**Electric Load Forecasting using different techniques**

***K-Nearest Neighbor Regressor***

*Introduction:*

The problem statement deals with hourly load forecasting. We have been given dataset from 2002 to 2005 (i.e. 4 years data) on hourly basis. I have used **K-Nearest Neighbor Regressor** technique for forecasting the load for 2006 year on hourly basis.

*Dataset:*

The Dataset file is named **‘history.csv**’. The dataset provided has 4 variable columns namely ‘Date’ in dd/mm/yy (as per python date stamp), ‘Hour’ ranging from 1 to 24, ‘Temperature’ & ‘Load’. The total **number of observations given is 35064**. Another excel file containing the 2006 data is provided named ’**fcst.csv**’ in which load column is kept blank to be forecasted.

*Data-Preprocessing:*

In order to check if the data has any seasonality, trend or just in simple terms is following the same pattern, we have plotted a line graph in Excel for years 2002(Series-1) to 2005(Series-4).

Hence, we can clearly see that the entire years load majorly (approximately) repeats itself. For our convenience we have dropped the date column. The cluster has been formed on basis of similar hour and temperature features.

*Methodology:*

The coding was done on Python (Jupyter Notebook). The KNN algorithm works on simple logic that the forecast point is obtained using similar historical data of the previous similar features. In our case we have fed the entire data from excel to the KNN Regressor. That is for forecasting the load at given hour at given temperature the KNN algorithm will use similar hour and temperature feature data. Other similar data will not be used and will not affect the forecast in any way. Some brief information on the ***KNN algorithm*** used.

* KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses ‘**feature similarity**’ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.
* k-NN is a type of [instance-based learning](https://en.wikipedia.org/wiki/Instance-based_learning), or [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning), where the function is only approximated locally and all computation is deferred until function evaluation.
* The best choice of k depends upon the data; generally, larger values of k reduces effect of the noise on the classification but make boundaries between classes less distinct. A good k can be selected by various [heuristic](https://en.wikipedia.org/wiki/Heuristic_(computer_science)) techniques (see [hyperparameter optimization](https://en.wikipedia.org/wiki/Hyperparameter_optimization)). The special case where the class is predicted to be the class of the closest training sample (i.e. when k = 1) is called the nearest neighbor algorithm.
* The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance.

A screenshot of a cell phone

Description automatically generated

In our case the target variable is Load and the feature variables are Temperature & Hour. The logic behind the code runs like this; the forecast for the query point is the average of the loads corresponding to the similar hour and temperature loads.

*Code and Theory:*

The following libraries have been used:-

**NumPy:**

It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

**Pandas:**

In computer programming, pandas are a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

**Matplotlib:**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.

**Seaborn:**

Seaborn is a Python data visualization library based on [matplotlib](https://matplotlib.org/). It provides a high-level interface for drawing attractive and informative statistical graphics.

**Scikit-learn:**

Scikit-learn (also known as sklearn) is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language. It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including  [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [k-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy).

The below mentioned code is for the month of January and it has been run 11 more times for rest months to get the forecast.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A picture containing comb, fence

Description automatically generated

A screenshot of a cell phone

Description automatically generated

***We have divided the data in 70:30 % for training & testing purpose***.

*Parameter Selection:*

The best choice of k depends upon the data; generally, larger values of k reduces effect of the noise on the classification but make boundaries between classes less distinct. A good k can be selected by various [heuristic](https://en.wikipedia.org/wiki/Heuristic_(computer_science)) techniques (see [hyperparameter optimization](https://en.wikipedia.org/wiki/Hyperparameter_optimization)). The special case where the class is predicted to be the class of the closest training sample (i.e. when k = 1) is called the nearest neighbor algorithm.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A close up of a newspaper

Description automatically generated

.

.

.

.

.

A screenshot of a computer

Description automatically generated

**As we can see we have run the code to find the optimum value for K in the range 1-100. We check for the RMSE value for different K values and compare them. We can see that the RMSE value is 127380.64 for K= 44.**

We have also plotted a graph for K Value vs RMSE value.

For a very low value of k (suppose k=1), the model overfits on the training data, which leads to a high error rate on the validation set. On the other hand, for a high value of k, the model performs poorly on both train and validation set. If you observe closely, the validation error curve reaches a minima at a value of k = 44. This value of k is the optimum value of the model (it will vary for different datasets). This curve is known as an ‘**elbow curve**‘ (because it has a shape like an elbow) and is usually used to determine the k value.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

The forecasted file gets saved in array.csv file for temporary purpose.

*Conclusion:*

Hence, we have used the simple K-nearest neighbor model for forecasting year ahead hourly based load. Our model is working.

*The Mean Average Percentage Error is* ***10.56%.***