**ELECTRICITY PRICES PREDICTION USING MACHINE LEARNING**

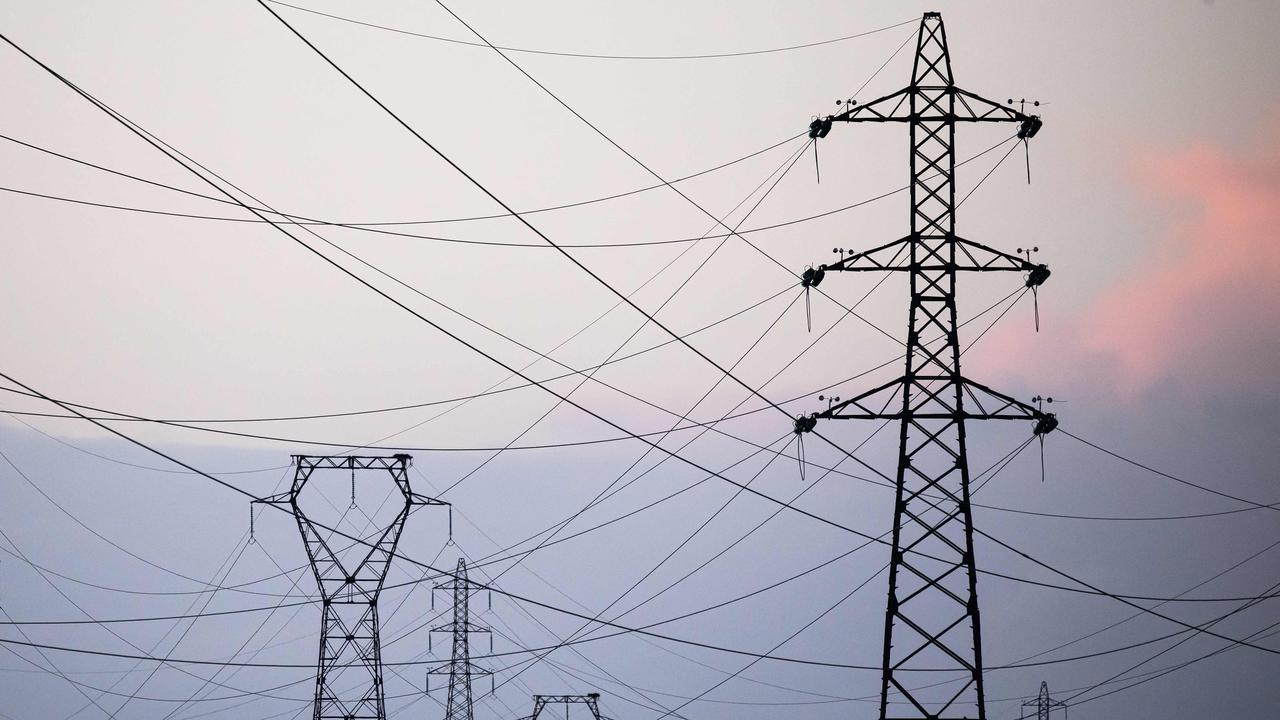
**Phase 3 submission document**

**Project title:** electricity prices prediction

**Phase 3: development part 1**

**Topic:** start building the electricity prices prediction model by

Loading and pre-processing the dataset.

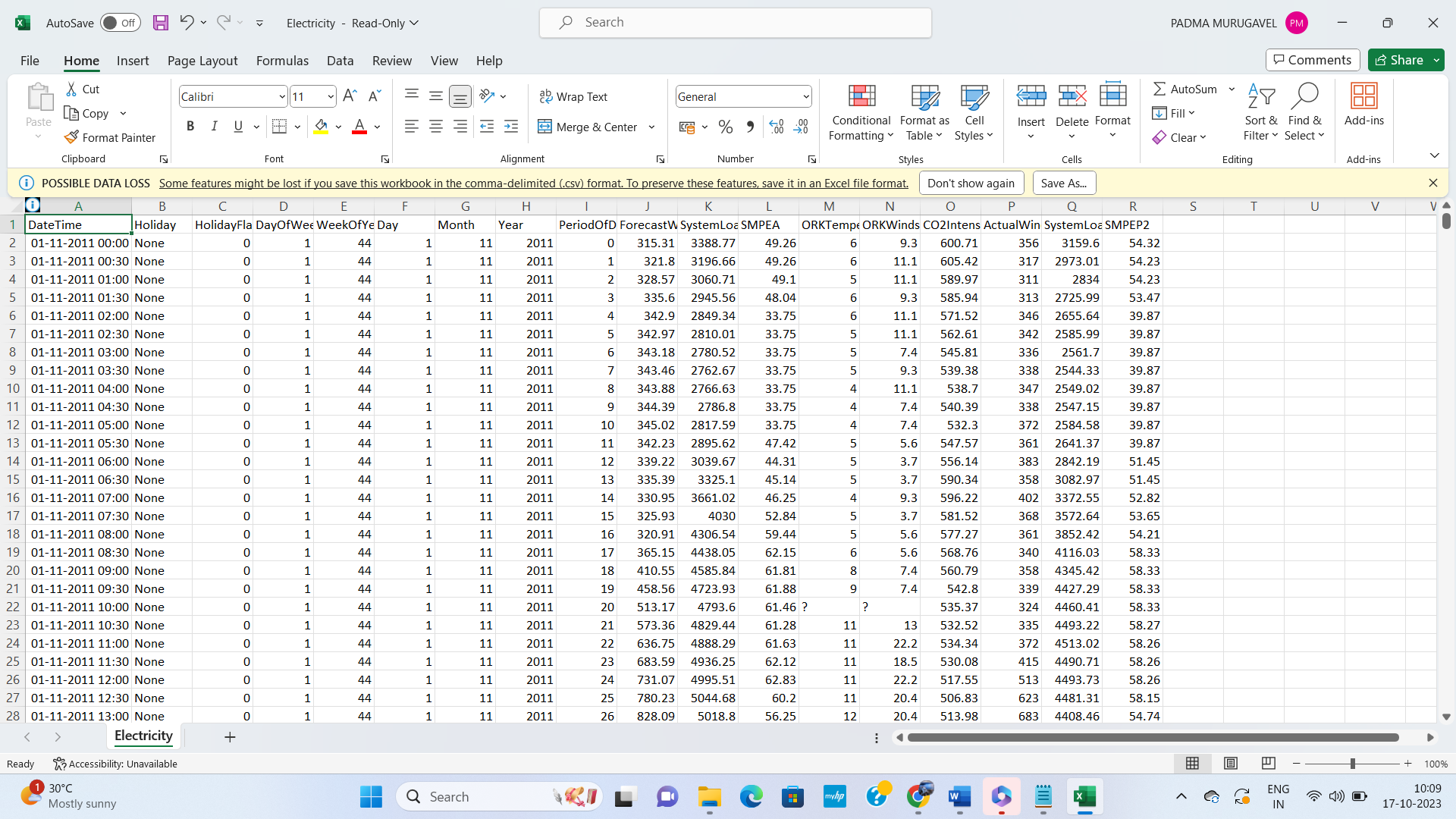


**Electricity prices prediction Using Machine Learning**

Introduction:

* Electricity price prediction is a critical and challenging task in the energy industry, with significant implications for various stakeholders, including energy providers, consumers, and investors.
* This introduction briefing provides an overview of electricity price prediction, its importance, and the factors involved.
* Electricity prices are dynamic and subject to a multitude of influences. Accurate prediction of electricity prices is essential for making informed decisions in the energy sector. It enables stakeholders to optimize their energy consumption, manage costs, and invest strategically.
* Electricity price prediction is a critical tool for energy stakeholders, enabling them to make informed decisions in a dynamic and rapidly evolving market.
* By understanding the key factors influencing prices and employing advanced prediction methods, stakeholders can gain a competitive edge and manage their energy resources effectively.

**Given data set:**



**Necessary step to follow:**

**1.Import Libraries:**

Start by importing the necessary libraries:

**Program:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

**2.Load the Dataset:**

Load your dataset into a Pandas DataFrame. You can typically find

house price datasets in CSV format, but you can adapt this code to otherformats as needed.

**Program:**

df = pd.read\_csv(' E:\USA\_Housing.csv ')

Pd.read()

**3. Exploratory Data Analysis (EDA):**

Perform EDA to understand your data better. This includes

checking for missing values, exploring the data's statistics, and

visualizing it to identify patterns.

**Program:**

# Check for missing values

print(df.isnull().sum())

# Explore statistics

print(df.describe())

# Visualize the data (e.g., histograms, scatter plots, etc.)

**4. Feature Engineering:**

Depending on your dataset, you may need to create new features or

transform existing ones. This can involve one-hot encoding categorical

variables, handling date/time data, or scaling numerical features.

**Program:**

# Example: One-hot encoding for categorical variables

df = pd.get\_dummies(df, columns=[' Avg. Area Income ', ' Avg. Area

House Age '])

**5. Split the Data:**

Split your dataset into training and testing sets. This helps you evaluate

your model's performance later.

X = df.drop('price', axis=1) # Features

y = df['price'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2,

random\_state=42)

**6. Feature Scaling:**

Apply feature scaling to normalize your data, ensuring that all

features have similar scales. Standardization (scaling to mean=0 and

std=1) is a common choice.

**Program:**

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**Importance of loading and processing dataset:**

* Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for house price prediction models, as house price datasets are often complex and noisy.
* By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately.

**Challenges involved in loading and preprocessing a house price**

**dataset;**

There are a number of challenges involved in loading and preprocessing a house price dataset, including:

* **Handling missing values:**

House price datasets often contain missing values, which can be due to a variety of factors, such as human error or incomplete data collection. Common methods for handling missing values include dropping the rows with missing values, imputing the missing values with the mean or median of the feature, or using a more sophisticated method such as multiple imputation.

* **Encoding categorical variables:**

House price datasets often contain categorical features, such as the type of house, the neighborhood, and the school district. These features need to be encoded before they can be used by machine learning models. One common way to encode categorical variables is to use one-hot encoding.

* **Scaling the features:**

It is often helpful to scale the features before training a machine learning model. This can help to improve the performance of the model and make it more robust to outliers. There are a variety of ways to scale the features, such as min-max scaling and standard scaling.

* **Splitting the dataset into training and testing sets:**

Once the data has been pre-processed, we need to split the dataset into training and testing sets. The training set will be used to train the model, and the testing set will be used to evaluate the performance of the model on unseen data. It is important to split the dataset in a way that is representative of the real world distribution of the data.

**How to overcome the challenges of loading and preprocessing a house price dataset:**

There are a number of things that can be done to overcome the challenges of loading and preprocessing a Electricity Bill Prediction, including:

* **Use a data preprocessing library:**

There are a number of libraries available that can help with data preprocessing tasks, such as handling missing values, encoding categorical variables, and scaling the features.

* **Carefully consider the specific needs of your model:**

The best way to preprocess the data will depend on the specific machine learning algorithm that you are using. It is important to carefully consider the requirements of the algorithm and to preprocess the data in a way that is compatible with the algorithm.

* **Validate the preprocessed data:**

It is important to validate the preprocessed data to ensure that it is in a format that can be used by the machine learning algorithm and that it is of high quality. This can be done by inspecting the data visually or by using statistical methods.

**1.Loading the dataset:**

Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.

The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used. However, there are some general steps that are common to most machine learning frameworks:

**a.Identify the dataset:**

The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

**b.Load the dataset:**

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

**c.Preprocess the dataset:**

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.

Here, how to load a dataset using machine learning in Python

**Program:**

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

%matplotlib inline

import warnings

warnings.filterwarnings("ignore")

/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146:

UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for

this version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np\_minversion} and

<{np\_maxversion}"

**Loading Dataset:**

dataset = pd.read\_csv('E:/USA\_Housing.csv')

**Data Exploration:**

Dataset:

[5 rows x 18 columns]

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 DateTime 38014 non-null object

1 Holiday 38014 non-null object

2 HolidayFlag 38014 non-null int64

3 DayOfWeek 38014 non-null int64

4 WeekOfYear 38014 non-null int64

5 Day 38014 non-null int64

6 Month 38014 non-null int64

7 Year 38014 non-null int64

8 PeriodOfDay 38014 non-null int64

9 ForecastWindProduction 38014 non-null object

10 SystemLoadEA 38014 non-null object

11 SMPEA 38014 non-null object

12 ORKTemperature 38014 non-null object

13 ORKWindspeed 38014 non-null object

[8 rows x 7 columns]

First few rows of the dataset:

DateTime Holiday ... SystemLoadEP2 SMPEP2

0 01/11/2011 00:00 None ... 3159.60 54.32

1 01/11/2011 00:30 None ... 2973.01 54.23

2 01/11/2011 01:00 None ... 2834.00 54.23

3 01/11/2011 01:30 None ... 2725.99 53.47

4 01/11/2011 02:00 None ... 2655.64 39.87

14 CO2Intensity 38014 non-null object

15 ActualWindProduction 38014 non-null object

16 SystemLoadEP2 38014 non-null object

17 SMPEP2 38014 non-null object

dtypes: int64(7), object(11)

memory usage: 5.2+ MB

None

Summary statistics of numeric columns:

HolidayFlag DayOfWeek ... Year PeriodOfDay

count 38014.000000 38014.000000 ...

38014.000000 38014.000000

mean 0.040406 2.997317 ... 2012.383859 23.501105

std 0.196912 1.999959 ... 0.624956 13.853108

min 0.000000 0.000000 ... 2011.000000 0.000000

25% 0.000000 1.000000 ... 2012.000000 12.000000

50% 0.000000 3.000000 ... 2012.000000 24.000000

**2.Preprocessing the dataset:**

* Data preprocessing is the process of cleaning, transforming, and integrating data in order to make it ready for analysis.
* This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

**Visualisation and Pre-Processing of Data:**

In [1]:

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

Out[1]:

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

In [2]:

sns.boxplot(dataset, x='Price', palette='Blues')

Out[2]:

<Axes: xlabel='Price'>

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

sns.jointplot(dataset, x='Avg. Area Income', y='Price')

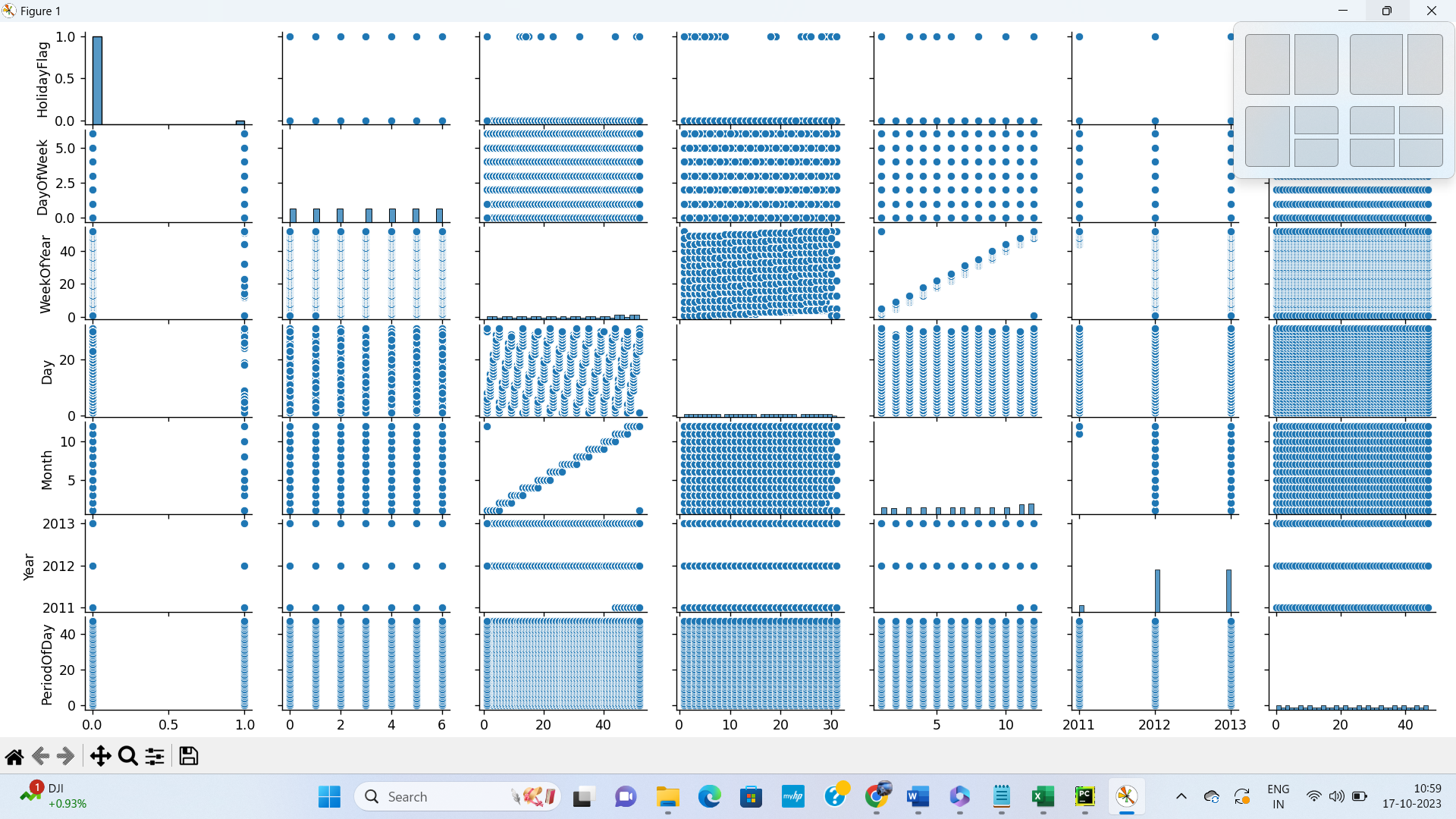
plt.figure(figsize=(12,8))sns.pairplot(dataset)

dataset.hist(figsize=(10,8))

dataset.corr(numeric\_only=True)

plt.figure(figsize=(10,5))sns.heatmap(dataset.corr(numeric\_only= True), annot=True)

**Output:**

****

**Some common data preprocessing tasks include:**

* **Data cleaning**: This involves identifying and correcting errors and inconsistencies in the data. For example, this may involve removing duplicate records, correcting typos, and filling in missing values.
* **Data transformation:** This involves converting the data into a format that is suitable for the analysis task. For example, this may involve converting categorical data to numerical data, or scaling the data to a suitable range.
* **Feature engineering:** This involves creating new features from

the existing data. For example, this may involve creating features

that represent interactions between variables, or features that

represent summary statistics of the data.

**Program:**

# Importing necessary libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

# Step 1: Load the dataset

data = pd.read\_csv('E:\USA\_Housing.csv')

# Step 2: Exploratory Data Analysis (EDA)

print("--- Exploratory Data Analysis ---")

print("1. Checking for Missing Values:")

missing\_values = data.isnull().sum()

print(missing\_values)

print("\n2. Descriptive Statistics:")

description = data.describe()

print(description)

# Step 3: Feature Engineering

print("\n--- Feature Engineering ---")

# Separate features and target variable

X = data.drop('price', axis=1)

y = data['price']

# Define which columns should be one-hot encoded (categorical)

categorical\_cols = [' Avg. Area House Age']

# Define preprocessing steps using ColumnTransformer and Pipeline

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), [' Avg. Area Number of Rooms ', ' Avg.

Area Number of Bedrooms ', ' Area Population ', ' Avg. Area Income ']),

('cat', OneHotEncoder(), categorical\_cols)

])

# Step 4: Data Splitting

print("\n--- Data Splitting ---")

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2,

random\_state=42)

print(f"X\_train shape: {X\_train.shape}")

print(f"X\_test shape: {X\_test.shape}")

print(f"y\_train shape: {y\_train.shape}")

print(f"y\_test shape: {y\_test.shape}")

# Step 5: Preprocessing and Feature Scaling using Pipeline

print("\n--- Feature Scaling ---")

model = Pipeline([

('preprocessor', preprocessor),

])

# Fit the preprocessing pipeline on the training data

X\_train = model.fit\_transform(X\_train)

# Transform the testing data using the fitted pipeline

X\_test = model.transform(X\_test)

print("--- Preprocessing Complete! ---")

**Output:**

Exploratory Data Analysis:

1. Checking for Missing Values:

Avg. Area Income 0

Avg. Area House Age 0

Avg. Area Number of Rooms 0

Avg. Area Number of Bedrooms 0

Area Population 0

Price 0

**Conclusion:**

* Data visualization and preprocessing are crucial steps in the data analysis and machine learning process. Data visualization helps us understand the dataset's characteristics, relationships, and potential issues. Data preprocessing ensures that the data is clean, properly formatted, and ready for modeling.
* Data visualization, performed using libraries like Matplotlib and Seaborn, allows us to create visual representations of data, understand relationships, and detect patterns.
* Data preprocessing involves handling missing data, feature selection, data scaling, encoding categorical variables, handling outliers, feature engineering, and more.
* These steps are essential to improve the quality of machine learning models and ensure they can make accurate predictions or classifications.
* The specific visualization and preprocessing steps should be adapted to the unique characteristics of your dataset and the goals of your analysis.