**Approach**

Name: **Mohd Azam**

Time series forecasting by Prophet Library

* Time series forecasting can be challenging as there are many different methods you could use and many different hyperparameters for each method.
* The Prophet library is an open-source library designed for making forecasts for univariate time series datasets. It is easy to use and designed to automatically find a good set of hyperparameters for the model in an effort to make skillful forecasts for data with trends and seasonal structure by default.
* *Implements a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects*
* It is designed to be easy and completely automatic, e.g. point it at a time series and get a forecast.
* The first step is to install the Prophet library using Pip
* Prophet requires data to be in Pandas DataFrames. Therefore, we will load and summarize the data using Pandas

**Load and Plot**

* A time-series dataset does not make sense to us until we plot it.
* Plotting a time series helps us actually see if there is a trend, a seasonal cycle, outliers, and more. It gives us a feel for the data.
* We can plot the data easily in Seaborn by calling the *lineplot()* function.
* We can clearly see the trend in sales over time and a monthly seasonal pattern to the sales. These are patterns we expect the forecast model to take into account.

**Filling missing values with interpolation**

* As we checked, there are some null values in our energy features, which could be filled with interpolate function with time as option as it would fill all the near values in the time series.

**Fit Prophet Model**

* To use Prophet for forecasting, first, a *Prophet()* object is defined and configured, then it is fit on the dataset by calling the *fit()* function and passing the data.
* The *Prophet()* object takes arguments to configure the type of model you want, such as the type of growth, the type of seasonality, and more. By default, the model will work hard to figure out almost everything automatically.
* The *fit()* function takes a *DataFrame* of time series data. The *DataFrame* must have a specific format. The first column must have the name ‘*ds*‘ and contain the date-times. The second column must have the name ‘*y*‘ and contain the observations.
* This means we change the column names in the dataset. It also requires that the first column be converted to date-time objects, if they are not already (e.g. this can be down as part of loading the dataset with the right arguments to *read\_csv*).

**Make an Out-of-Sample Forecast**

* We can achieve this by creating a future data frame by specifying the period of horizon and the type of frequency for which those periods are needed.
* This *DataFrame* can then be provided to the *predict()* function to calculate a forecast.
* The result of the predict() function is a *DataFrame* that contains many columns. Perhaps the most important columns are the forecast date time (‘*ds*‘), the forecasted value (‘*yhat*‘), and the lower and upper bounds on the predicted value (‘*yhat\_lower*‘ and ‘*yhat\_upper*‘) that provide uncertainty of the forecast.
* Finally, we made the final dataframe by taking our ‘row\_id’ as index and ‘yhat’ as ‘energy’ only.