NAAN MUDHALVAN

MEASURE ENERGY CONSUMPTION

FINAL DEVELOPMENT :

This research was done as part of a research seminar course at LAB University of Applied Science (fall 2022). The report was written using the thesis template of LAB University.

Abstract

The purpose of this research was to predict energy consumption using the data of Finland's transmission system operator. The objective of this project was to test if a machine learning model can yield good enough results in a complex forecasting problem, exploring machine learning techniques and developing a data-driven model for forecasting energy. The data contained 6-year hourly electrical consumption in Finland, and it is a univariate time series, as it is seasonal. We used a long-short term memory (LSTM) model to train the data. The model was evaluated using root mean squared error (RMSE) to be directly comparable to energy readings in the data. The result shows that electricity consumption can be predicted using machine learning algorithms so we can use the results to deploy renewable energy, plan for high/low load days, and reduce wastage from polluting on reserve standby generation.

Model Implementation

The data was imported from Finland's transmission system operator as a CSV file and then exported to a GitHub repository. There was a total number of 52965 observations and 5 variables in this dataset and no missing values were found. The minimum load volume is 5341 MWh, and the maximum load volume is 15105 MWh along with an average volume of 9488.750519 MWh. The data is univariate time series, where there is a need for one column to present time and another one to present energy consumption. For predicting day consumption, data were down-sampled using resample function. This function changed the data from hourly frequency to daily frequency. Training the LSTM model was done using the training set and the validation dataset for testing the results through the training process. The learning algorithm worked through the entire training dataset 60 times (Epoch), and the model weights were updated after each batch where the batch size is 20.

HISTORICAL DATA: When historical data are given by steps of 15 minutes, forecasts are required by steps of 15 minutes. When historical data are given by steps of 1 hour, forecasts are required by steps of 1 hour. When historical data are given by steps of 1 day, forecasts are required by steps of 1 day. • Timestamp - The time of the measurement • Value - A measure of consumption for that building
WEATHER FORECAST: It can be obtained for the OpenWeatherStation Website for the weather forecast of the place where the building is located. • Timestamp - The time of the measurement • Temperature - The temperature as measured at the weather station
BUILDING DATA: • DAY_OF_WEEK]IsDayOff - True if DAY_OF_WEEK is not a working day • Number of employees everyday
FEATURE EXTRACTION: Feature Extraction: In this phase we need to validate the predictive power of new features as well as existing features. There are many techniques applied to validate the feature importance such as correlation analysis, ensemble and tree based model based feature importance. Feature Transformation/Derivation: during the validation with a baseline model some of the feature may require transformation. These transformations include log transformation, Standard Scaling (SS) and Min Max Scaling (MMS). After literature survey and consultation with subject matter expert, a set of most desirable features for electricity load/consumption forecasting were listed. : a) Past consumption pattern: electrical consumption pattern cannot change abruptly until unless some major changes happen at the place. So past consumption pattern carries information for future consumption pattern.
b) Calendar: month, day of week, c) Demography: The population of building can affect the consumption pattern d) Geography:, temperature, etc. If temperature is high, people will use more electrical appliance and similarly when temperature is low.

ALGORITHMS AND MODELS TO BE USED FOR THE PREDICTION ANALYSIS:

LINEAR REGRESSION We'll train a Linear Regression model for predicting building energy consumption based on historical enregy data, several weather variables, hour of the day, day of the week, weekends and holidays. The accuracy obtained is 63.74% USING NEURAL NETWORKS: LSTMs (or long-short term memory networks) allow for analysis of sequential or ordered data with long-term dependencies present. Traditional neural networks fall short when it comes to this task, and in this regard an LSTM will be used to predict electricity consumption patterns in this instance. The accuracy obtained for this model is 97.12%

So we can conclude that deep neural networks such as LSTMs are more useful for prediction of energy consumption.

missing data was restructured based on the compressed details as predicted data points.

The imputation method was also evaluated to determine its performance. The resultant cleaned data was then further pre-processed using standardisation. Standardisation or

also known as Z-score normalisation is a transformation to change the observed data to have characteristics of standard normal distribution in which the mean is 0, and the standard deviation is 1. This transformed the data to be equally distributed above and below the mean value by using the formula in equation

where is mean and is standard deviation.

The standardisation process under Caret Package consists of 2 steps which are centring and scaling. The centring transformation computes the mean for a feature and subtracts it from each data point of the feature. On the other hand, the scale transformation computes the standard deviation for a feature and divides the output from centring transformation with the standard deviation.

3.3. Step 3: model development (training)

This research used a supervised machine learning methodology to predict energy consumption. After data was prepared, it was then inputted into the learning algorithm. Different feature combinations were fed into the algorithm to generate a candidate for the predictive model. Before using the data to create and train the model, data partitioning was done to separate the data into two groups – a training group and a testing group.

The predictive modeling for this research used a classification method to predict discrete variables instead of regressive prediction. As Azure ML does not have k-Nearest Neighbour and Artificial Neural Network for classification, the modeling function in Caret R package was utilised for all prediction to ensure uniform execution. Three types of machine learning algorithm were used for this research which were Artificial Neural Network (ANN-MLP), k-Nearest Neighbour (k-NN), and Support Vector Machine (SVM-RBF). Fig. 1 below shows the process after the data preparation until the generation of the predictive model.

```
import numpy as np # linear algebra

                                    import pandas as pd # data processing
```

```
df =
pd.read_csv('household_power_consumption
.txt', sep=';',
                parse_dates={'dt' :
['Date', 'Time']},
infer_datetime_format=True,
                low_memory=False,
na_values=['nan','?'], index_col='dt')


df.shape
```

The dataset contains 2,075,259 rows and 7 columns, let's take a look at the number of null values:

```
df.isnull().sum()
```

```
Global_active_power      25979
Global_reactive_power    25979
Voltage                  25979
Global_intensity         25979
Sub_metering_1           25979
Sub_metering_2           25979
Sub_metering_3           25979
dtype: int64
```

We have so many null values in the dataset, I will fill these null values with the mean values:

```
df = df.fillna(df.mean())
```

Data Visualization

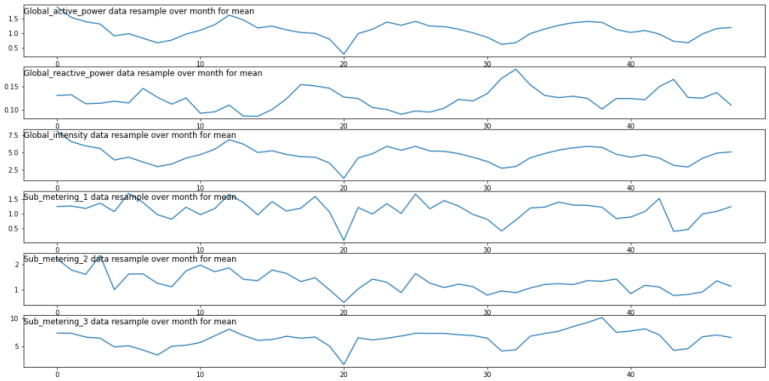
Let's have a look at the data more closely by visualizing it:

```
import matplotlib.pyplot as plt
```

```
i = 1

cols=[0, 1, 3, 4, 5, 6]

plt.figure(figsize=(20, 10))

for col in cols:

   plt.subplot(len(cols), 1, i)


plt.plot(df.resample('M').mean().values[
:, col])

   plt.title(df.columns[col] + ' data
resample over month for mean', y=0.75,
loc='left')

   i += 1

plt.show()
```

view raw

energy consumption.py hosted with ❤ by GitHub

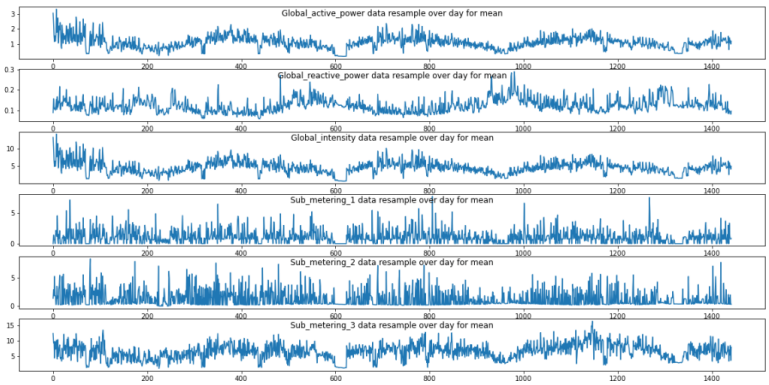



```
i = 1
```

```
cols=[0, 1, 3, 4, 5, 6]

plt.figure(figsize=(20, 10))

for col in cols:

   plt.subplot(len(cols), 1, i)


plt.plot(df.resample('D').mean().values[
:, col])

   plt.title(df.columns[col] + ' data
resample over day for mean', y=0.75,
loc='center')

   i += 1

plt.show()
```

view raw

energy consumption.py hosted with ❤ by GitHub




```
i = 1

cols=[0, 1, 3, 4, 5, 6]

plt.figure(figsize=(20, 10))

for col in cols:
```

```
    plt.subplot(len(cols), 1, i)

plt.plot(df.resample('H').mean().values[
:, col])

    plt.title(df.columns[col] + ' data
resample over hour for mean', y=0.75,
loc='left')

    i += 1

plt.show()
```

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

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

In [2]:

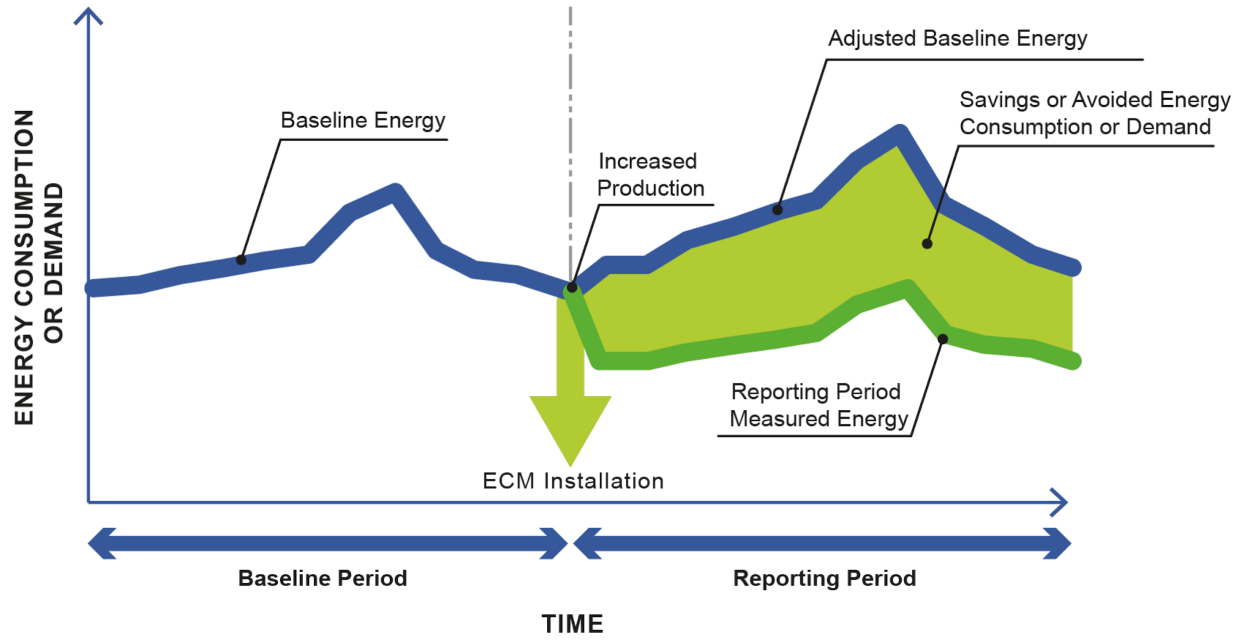
```
import sklearn
from sklearn import metrics
from sklearn.neighbors import KNeighborsRegressor
from scipy.cluster.vq import kmeans, vq, whiten
from scipy.spatial.distance import cdist
import numpy as np
from datetime import datetime
```

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.

Prediction for Measurement and Verification



There is an open publication that gives more information on how prediction in this realm can be approached: https://www.mdpi.com/2504-4990/1/3/56

There is an entire Kaggle Machine Learning competition also focused on this application: https://www.kaggle.com/c/ashrae-energy-prediction

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

In [3]:

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

In [4]:

```
elec_all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 17544 entries, 2016-01-01 00:00:00 to 2017-12-31 23:00:00
Columns: 1578 entries, Panther_parking_Lorriane to Mouse_science_Micheal
dtypes: float64(1578)

memory usage: 211.3 MB
```

Introduction to Machine Learning for the Built Environment: Energy Consumption Forecasting

This notebook is adapted from Dr. Clayton Miller. It uses the Building Data Genome Project data set to analyze electrical meter data from non-residential buildings.

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

In [1]:

```
import pandas as pd

import matplotlib.pyplot as plt

import matplotlib
```

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

In [2]:

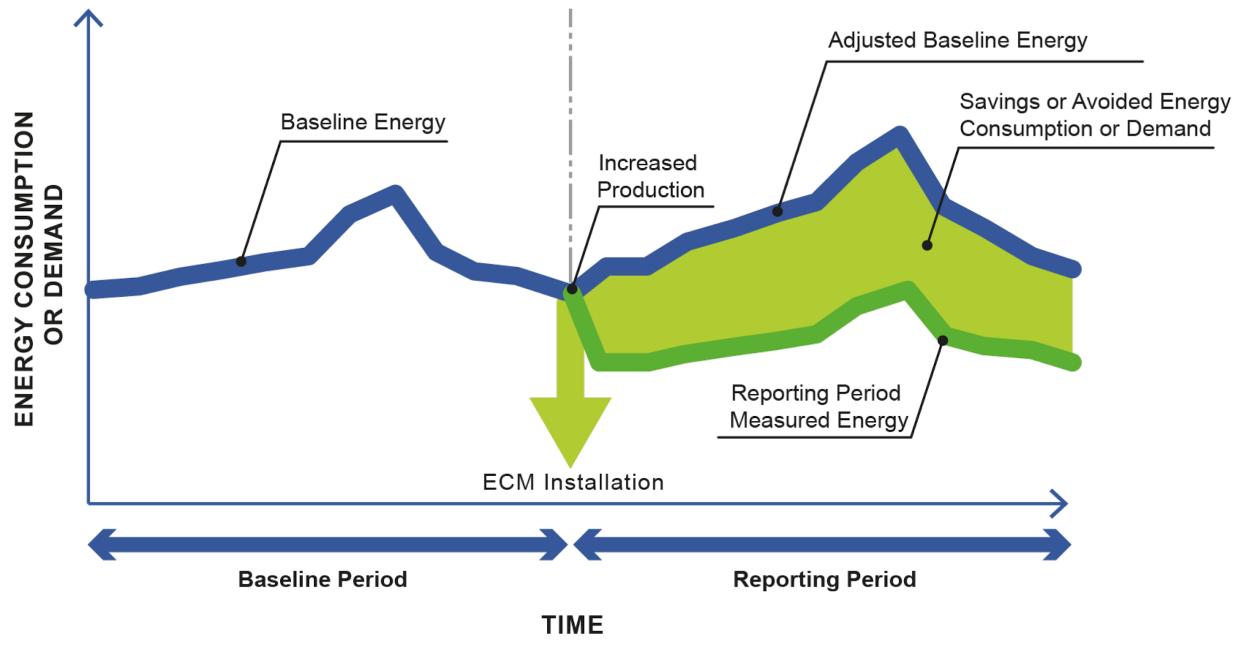
```
import sklearn

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.

Prediction for Measurement and Verification



There is an open publication that gives more information on how prediction in this realm can be approached: https://www.mdpi.com/2504-4990/1/3/56

There is an entire Kaggle Machine Learning competition also focused on this application: https://www.kaggle.com/c/ashrae-energy-prediction

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

In [3]:

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

In [4]:

```
elec_all_data.info()
```

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

DatetimeIndex: 17544 entries, 2016-01-01 00:00:00 to 2017-12-31 23:00:00

Columns: 1578 entries, Panther_parking_Lorriane to Mouse_science_Micheal

dtypes: float64(1578)

memory usage: 211.3 MB
```

In [5]:

linkcode

```
buildingname = 'Panther_office_Hannah'
```

In [6]:

```
office_example_prediction_data =
pd.DataFrame(elec_all_data[buildingname].truncate(before='2017-01-01')).fi
llna(method='ffill')
```

In [7]:

```
office_example_prediction_data.info()
```

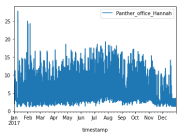
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 8760 entries, 2017-01-01 00:00:00 to 2017-12-31 23:00:00
Data columns (total 1 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Panther_office_Hannah  8760 non-null   float64
dtypes: float64(1)
memory usage: 136.9 KB
```

In [8]:

```
office_example_prediction_data.plot()
```

Out[8]:

<AxesSubplot:xlabel='timestamp'>



In [9]:

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

In [10]:

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

In [11]:

```
weather_data_site.info()
```

```
<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
dtypes: float64(8), object(1)
memory usage: 684.4+ KB
```

In [12]:

```
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')
```

In [13]:

```
temperature = weather_hourly_nooutlier_nogaps["airTemperature"]
```

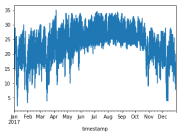
In [14]:

```
temperature.plot()
```

```
```

Out[14]:

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



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.

In [15]:

```
training_months = [4,5,6]

test_months = [7]
```

We can divide the data set by using the `datetime index` of the data frame and a function known as `.isin` to extract the months for the model

In [16]:

```
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)]
```

```
In [17]:
```

```
trainingdata.info()
```

```
<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
```

Introduction to Machine Learning for the Built Environment: Energy Consumption Forecasting

This notebook is adapted from Dr. Clayton Miller. It uses the Building Data Genome Project data set to analyze electrical meter data from non-residential buildings.

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

In [1]:

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

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

In [2]:

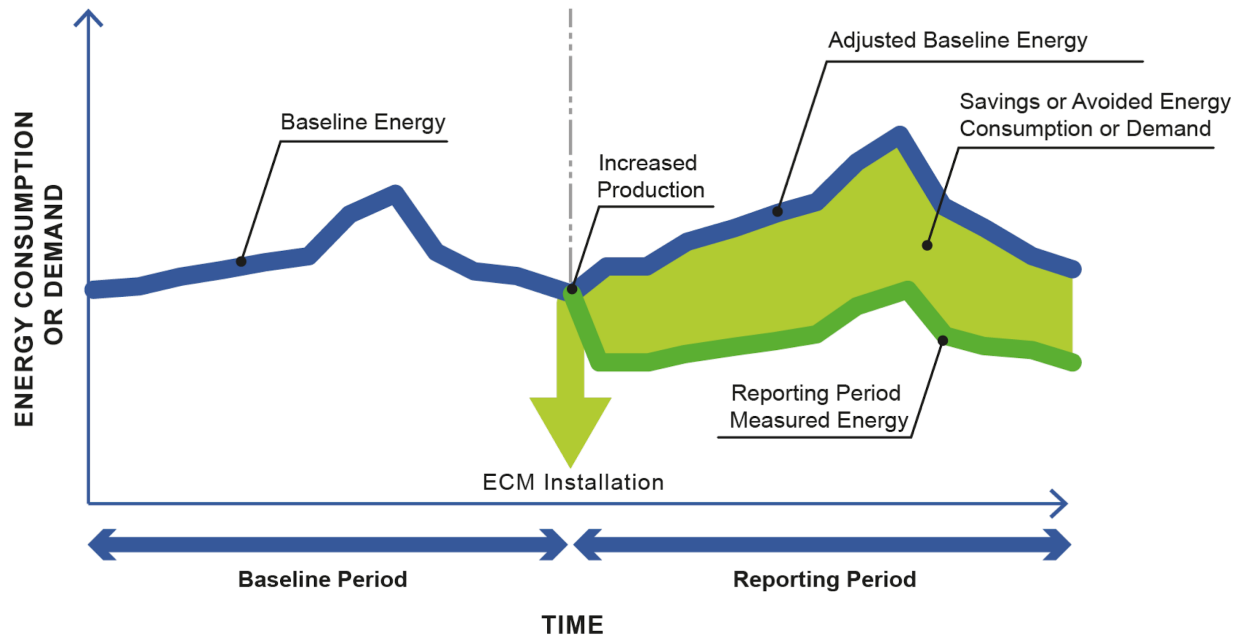
```
import sklearn
from sklearn import metrics
from sklearn.neighbors import KNeighborsRegressor
from scipy.cluster.vq import kmeans, vq, whiten
from scipy.spatial.distance import cdist
import numpy as np
from datetime import datetime
```

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.

Prediction for Measurement and Verification



There is an open publication that gives more information on how prediction in this realm can be approached: https://www.mdpi.com/2504-4990/1/3/56

There is an entire Kaggle Machine Learning competition also focused on this application: https://www.kaggle.com/c/ashrae-energy-prediction

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

In [3]:

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

In [4]:

```
elec_all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 17544 entries, 2016-01-01 00:00:00 to 2017-12-31 23:00:00
Columns: 1578 entries, Panther_parking_Lorriane to Mouse_science_Micheal
dtypes: float64(1578)
memory usage: 211.3 MB
```

In [5]:

```
buildingname = 'Panther_office_Hannah'
```

In [6]:

```
office_example_prediction_data =
pd.DataFrame(elec_all_data[buildingname].truncate(before='2017-01-01')).fi
llna(method='ffill')
```

In [7]:

```
office_example_prediction_data.info()
```

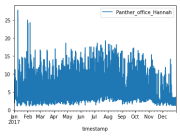
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 8760 entries, 2017-01-01 00:00:00 to 2017-12-31 23:00:00
Data columns (total 1 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Panther_office_Hannah  8760 non-null   float64
dtypes: float64(1)
memory usage: 136.9 KB
```

In [8]:

```
office_example_prediction_data.plot()
```

Out[8]:

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



In [9]:

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

In [10]:

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

In [11]:

```
weather_data_site.info()
```

```
<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
dtypes: float64(8), object(1)
memory usage: 684.4+ KB
```

In [12]:

```
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')
```

In [13]:

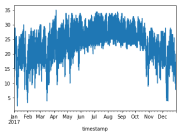
```
temperature = weather_hourly_nooutlier_nogaps["airTemperature"]
```

In [14]:

```
temperature.plot()
```

Out[14]:

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



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.

In [15]:

```
training_months = [4,5,6]
test_months = [7]
```

We can divide the data set by using the `datetime index` of the data frame and a function known as `.isin` to extract the months for the model

In [16]:

```
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)]
```

In [17]:

```
trainingdata.info()
```

```
<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
```

In [18]:

```
testdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 744 entries, 2017-07-01 00:00:00 to 2017-07-31 23:00:00
Data columns (total 1 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Panther_office_Hannah  744 non-null    float64
dtypes: float64(1)
memory usage: 11.6 KB
```

linkcode

We can extract the training input data features that will go into the model and the training **label** data which is what are are targeting to predict.

Encoding Categorical Variables

We use the pandas `.get_dummies()` function to change the temporal variables of *time of day* and *day of week* into categories that the model can use more effectively. This process is known as enconding

In [19]:

```
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()
```

In [20]:

```
train_features.head()
```

Out[20]:

	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

5 rows × 32 columns

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.

In [21]:

```
model = KNeighborsRegressor().fit(np.array(train_features),
np.array(trainingdata.values));
```

In [22]:

```
test_features = np.array(pd.concat([pd.get_dummies(testdata.index.hour),

pd.get_dummies(testdata.index.dayofweek),
```

```
pd.DataFrame(temperature[temperature.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

In [23]:

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

In [24]:

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

In [25]:

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

In [26]:

```
predicted_vs_actual.head()
```

Out[26]:

	Actual	Predicted

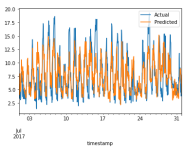
timestamp		
2017-07-01 00:00:00	5.3370	5.75910
2017-07-01 01:00:00	3.8547	6.02898
2017-07-01 02:00:00	5.5751	4.39686
2017-07-01 03:00:00	4.1248	4.23180
2017-07-01 04:00:00	3.3497	4.03858

In [27]:

```
predicted_vs_actual.plot()
```

Out[27]:

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



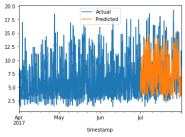
In [28]:

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

```
In [29]:
predicted_vs_actual_plus_training = pd.concat([trainingdata,
predicted_vs_actual], sort=True)
```

```
In [30]:
predicted_vs_actual_plus_training.plot()
```

```
Out[30]:

<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 errors
errors = abs(predicted_vs_actual['Predicted'] -
predicted_vs_actual['Actual'])
# Calculate mean absolute percentage error (MAPE) and add to list
MAPE = 100 * np.mean((errors / predicted_vs_actual['Actual']))
```

```
In [32]:
```

MAPE

```

```

Out[32]:

```
34.22379683897996
```