

Reg no :721921243049

Name:Hariprasath.R

Project: *Measure Energy Consumption*



Introduction:

- Certainly, using time series analysis and machine learning models to predict future energy consumption patterns is a promising approach.
- Here's how you can explore these techniques for energy consumption prediction.

Objective

- ❑ **Explore innovative techniques such as time series analysis and machine learning models to predict future energy consumption patterns**
- ❑ **In PHASE 1 we discussed about the problem definition and their application used in artificial intelligence.**

Phase 2:

- ❑ **Consider Certainly, using time series analysis and machine learning models to predict future energy consumption patterns is a promising approach.**
- ❑ **Here's how you can explore these techniques for energy consumption prediction..Here is the following contents.**
- ❑ **Here are some key components and methods commonly used in such solutions:**

1.Data Collection:

- ❑ **Gather historical energy consumption data.**
- ❑ **This data should include information about the time and date of measurements, energy usage, and potentially other relevant factors like temperature, holidays, or special events.**
- ❑ **Dataset link is below here:**

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



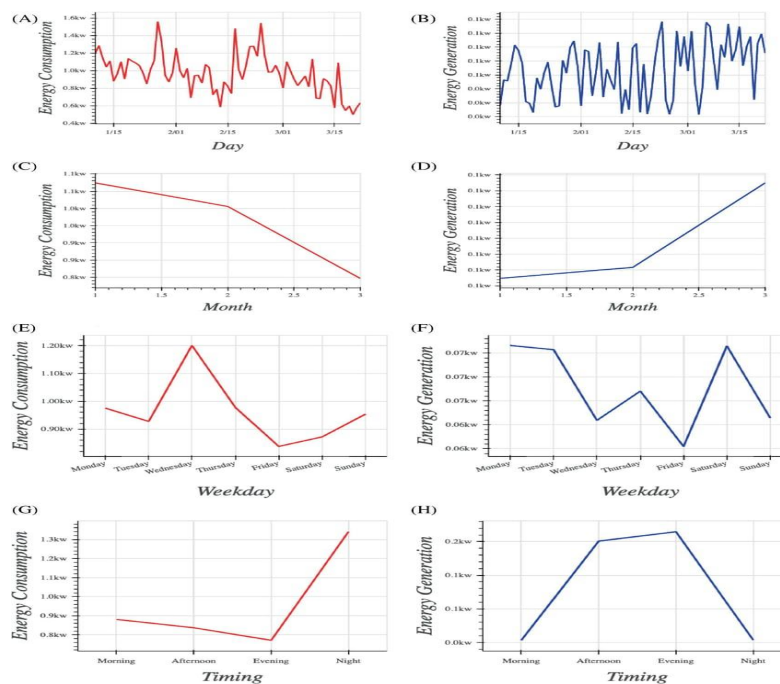
2. Data Preprocessing:

- ❑ **Data preprocessing is an important step before applying machine learning methods for energy or load prediction.**
- ❑ **It improves accuracy and reliability.**
- ❑ **There are four types of data processing**
 - 1. Data cleaning**
 - 2. Data integration**
 - 3. Data transformation**
 - 4. Data reduction**



3.Exploratory Data Analysis (EDA):

- **Perform EDA to gain insights into the data.**
- **Learn everything you need to know about exploratory data analysis, a method used to analyze and summarize data sets.**
- **Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.**
- **Visualize trends, seasonality, and correlations between energy consumption and other variables.**



4. Time Series Analysis:

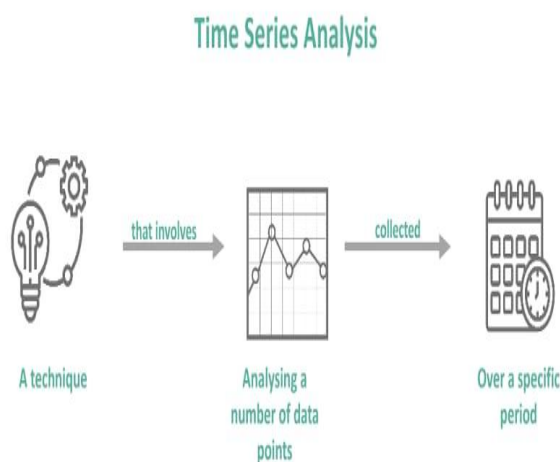
- **It comprises of ordered sequence of data at equally spaced interval.**
- **In particular, a time series allows one to see what factors influence certain variables from period to period.**

1. Decomposition:

Decompose the time series data into its trend, seasonal, and residual components to understand underlying patterns.

2. Smoothing:

Apply smoothing techniques like moving averages or exponential smoothing to reduce noise in the data.



5.Feature Engineering:

- **Create additional features that could impact energy consumption, such as holidays, weather data, day of the week, and time of day.**
- **This includes the use of electricity, gas, diesel, oil, and biomass. The concept of energy consumption is directly related to energy efficiency since higher consumption results in lower energy efficiency.**
- **TEE includes three core components**

1.Resting metabolic rate, or resting energy expenditure (REE)

2.The thermic effect of food (TEF)

3. Diet-induced thermogenesis DIT)

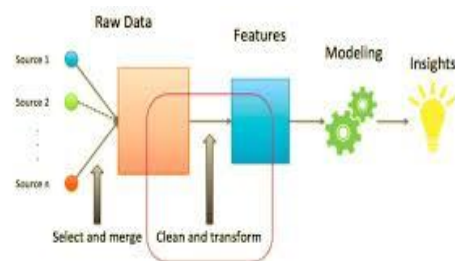


Figure 1-2: The place of feature engineering in the machine learning workflow.

6.Machine Learning Models:

A) Regression Models:

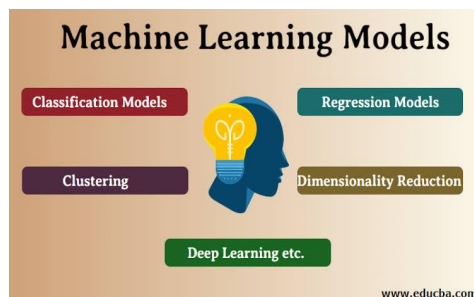
- **Utilize linear regression, decision trees, or random forests to build models that predict energy consumption based on relevant features.**

B) Time Series Forecasting:

- ❑ **Implement specialized time series forecasting models like ARIMA, Prophet, or LSTM (Long Short-Term Memory) neural networks.**

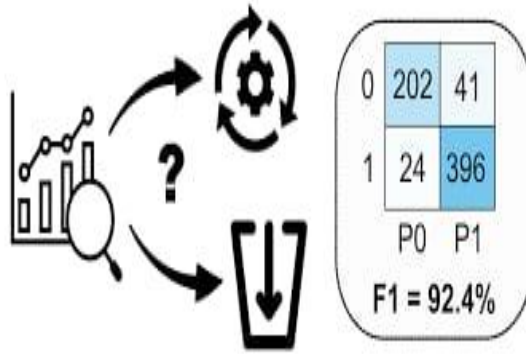
C) Ensemble Methods:

- ❑ **Combine multiple models to improve prediction accuracy.**



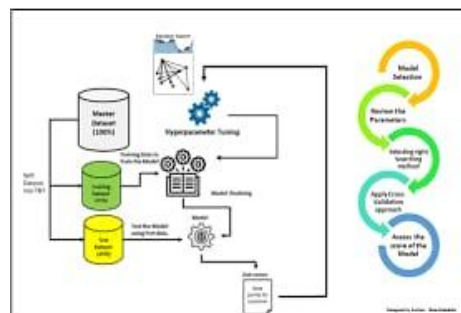
7. Model Evaluation:

- ❑ **Energy consumption modeling seeks to determine energy requirements as a function of input parameters.**
- ❑ **Models may be used for determining the requirements of energy supply and the consumer consumption variations while an upgrade or addition of technology exist.**
- ❑ **Use appropriate metrics such as**
 - Mean Absolute Error (MAE)**
 - Mean Squared Error (MSE)**
 - Root Mean Squared Error (RMSE)**



8. Hyperparameter Tuning:

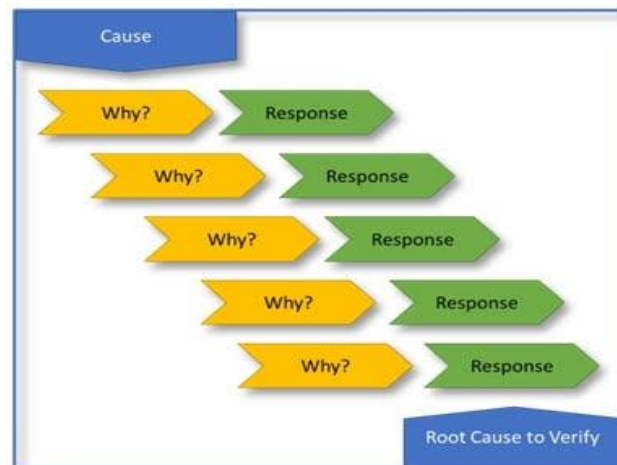
- ❑ **Optimize model hyperparameters to enhance predictive accuracy.**
- ❑ **Hyperparameter tuning allows data scientists to tweak model performance for optimal results.**



9. Cross-Validation:

- ❑ **Employ cross-validation techniques to assess the model's generalization performance.**
- ❑ **Cross-validation is a statistical method used to estimate the performance (or accuracy) of machine learning models.**

- It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited.



10. Monitoring and Updating:

- Continuously monitor the model's performance and update it as needed with new data.

11. Deployment:

- Once you have a reliable model, deploy it in a real-world environment to make real-time predictions.

12. Interpretability:

- **Understand the factors driving energy consumption by examining feature importance and model explanations.**

13. Integration:

- **Integrate the energy consumption prediction model into your energy management system to optimize resource allocation and reduce costs.**

Source code:

Int[1]:

```
Import numpy as np  
Import pandas as pd  
Import matplotlib.pyplot as plt  
Import matplotlib.dates as mdates  
%matplotlib inline  
Import seaborn as sns  
Import warnings  
Warnings.filterwarnings("ignore")  
From pandas.plotting import lag_plot  
From pylab import rcParams  
From statsmodels.tsa.seasonal import  
seasonal_decompose  
From pandas import DataFrame  
From pandas import concat
```

Int[2]:

```
Df=pd.read_csv("../input/hourly-energy-consumption/AEP_hourly.csv",index_col='Datetime',parse_dates=True)
```

```
Df.head()
```

Out[2]:

	AEP_MW
Datetime	
2004-12-31 01:00:00	13478.0
2004-12-31 02:00:00	12865.0
2004-12-31 03:00:00	12577.0
2004-12-31 04:00:00	12517.0
2004-12-31 05:00:00	12670.0

Int[3]:

```
df.sort_values(by='Datetime', inplace=True)  
print(df)
```

Int[4]:

```
df.shape
```

Out[4]:

(121273, 1)

Int[5]:

df.info()

Out[5]:

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

**DatetimeIndex: 121273 entries, 2004-10-01 01:00:00
to 2018-08-03 00:00:00**

Data columns (total 1 columns):

Column Non-Null Count Dtype

--- -----

0 AEP_MW 121273 non-null float64

dtypes: float64(1)

memory usage: 1.9 MB

Int[6]:

df.describe()

Out[6]:

	AEP_MW
--	---------------

count	121273.000000
mean	15499.513717
std	2591.399065
min	9581.000000
25%	13630.000000
50%	15310.000000
75%	17200.000000
100%	25695.000000

Int[7]:

```
df.index = pd.to_datetime(df.index)
```

Int[8]:

Extract all Data Like Year MOnth Day Time etc

```
df["Month"] = df.index.month
```

```
df["Year"] = df.index.year
```

```
df["Date"] = df.index.date
```

```
df["Hour"] = df.index.hour
```

```
df["Week"] = df.index.week
```

```
df["Day"] = df.index.day_name()
```

df.head()

Out[8]:

	AEP _M _W	Mon th	Year	Date	Hour	Wee k	Day
Date time							
2004 -10-01 01:00:00	1237 9.0	10	2004	2004 -10-01 11	1	4	Frid ay
2004 -10-01 02:00:00	1193 5.0	10	2004	2004 -10-01 11	2	4	Frid ay
2004 -10-01 03:00:00	1169 2.0	10	2004	2004 -10-01 11	3	4	Frid ay

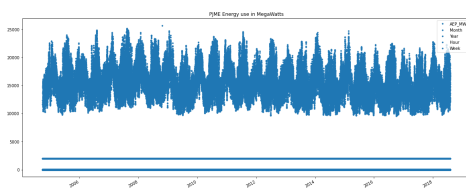
2004 -10-01 04:00:00	1159 7.0	10	2004	2004 -10-01 11	4	4	Frid ay
2004 -10-01 05:00:00	05:00:00	10	2004	2004 -10-01 11	5	4	Frid ay

Int[9]:

```
df.plot(title="PJME Energy use in MegaWatts",  
        figsize=(20, 8),  
        style=".",  
        color=sns.color_palette()[0])
```

plt.show()

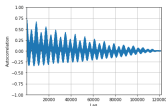
Out[9]:



Int[10]:


```
from pandas.plotting import autocorrelation_plot  
autocorrelation_plot(df['AEP_MW'])  
plt.show()
```

Out[10]:



Int[11]:

```
#Train Arima Model
```

```
train_arima = train_data['AEP_MW']
```

```
test_arima = test_data['AEP_MW']
```

```
history = [x for x in train_arima]
```

```
y = test_arima
```

```
# make first prediction
```

```
predictions = list()
```

```
model = sm.tsa.arima.ARIMA(history,  
order=(5,1,0))
```

```
model_fit = model.fit()
```

```
yhat = model_fit.forecast()[0]
```

```
predictions.append(yhat)  
history.append(y[0])  
# rolling forecasts  
for i in range(1, len(y)):  
    # predict  
    model = sm.tsa.arima.ARIMA(history,  
order=(5,1,0))  
    model_fit = model.fit()  
    yhat = model_fit.forecast()[0]  
    # invert transformed prediction  
    predictions.append(yhat)  
    # observation  
    obs = y[i]  
    history.append(obs)  
  
plt.figure(figsize=(14,8))  
plt.plot(df.index, df['AEP_MW'], color='green',  
label = 'Train Energy AEP_MW')  
plt.plot(test_data.index, y, color = 'red', label = 'Real  
Energy AEP_MW')  
plt.plot(test_data.index, predictions, color = 'blue',  
label = 'Predicted Energy AEP_MW')
```

```
plt.legend()
```

```
plt.grid(True)
```

```
plt.show()
```

```
plt.figure(figsize=(14,8))
```

```
plt.plot(df.index[-600:], df['AEP_MW'].tail(600),  
color='green', label = 'Train Energy AEP_MW')
```

```
plt.plot(test_data.index, y, color = 'red', label = 'Real  
Energy AEP_MW')
```

```
plt.plot(test_data.index, predictions, color = 'blue',  
label = 'Predicted Energy AEP_MW')
```

```
plt.legend()
```

```
plt.grid(True)
```

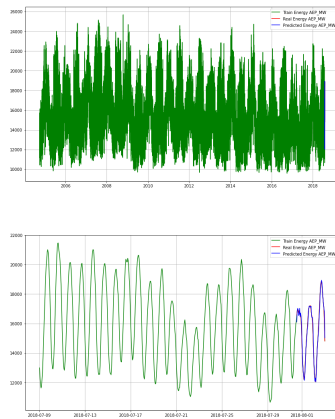
```
plt.show()
```

```
print('MSE: '+str(mean_squared_error(y,  
predictions)))
```

```
print('MAE: '+str(mean_absolute_error(y,  
predictions)))
```

```
print('RMSE: '+str(sqrt(mean_squared_error(y,  
predictions))))
```

```
Out[11]:
```



Conclusion:

- Remember that the accuracy of your predictions will depend on the quality and quantity of data, as well as the choice of the most appropriate modeling techniques.
- Regularly updating the model with new data is crucial to ensure it remains accurate as consumption patterns evolve.