MEASUREMENT OF ENERGY CONSUMPTION:

Statement:

The measurement of energy consumption is a critical aspect of managing and optimizing energy usage in various sectors such as manufacturing, residential homes, commercial buildings, and transportation. However, traditional methods of manually collecting and analyzing energy consumption data are not only time-consuming but also prone to errors, making it challenging to make informed decisions for energy efficiency and sustainability. To address these challenges, there is a need for an automated and efficient system that can collect, analyze, and visualize energy consumption data in real-time, enabling better decision-making and resource optimization

Technique:

To automate the measurement of energy consumption and facilitate data-driven decision making, we propose the development of an integrated Energy Management System (EMS). This system will consist of several key components and functionalities:

1.Data Collection:

- Sensors and meters will be strategically installed at various energy consumption points within the target area (e.g., manufacturing facility, building, vehicle).

- These sensors will continuously collect data on energy usage, including electricity, gas, and other energy sources.

- Data will be collected in real-time or at regular intervals and transmitted to a central database.

2.Data Processing and Analysis:

- The collected energy consumption data will be processed in real-time using advanced data analytics techniques.

- Algorithms will be employed to identify patterns, anomalies, and trends in energy consumption.

- Energy efficiency metrics and benchmarks will be calculated to assess the performance of the system.

3.Visualization and Reporting:

- A user-friendly dashboard will be developed to provide real-time visualizations of energy consumption data.

Users, such as facility managers, homeowners, or transportation fleet operators, can access the dashboard to monitor energy usage.

- Customizable reports will be generated, highlighting key performance indicators and actionable insights.

4.Alerts and Notifications:

- The system will be equipped with alert mechanisms to notify users of abnormal energy consumption patterns or potential issues.
- Notifications can be sent via email, SMS, or other preferred communication channels.

5.Integration and Control:

- The EMS will integrate with existing control systems to enable automated energy optimization strategies.

- For example, it can adjust HVAC settings, lighting, or machinery operation based on real time energy data and predefined rules.

6.Data Security and Compliance:

- Robust data security measures will be implemented to protect sensitive energy consumption data.

- Compliance with relevant data privacy regulations will be ensured.

7.Scalability and Customization:

The EMS will be designed to be scalable and adaptable to various sectors and energy sources.

Users can customize the system to meet their specific energy management needs.

INOVATION:

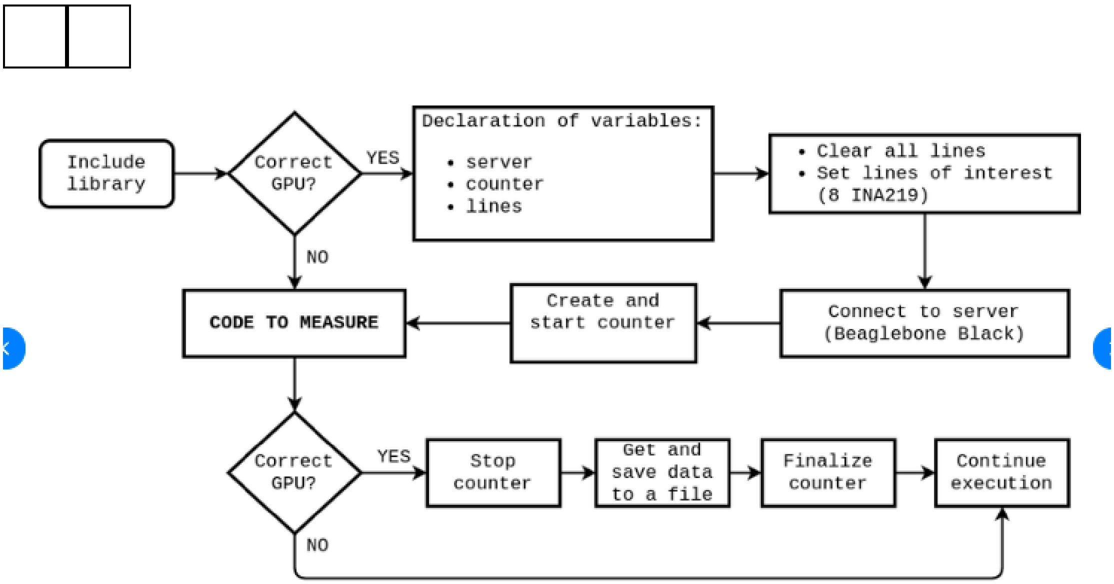
Resource Optimization: By tracking energy usage, you can identify areas where energy is being wasted or used inefficiently. This information can help you optimize resource allocation and reduce operational costs.

Environmental Impact: Understanding the environmental impact of your innovation is essential in today's sustainability-conscious world. Measuring energy consumption allows you to calculate your carbon footprint and make efforts to reduce it.

Compliance: Many regions have regulations and standards related to energy consumption and environmental impact. Measuring energy usage helps ensure compliance with these regulations.

Cost Control: Energy costs can be a significant part of your project budget. Monitoring energy consumption can help you control and reduce costs.

FLOW DIAGRAM:



Determine what specific aspects of energy consumption you want to measure. This could include electricity usage, fuel consumption, or any other relevant energy sources. Set up energy monitoring systems and sensors where energy is being consumed. This might involve installing smart meters, using IoT devices, or other data collection methods.Collect data from the monitoring systems. Use software or tools to analyze the data to gain insights into energy consumption patterns. Look for peaks and valleys in consumption that may indicate inefficiencies.

CONCLUSION:

The measurement of energy consumption is developed furtherly innovated in more steps and ways that are been developed with the question and answers.

PHASE-III

Import relevant python packages:

Let's use the electrical meter data to create clusters of typical load profiles for analysis. First we can load our conventional packages.

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
```

Next let's load all the packages we will need for analysis

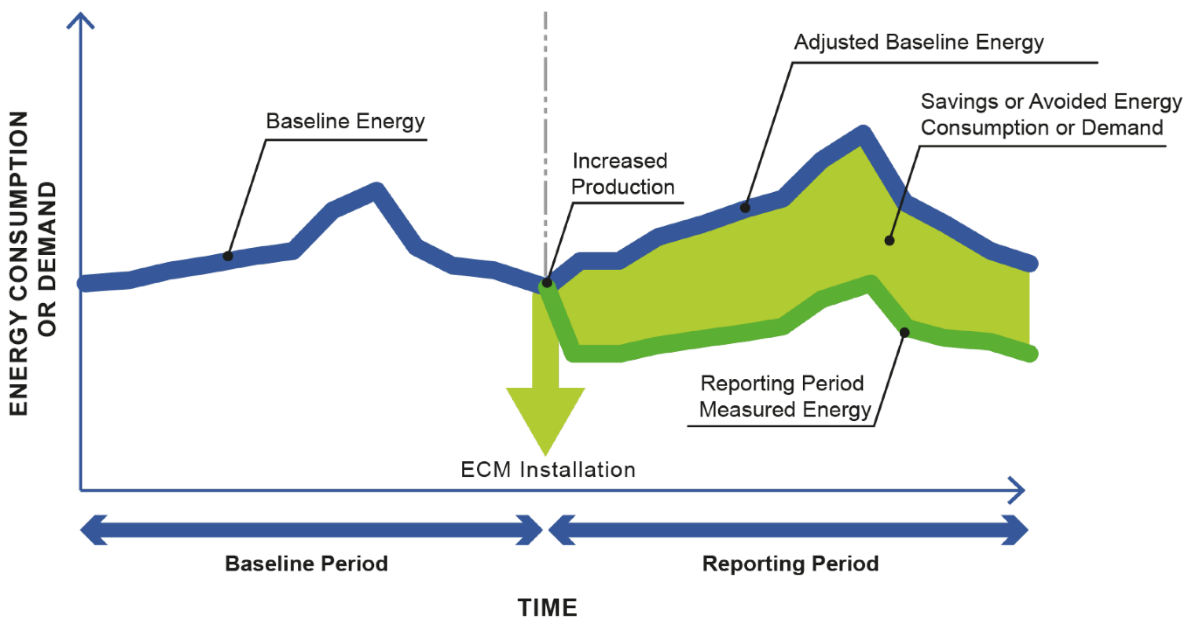
```
import sklearn
from sklearn import metrics
```

```
from sklearn.neighbors import KNeighborsRegressor
from scipy.cluster.vq import kmeans, vq, whiten
from scipy.spatial.distance import cdist
import numpy as np
from datetime import datetime
```

Electricity Prediction for Measurement and Verification

Prediction is a common machine learning (ML) technique used on building energy consumption data. This process is valuable for anomaly detection, load profile-based building control and measurement and verification procedures.

The graphic below comes from the IPMVP to show how prediction can be used for M&V to calculate how much energy **would have** been consumed if an energy savings intervention had not been implemented.



Load electricity data and weather data

First we can load the data from the BDG in the same as our previous weather analysis influence notebook from the Construction Phase videos

```
elec_all_data = pd.read_csv("electricity.csv",
parse_dates=True)
```

```
elec_all_data.info()
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3174 entries, 0 to 3173
Data columns (total 38 columns):
 #   Column                           Non-Null Count
Dtype
---  ------                           --------------  ---
--
 0   Utility.Number                   3174 non-null
int64
 1   Utility.Name                     3174 non-null
object
 2   Utility.State                    3173 non-null
object
 3   Utility.Type                     3174 non-null
object
 4   Demand.Summer Peak               3174 non-null
float64
 5   Demand.Winter Peak               3174 non-null
float64
 6   Sources.Generation               3174 non-null
float64
 7   Sources.Purchased                3174 non-null
float64
 8   Sources.Other                    3174 non-null
float64
 9   Sources.Total                    3174 non-null
float64
 10  Uses.Retail                      3174 non-null
float64
 11  Uses.Resale                      3174 non-null
float64
```

```
 12   Uses.No Charge                         3174 non-null   
float64
 13   Uses.Consumed                          3174 non-null   
float64
 14   Uses.Losses                            3174 non-null   
float64
 15   Uses.Total                             3174 non-null   
float64
 16   Revenues.Retail                        3174 non-null   
float64
 17   Revenue.Delivery                       3174 non-null   
float64
 18   Revenue.Resale                         3174 non-null   
float64
 19   Revenue.Adjustments                    3174 non-null   
float64
 20   Revenue.Transmission                   3174 non-null   
float64
 21   Revenue.Other                          3174 non-null   
float64
 22   Revenue.Total                          3174 non-null   
float64
 23   Retail.Residential.Revenue             3174 non-null   
float64
 24   Retail.Residential.Sales               3174 non-null   
float64
 25   Retail.Residential.Customers           3174 non-null   
float64
 26   Retail.Commercial.Revenue              3174 non-null   
float64
 27   Retail.Commercial.Sales                3174 non-null   
float64
 28   Retail.Commercial.Customers            3174 non-null   
float64
 29   Retail.Industrial.Revenue              3174 non-null   
float64
 30   Retail.Industrial.Sales                3174 non-null   
float64
 31   Retail.Industrial.Customers            3174 non-null   
float64
 32   Retail.Transportation.Revenue          3174 non-null   
float64
 33   Retail.Transportation.Sales            3174 non-null   
float64
 34   Retail.Transportation.Customers        3174 non-null   
float64
 35   Retail.Total.Revenue                   3174 non-null   
float64
 36   Retail.Total.Sales                     3174 non-null   
float6
```

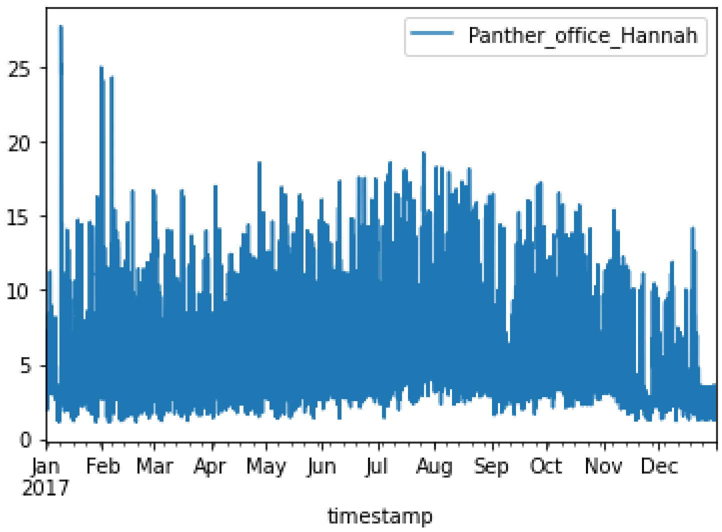
```
 37  Retail.Total.Customers          3174 non-null
float64
dtypes: float64(34), int64(1), object(3)
memory usage: 942.4+ KB
```

Load data:

```
office_example_prediction_data.plot()
```

```
<AxesSubplot:xlabel='timestamp'>
```

Output:




```
weather_data = pd.read_csv("../input/buildingdatagenomeproject2/weather.csv", index_col='timestamp', parse_dates=True)
```

```
weather_data_site = weather_data[weather_data.site_id == 'Panther'].truncate(before='2017-01-01')
```

```
weather_data_site.info()
```

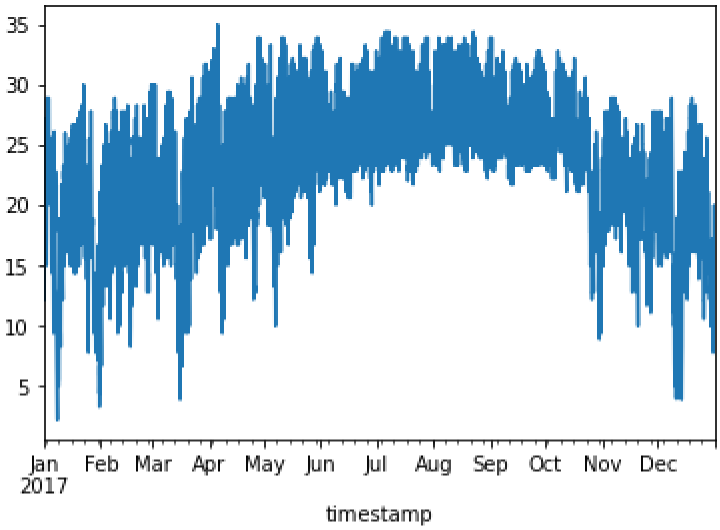
Output:

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 8760 entries, 2017-01-01 00:00:00 to 2017-12-31 23:00:00
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   site_id         8760 non-null   object
 1   airTemperature  8760 non-null   float64
 2   cloudCoverage   5047 non-null   float64
 3   dewTemperature  8760 non-null   float64
 4   precipDepth1HR  8752 non-null   float64
 5   precipDepth6HR  329 non-null    float64
```

```
6   seaLvlPressure  8522 non-null   float64
7   windDirection   8511 non-null   float64
8   windSpeed       8760 non-null   float64
```

```
weather_hourly = weather_data_site.resample("H").mean()weather_hourly_nooutlier = weather_hourly[weather_hourly > -40]weather_hourly_nooutlier_nogaps = weather_hourly_nooutlier.fillna(method='ffill')
```

```
temperature = weather_hourly_nooutlier_nogaps["airTemperature"]
temperature.plot()
```



Create Train and Test Datasets

The model is given a set of data that will be used to **train** the model to predict a specific objectice. In this case, we will use a few simple time series features as well as outdoor air temperature to predict how much energy a building uses.

For this demonstration, we will use three months of data from April, May, and June to prediction July.

```
training_months = [4,5,6]test_months = [7]
trainingdata = office_example_prediction_data[office_example_prediction_data.index.month.isin(training_months)]testdata = office_example_prediction_data[office_example_prediction_data.index.month.isin(test_months)]
trainingdata.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2184 entries, 2017-04-01 00:00:00 to 2017-06-30 23:00:00
Data columns (total 1 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Panther_office_Hannah  2184 non-null   float64
dtypes: float64(1)
memory usage: 34.1 KB
```

Encoding Categorical Variables:

```
train_features = pd.concat([pd.get_dummies(trainingdata.index.hour),

 pd.get_dummies(trainingdata.index.dayofweek),

 pd.DataFrame(temperature[temperature.index.month.isin(training_months)].values)], axis=1).dropna()
```

```
train_features.head()
```

Output:

0	1	2	3	4	5	6	7	8	9	...	22	23	0	1	2	3	4	5	6	0	
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	21.7
1	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	21.0
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	18.9
3	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	20.6
4	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	21.0

Train a K-Neighbor Model

This model was chosen after following the process in the cheat sheet until a model that worked and provided good results was found.

```
test_features = np.array(pd.concat([pd.get_dummies(testdata.index.hour),
                                    pd.get_dummies(testdata.index.dayof
week),
                                    pd.DataFrame(temperature[temperatur
e.index.month.isin(test_months)].values)], axis=1).dropna())
```

Use the Model to predict for the *Test* period

Then the model is given the test_features from the period which we want to predict. We can then merge those results and see how the model did

```
predictions = model.predict(test_features)
```

```
predicted_vs_actual = pd.concat([testdata, pd.DataFrame(predi
ctions, index=testdata.index)], axis=1)
```

```
predicted_vs_actual.columns = ["Actual", "Predicted"]
```

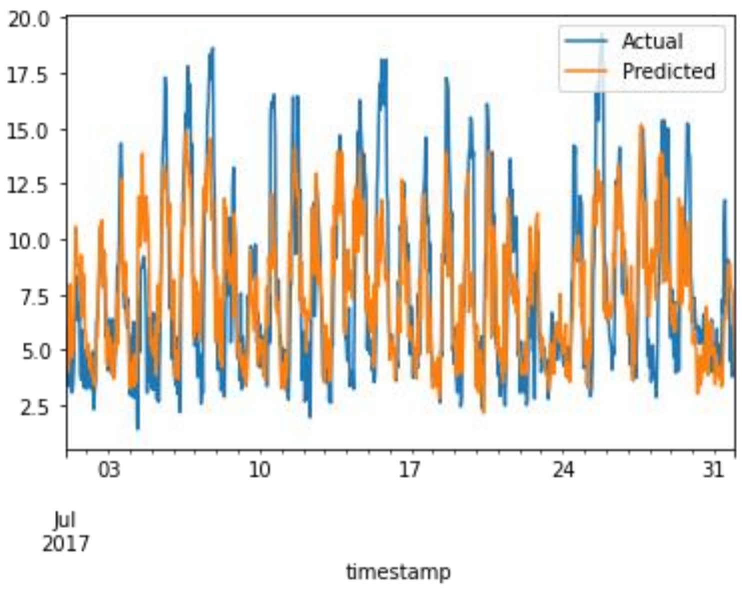
```
predicted_vs_actual.head()
```

timestamp		
2017-07-01 00:00:00	5.3370	5.75910
2017-07-01 01:00:00	3.8547	6.02898
2017-07-01 02:00:00	5.5751	4.39686
2017-07-01 03:00:00	4.1248	4.23180
2017-07-01 04:00:00	3.3497	4.03858

```
predicted_vs_actual.plot()
```

Out[27]:

```
<AxesSubplot:xlabel='timestamp'>
```

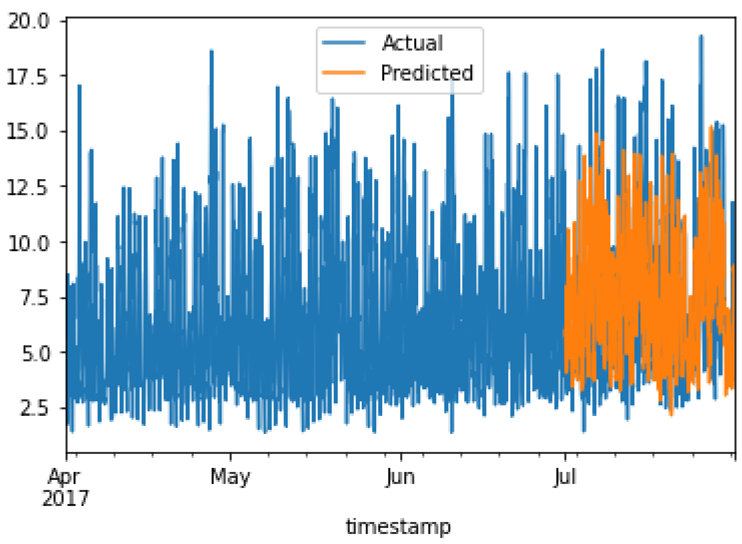



```
trainingdata.columns = ["Actual"]
```

```
predicted_vs_actual_plus_training = pd.concat([trainingdata,
predicted_vs_actual], sort=True)
```

```
predicted_vs_actual_plus_training.plot()
```

<AxesSubplot:xlabel='timestamp'>



Evaluation metrics

In order to understand quanitatively how the model performed, we can use various evaluation metrics to understand how well the model compared to reality.

In this situation, let's use the error metric Mean Absolute Percentage Error (MAPE)

In [31]:

```
# Calculate the absolute errorserrors = abs(predicted_vs_actual['Predicted'] - predicted_vs_actual['Actual'])# Calculate mean absolute percentage error 

MAPE = 100 * np.mean((errors / predicted_vs_actual['Actual'])

MAPE
```

34.22379683897996