MEASURE ENERGY CONSUMPTION

USING MACHINE LEARNING

- Smart Grids (SG) have emerged as a solution to the increasing demand on energy worldwide. The grid refers to the traditional electrical grid that is a collection of transmission lines, substations, and other components that make sure energy is delivered from the power plant to the home or business [1]. The smartness in the SG resides in the two-way communication between the utility and the customers, in addition to the sensing along the lines..

- The SG has many benefits among which we state: more efficient energy transmission, improving security, reducing peak demand which helps with the decrease of electricity rates, etc. SG are also known by the use of renewable energy sources.

- The prediction and scheduling are two of the main pillars of efficient Energy Management Systems (EMS). EMSs are very crucial for the well-functioning of the SG. They are responsible for managing the power flux within the SG elements in order to minimize the costs and optimize the quality [2].

- The prediction of the energy consumed by different appliances is one of the building blocks of the concept of SGs. The energy consumption can be seen as a nonlinear time series with a number of complex factors [3]. With many renewable energy sources used in the SGs, the energy prediction methods are getting more and more accurate, and hence, the prediction becomes a crucial part in the efficient planning of the entire SG.

There are different approaches that are used for the prediction of the energy consumption. The most popular ones use machine learning (ML).

- Machine learning (ML) is one of the growing technical fields that merge between computer science and statistics. It tackles the issue of building computers that learn through experiences and hence provide more improved algorithms.

ANNs have seen light in the early 1940s but have not been widely used until lately. They became very popular thanks to the outstanding results they offer. They are very powerful with large datasets which gives the neural network enough data to train the model. In brief, ANNs are inspired by the way the brain processes information.

- They build an informational processing model that mimics the work of the neurons in the brain [5]. Their ability to learn quickly is what makes ANNs very powerful. This learning is done through an information flow that goes in two directions. Patterns from the training dataset are given to the ANN through the input neurons, then goes through the hidden layers and arrives to the output neurons.

- ANN and GA models are usually implemented in commodity computers or lately in Raspberry Pis. The NI CompactRIO is considered as a good alternative for deploying the ANN algorithms.

- NI CompactRIO is a high-performance embedded controller with Input/Output modules. It has two targets: a real-time controller chassis, and an FPGA module. It includes a microprocessor to implement control algorithms and offer a support of a large pool of frequencies. The FPGA module is mainly used to accommodate for the high speed of certain modules and even certain programs. It deals with the data streaming from the I/O modules attached to the CompactRIO. The FPGA module is brought by Xilinx Virtex.

- The CompactRIO is programmable using a specific graphical programming language named LabVIEW. This latter allows a better visualization of the data and an intuitive and easy way to implement control approaches.

- In this paper, we are training an ANN model to predict the energy consumed by different appliances in a building. The model is developed in Python programming language but interfaced with LabVIEW for a potential integration in the NI CompactRIO.

- The rest of the paper is organized as follows: Sect. 2 presents the scope of the research project under which this work is done. Section 3 contains the background of our work. In Sect. 4, we present the implementation

steps and discussing the results obtained. Then, we conclude and present our future work in Sect. 5.

Example **:**

```
import datetime

import warnings

import pandas as pd

import numpy as np

import lightgbm as lgb

import xgboost as xgb

import seaborn as sns

import matplotlib.pyplot as plt

import plotly.express as px


from typing import Optional, List, Dict

from fbprophet import Prophet

from xgboost import plot_importance, plot_tree

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, mean_absolute_error


warnings.filterwarnings('ignore')

plt.style.use('fivethirtyeight')


# loading and preprocessing the data
```

```
ts = pd.read_csv('/kaggle/input/hourly-energy-consumption/PJME_hourly.csv',
                 parse_dates=[0])
ts = ts.rename(columns={"Datetime":"datetime",
                        "PJME_MW":"observation"}).sort_values(by="datetime")
MIN_DATE = datetime.datetime(2014, 1, 1)
MAX_DATE = datetime.datetime(2018, 1, 1)
ts = ts.loc[(ts["datetime"] > MIN_DATE) &
            (ts["datetime"] < MAX_DATE)
           ].reset_index(drop=True)
ts.head()
```

OUTPUT :

	datetime	observation
0	2014-01-01 01:00:00	31440.0
1	2014-01-01 02:00:00	30626.0
2	2014-01-01 03:00:00	29949.0
3	2014-01-01 04:00:00	29716.0
4	2014-01-01 05:00:00	29905.0

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

LOADING & PREPROCESSING DATA SET :

When loading and preprocessing a dataset for measuring energy consumption using machine learning, you'll need to consider several subtopics and steps to prepare the data for model development. Here are subtopics related to this process:

1. Data Collection and Retrieval

- Data sources: Identify where you collect energy consumption data, such as smart meters, sensors, or historical records.

- Data retrieval: Set up processes to fetch data from these sources, potentially in real-time or batch mode.

2. Data Cleaning

- Missing data: Handle missing values through imputation or removal.

- Outlier detection: Identify and address outliers in the data that might negatively impact machine learning models.

3. Data Transformation

- Time series handling: Convert timestamp data into a time series format for temporal analysis.

- Data aggregation: Aggregate data to different time granularities (e.g., hourly, daily) for machine learning.

4. Feature Engineering

- Create features: Develop relevant features like day of the week, time of day, and seasonality to improve model performance.

- Feature selection: Choose the most relevant features and potentially reduce dimensionality.

5. Data Normalization/Scaling:

- Standardize or normalize data to ensure consistent scales for different features, which is important for many machine learning algorithms.

6. Data Splitting

- Divide the dataset into training, validation, and test sets for model development and evaluation.

- Consider time-based splitting for time series data to maintain temporal integrity.

7. **Data Preprocessing for Machine Learning**:

- Encode categorical variables: Convert categorical data (e.g., location, appliance type) into numerical format for machine learning.

- Time series preprocessing: Apply techniques like differencing, scaling, or rolling statistics to make time series data suitable for modeling.

8. **Data Quality Assessment**:

- Evaluate the quality of data sources and preprocessing steps to ensure the accuracy and reliability of the dataset for machine learning.

9. Data Storage and Retrieval

- Decide on a data storage and retrieval strategy to efficiently access and manage the preprocessed data during model training and deployment.

10. **Data Security and Privacy**:

- Implement measures to protect sensitive energy consumption data and ensure compliance with data privacy regulations, especially when working with real-world data.

11. Data Integration (Optional)

- Combine energy consumption data with other relevant datasets, such as weather data or occupancy information, to enhance the predictive power of the machine learning models.

12. Data Documentation

- Create documentation that explains the dataset, preprocessing steps, and any assumptions made during the process. This is crucial for model transparency and reproducibility.

13. Feature Scaling and Normalization:

- Scale or normalize features as needed to make them suitable for machine learning algorithms, particularly when working with algorithms like neural networks or support vector machines.

14. **Handling Class Imbalance (If Applicable**)

- If you're dealing with classification tasks, address class imbalance issues through techniques like oversampling or undersampling.

Each of these subtopics is crucial when using machine learning for measuring energy consumption, and they will help you prepare the dataset for accurate modeling and analysis. The specific steps and

emphasis on each subtopic may vary depending on the machine learning approach and the characteristics of your dataset.

Load the dataset

The Dayton Power and Light Company and DPL Energy Resources, DP&L sells to, and generates electricity for, a customer base of over 500,000 people within a 6,000-square-mile (16,000 km2) area of West Central Ohio, including the area around Dayton, Ohio. The dataset provides 121275 entries as estimated hourly energy consumption in Megawatts (MW) from 31st December 2004, 01:00:00 to 2nd January 2018, 00:00:00

#loading raw data

df = pd.read_csv("../input/hourly-energy-consumption/DAYTON_hourly.csv", index_col=0)

df.head().style.set_properties(**{'background-color': 'rgb(211, 176, 176)'})

	DAYTON_MW	Datetime
2004-12-31 01:00:00	1596.000000	
2004-12-31 02:00:00	1517.000000	
2004-12-31 03:00:00	1486.000000	
2004-12-31 04:00:00	1469.000000	
2004-12-31 05:00:00	1472.000000	

df.sort_index(inplace = True)

df.head().style.set_properties(**{'background-color': 'rgb(211, 276, 176)'})

	DAYTON_MW	Datetime
2004-10-01 01:00:00	1621.000000	
2004-10-01 02:00:00	1536.000000	
2004-10-01 03:00:00	1500.000000	
2004-10-01 04:00:00	1434.000000	
2004-10-01 05:00:00	1489.000000	

```
display_plot(df.iloc[-2*8766:,:],
'Dayton Power & Light Company (DP&L)
hourly energy consumption in
MegaWatts (MW) for the last year')
```

Data Visualization¶

```
import holoviews as hv
from holoviews import opts
hv.extension('bokeh')
from bokeh.models.annotations import Label
hv.Distribution(df['DAYTON_MW']).opts(title="Dayton Power & Light Company (DP&L) Hourly Energy Consumption in MW", color="red",
                                      xlabel="Hourly Energy Consumption", ylabel="Density")\
.opts(opts.Distribution(width=700, height=300,tools=['hover'],show_grid=True))
```

```
df.plot(style=".", figsize=(15,5), title="PJME Hourly Power Consumption in MW")
```

Out[5]:

```
<AxesSubplot:title={'center':'PJME Hourly Power Consumption in MW'}, xlabel='Datetime'>
```