**Project 4: Measure Energy Consumption**

**Design**:

Automate energy consumption data collection, analysis, and visualization.

**Applicability**:

Optimize energy usage in various sectors with accurate insights.

**Technology**:

loT devices and data analysis tools for energy data.

**Coding**:

Combination of languages like C++, Python, and data analysis libraries

**Architecture**:

A digital energy detective tracking usage for informed decisions.

**Transformation**:

Shifts energy management from manual to data-driven Real-World Analogy: A smart meter that helps monitor and manage home energy

We are implement our AI program by this data set:

https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption

**Project Implementation**

**Steps:**

**1.Identify the Device**:

Decide on the device whose energy consumption you want to measure.

**2.Understand the Device's Power Rating**:

Most electronic devices have a power rating, often found on a label or in the device's manual. This rating is usually given in watts (W).

**3.Get an Energy Monitor**: Purchase a plug-in energy monitor, also known as an electricity usage monitor. These devices measure the real-time power usage and can often estimate the cost over time based on your electricity rates.

**4.Plug in the Device**:

* + Plug the energy monitor into a wall socket.
  + Then plug the device you want to measure into the energy monitor.

**5.Operate the Device**: Turn on the device and let it operate normally. The energy monitor will display the power the device is using in real-time.

**6.Collect Data**:

Many energy monitors will also record usage over time, so you can see how much energy the device uses in a day, week, or month.

**7.Calculate Total Consumption**:

* + If your monitor doesn't give a cumulative reading, you can calculate it.
  + Energy (in watt-hours) = Power (in watts) x Time (in hours).
  + To convert to kilowatt-hours (kWh), divide by 1000.

**8.Cost Estimation**:

If you want to calculate the cost of running that device:

* + Find out your electricity rate (usually in cents per kWh) from your electricity bill or provider.
  + Multiply the energy consumption in kWh by your rate.

**9.Analyze and Optimize**:

Based on your findings, decide if the device is consuming more power than necessary. Consider replacing old and inefficient appliances or adjusting usage patterns to save energy.

**10.Scale Up**:

If you want to measure the energy consumption of your entire home or building:

* + Consider using a whole-home energy monitor system.
  + Engage with your utility provider. They often provide tools, resources, and sometimes even rebates for energy audits and consumption measurements

**Technologies**

**1. Non-intrusive load monitoring (NILM):** This technology uses machine learning algorithms to identify and measure the energy usage of individual appliances in a household or building. It works by analyzing the fluctuations in voltage and current signals in the electrical system.

**2. Optical sensors:** Optical sensors can be used to measure energy consumption in lighting fixtures. They measure the light output of individual bulbs or fixtures to calculate energy usage.

**3. Ultrasonic flow meters:** These meters are used to measure the flow rate of liquids or gases (such as water or natural gas) through pipes. By measuring the flow rate, energy consumption can also be calculated**.**

**4. Current transformers:** Current transformers are used to measure the flow of electric current in a circuit. They work by inducing a secondary current that is proportional to the primary current, which can then be measured.

**5. Power quality analyzers:** These devices are used to measure the quality of electrical power, including voltage, current, and frequency. They can also measure power factor, harmonic distortion, and other parameters that affect energy consumption.

**6. Internet of Things (IoT) sensors:** IoT sensors can be used to measure energy consumption in a variety of devices and applications. They can provide real-time data and insights into energy usage patterns for optimization and cost-saving purposes.

**Advanced Innovative Techniques**

**Time series analysis:**

* Time series analysis is a technique that involves analyzing time-dependent datasets to identify trends, seasonality, or other patterns.

* In the context of energy consumption, time series analysis can be used to analyze historical energy usage data and then forecast future energy consumption based on these patterns.

**Machine learning models:**

* Machine learning models, on the other hand, use statistical algorithms to learn from data and make predictions or decisions without being explicitly programmed.

* When applied to energy consumption, machine learning models use historical consumption data and external factors such as temperature, weather conditions, and occupancy patterns to understand energy usage patterns better.
* By analyzing these patterns, machine learning models can make predictions about future energy consumption and even optimize energy usage by automatically adjusting settings on devices and appliances.

Some examples of machine learning models used in energy consumption forecasting include:

**1. Neural Networks:** Neural networks are used in energy consumption forecasting to recognize complex relationships between different energy systems and predict future energy usage.

**2. Decision Trees**: Decision trees are used to determine the most important variables when predicting energy consumption trends. They can be used to make predictions based on time of day, weather, occupancy patterns, and other factors.

**3. Random Forests:** Random forests use multiple decision trees to make more accurate predictions about energy consumption. They are particularly useful for identifying the most important variables and factors.

**4. Support Vector Machines:** Support Vector Machines are used to separate data into categories and identify patterns within those categories. They can be used to analyze energy consumption data and predict future usage.

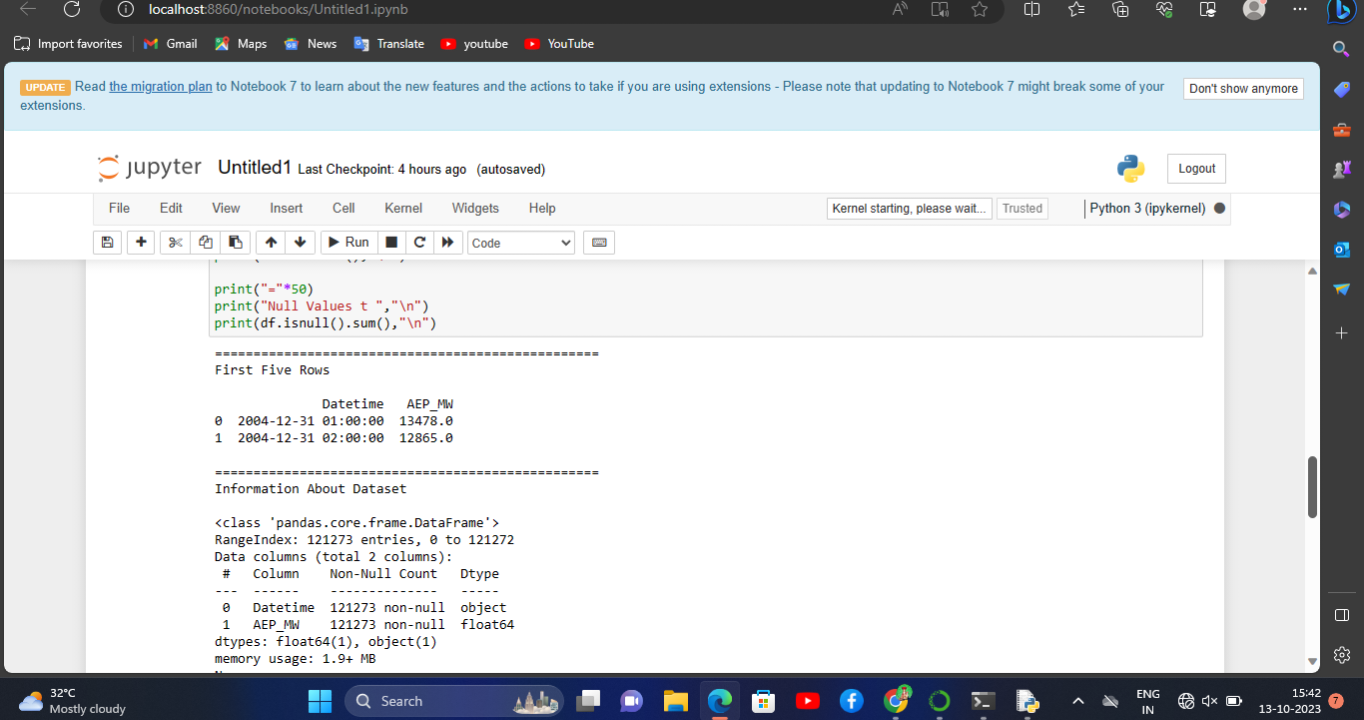
**5. Recurrent Neural Networks:** Recurrent Neural Networks can analyze large datasets and identify patterns over time. They are particularly useful for predicting energy consumption trends over several days or weeks.

Overall, time series analysis and machine learning models can help organizations and individuals better understand their energy usage patterns and optimize their energy consumption. By using these techniques, it is possible to predict future energy consumption patterns and make data-driven decisions to reduce energy usage and save on electricity bills.

**Development Part 1**

In this part you will begin building your project by loading and preprocessing the dataset.

We provide a code to implement the dataset and preprocessing the data. Then, we cleaned the data in the dataset.



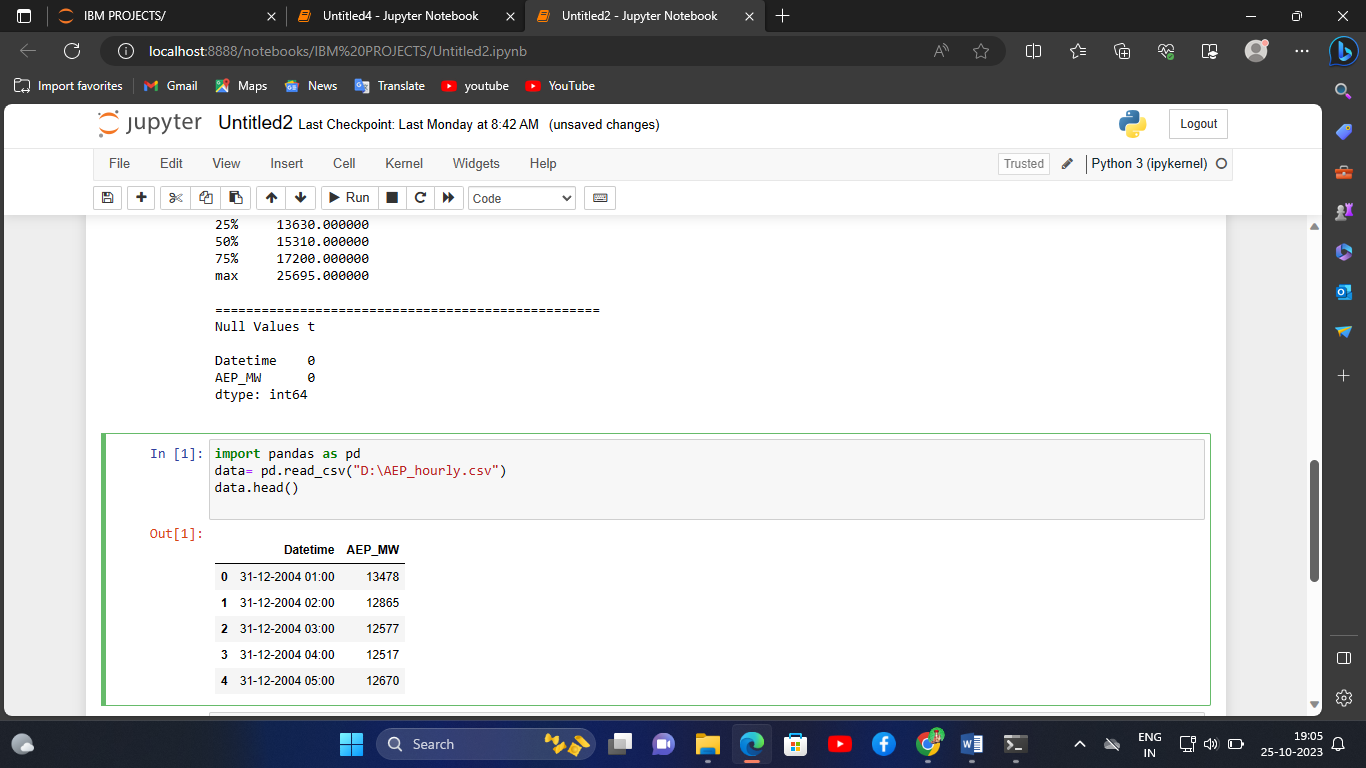
We process the code and import the dataset. Then, we get the output from the dataset.

We got the data easily.

**Development Part 2**

* In this part you will continue building your project.
* Continue the development by:
* Analyzing the energy consumption data
* Creating visualizations.
* We training our spam detection with help of the below dataset link

[**https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption**](https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption)



**PROGRAM CODE:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pprint

df = pd.read\_csv("D:\AEP\_hourly.csv")

print("="\*50)

print("First Five Rows ","\n")

print(df.head(2),"\n")

print("="\*50)

print("Information About Dataset","\n")

print(df.info(),"\n")

print("="\*50)

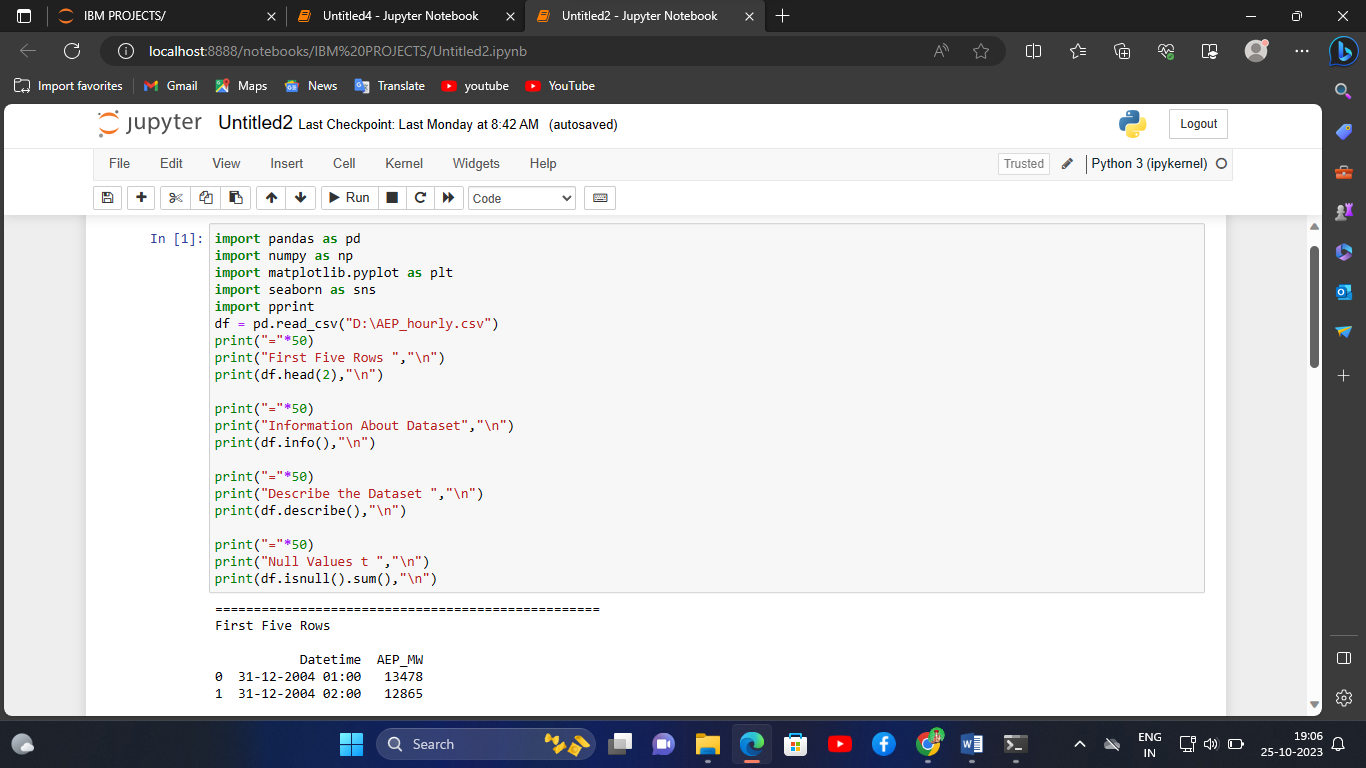
print("Describe the Dataset ","\n")

print(df.describe(),"\n")

print("="\*50)

print("Null Values t ","\n")

print(df.isnull().sum(),"\n")

****

**OUTPUT:**

==================================================

First Five Rows

Datetime AEP\_MW

0 31-12-2004 01:00 13478

1 31-12-2004 02:00 12865

==================================================

Information About Dataset

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 121273 entries, 0 to 121272

Data columns (total 2 columns):

# Column Non-Null Count Dtype

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

0 Datetime 121273 non-null object

1 AEP\_MW 121273 non-null int64

dtypes: int64(1), object(1)

memory usage: 1.9+ MB

None

==================================================

Describe the Dataset

AEP\_MW

count 121273.000000

mean 15499.513717

std 2591.399065

min 9581.000000

25% 13630.000000

50% 15310.000000

75% 17200.000000

max 25695.000000

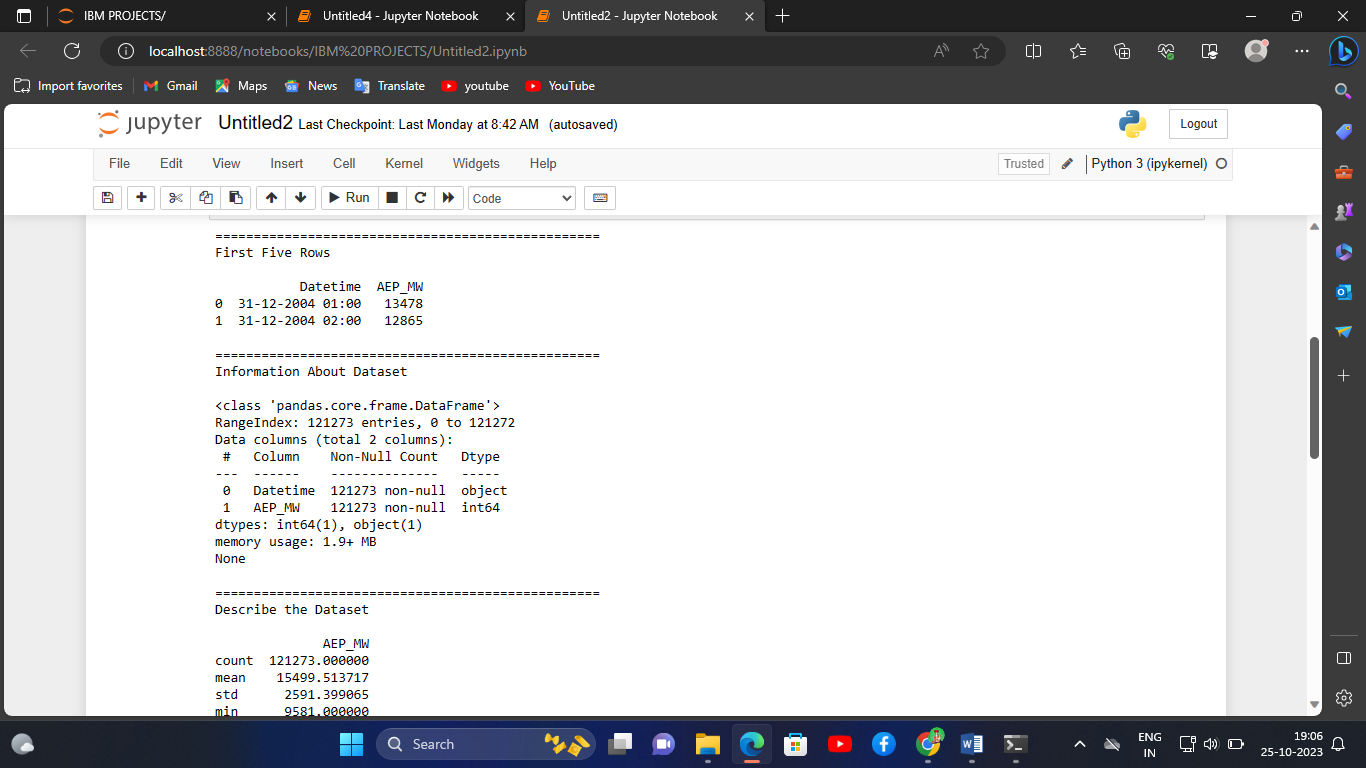
==================================================

Null Values t

Datetime 0

AEP\_MW 0

dtype: int64

****

**VISUALIZATION THE DATA:**

**PROGRAM CODE:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore", category=UserWarning)

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

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error, r2\_score

RED = "\033[91m"

GREEN = "\033[92m"

YELLOW = "\033[93m"

BLUE = "\033[94m"

RESET = "\033[0m"

df = pd.read\_csv("D:/AEP\_hourly.csv")

df["Datetime"] = pd.to\_datetime(df["Datetime"])

# DATA CLEANING

print(BLUE + "\nDATA CLEANING" + RESET)

# --- Check for missing values

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

print(GREEN + "Missing Values : " + RESET)

print(missing\_values)

# --- Handle missing values

df.dropna(inplace=True)

# --- Check for duplicate values

duplicate\_values = df.duplicated().sum()

print(GREEN + "Duplicate Values : " + RESET)

print(duplicate\_values)

# --- Drop duplicate values

df.drop\_duplicates(inplace=True)

# DATA ANALYSIS

print(BLUE + "\nDATA ANALYSIS" + RESET)

# --- Summary Statistics

summary\_stats = df.describe()

print(GREEN + "Summary Statistics : " + RESET)

print(summary\_stats)

# SUPPORT VECTOR MODELLLING

print(BLUE + "\nMODELLING" + RESET)

# Reduce the dataset size for faster training

df = df.sample(frac=0.2, random\_state=42)

# Split the data into features (Datetime) and target (AEP\_MW)

X = df[["Datetime"]]

y = df["AEP\_MW"]

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42)

# Preprocess the features (Datetime) to extract the day of the year

X\_train["DayOfYear"] = X\_train["Datetime"].dt.dayofyear

X\_test["DayOfYear"] = X\_test["Datetime"].dt.dayofyear

# Convert X\_train and X\_test to NumPy arrays

X\_train = X\_train["DayOfYear"].values.reshape(-1, 1)

X\_test = X\_test["DayOfYear"].values.reshape(-1, 1)

# Standardize the data

scaler = StandardScaler()

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

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

# Create an SVR (Support Vector Regression) model with a linear kernel

svr = SVR(kernel="linear", C=1.0)

# Train the SVR model

svr.fit(X\_train\_scaled, y\_train)

# Predict on the test set

y\_pred = svr.predict(X\_test\_scaled)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R-squared: {r2}")

# Plot the actual vs. predicted values

plt.figure(figsize=(10, 6))

plt.scatter(X\_test, y\_test, color="b", label="Actual")

plt.scatter(X\_test, y\_pred, color="r", label="Predicted")

plt.xlabel("Day of the Year")

plt.ylabel("Energy Consumption (MW)")

plt.title("SVR Model: Actual vs. Predicted")

plt.legend()

plt.grid()

plt.show()

# DATA VISUALIZATION

print(BLUE + "\nDATA VISUALIZATION" + RESET)

# --- Line plot

print(GREEN + "LinePlot : " + RESET)

plt.figure(figsize=(10, 6))

sns.lineplot(data=df, x="Datetime", y="AEP\_MW")

plt.xlabel("Datetime")

plt.ylabel("Energy Consumption (MW)")

plt.title("Energy Consumption Over Year")

plt.grid()

plt.show()

# --- Histogram

print(GREEN + "Histogram : " + RESET)

plt.figure(figsize=(10, 6))

plt.hist(

df["AEP\_MW"],

bins=100,

histtype="barstacked",

edgecolor="white",

)

plt.xlabel("AEPMW")

plt.ylabel("Frequency")

plt.title("Histogram of MEGAWATT USAGE")

plt.show()

# SAVING THE FILE

df.to\_csv("D:/AEP\_hourly.csv", index=False)

print(BLUE + "\nDATA ANALYSIS" + RESET)

print(GREEN + "Data Cleaned and Saved !" + RESET)

**DATA CLEANING**

Missing Values :

Datetime 0

AEP\_MW 0

dtype: int64

Duplicate Values :

0

**DATA ANALYSIS**

Summary Statistics :

Datetime AEP\_MW

count 121273 121273.000000

mean 2011-09-02 03:17:01.553025024 15499.513717

min 2004-10-01 01:00:00 9581.000000

25% 2008-03-17 15:00:00 13630.000000

50% 2011-09-02 04:00:00 15310.000000

75% 2015-02-16 17:00:00 17200.000000

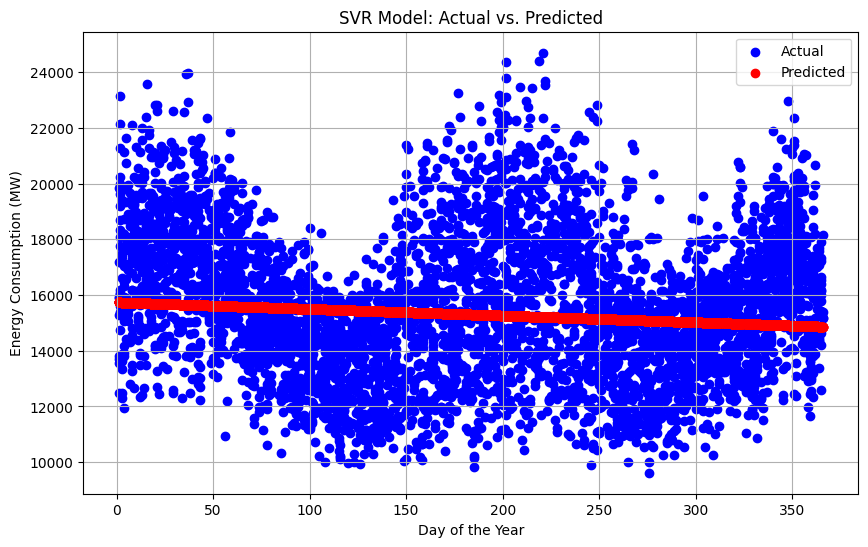
max 2018-08-03 00:00:00 25695.000000

std NaN 2591.399065

**MODELLING**

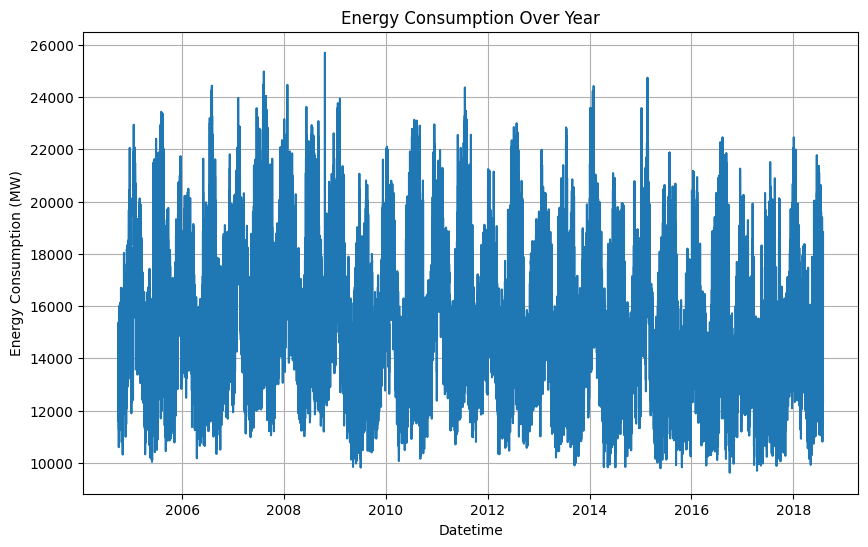
Mean Squared Error: 6758395.805638685

R-squared: 0.00270160624748228



**DATA VISUALIZATION**

**LinePlot :**

****