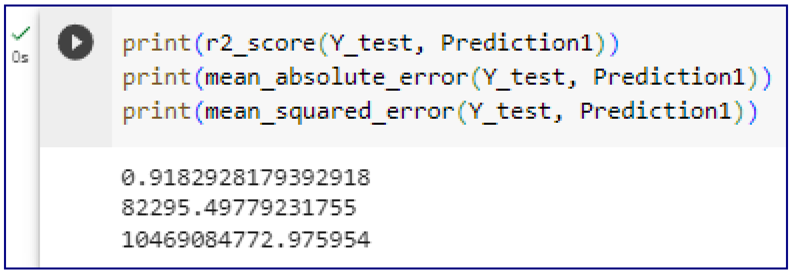


```
sns.histplot((Y_test-Prediction1), bins=50)
```

```
<Axes: xlabel='Price', ylabel='Count'>
```

EVALUATION:

Common metrics for evaluating linear regression models include Mean Squared Error (MSE) and R-squared. MSE measures the average squared difference between predicted and actual values, while R-squared represents the proportion of the variance in the dependent variable that is predictable from the independent variables.



```
print(r2_score(Y_test, Prediction1))
print(mean_absolute_error(Y_test, Prediction1))
print(mean_squared_error(Y_test, Prediction1))
```

```
0.9182928179392918
82295.49779231755
10469084772.975954
```

Linear regression is a great starting point for many predictive modeling tasks, and it forms the foundation for more complex models. It's widely used in various fields due to its simplicity and interpretability.