ENERGY CONSUMPTION

Abstract

Building Energy Management System (BEMS) has been a substantial topic nowadays due to its importance in reducing energy wastage. However, the performance of one of BEMS applications which is energy consumption prediction has been stagnant due to problems such as low prediction accuracy. Thus, this research aims to address the problems by developing a predictive model for energy consumption in Microsoft Azure cloud-based machine learning platform. Three methodologies which are Support Vector Machine, Artificial Neural Network, and k-Nearest Neighbour are proposed for the algorithm of the predictive model. Focusing on real-life application in Malaysia, two tenants from a commercial building are taken as a case study. The data collected is analysed and pre-processed before it is used for model training and testing. The performance of each of the methods is compared based on RMSE, NRMSE, and MAPE metrics. The experimentation shows that each tenant's energy consumption has different distribution characteristics.

1. Introduction

Recently, smart building concept has been adapted more frequently as an initiative to create an intelligent space area by taking advantage of the rapid development of computational and communication architecture (Cheng and Kunz, 2009). This concept is not only limited to Malaysia but other countries as well. General public understanding of smart building concept rotates on the idea of automated process, which is able to automatically control the building's operation through the usage of instrumentation measures and microcontrollers in two-way communication (Qolomany et al., 2019). Other than automated control, a smart building also consists of an intelligent system which provides energy consumption forecasts as an energy efficiency initiative. This is due to its advantage of yielding economical savings and as a sustainable approach for energy management to minimize energy wastage (Xu et al., 2018).

A smart energy consumption forecasting is important, especially for buildings as buildings' energy usage is increasing and almost reaches 40% of primary energy use in developed countries (Berardi, 2015). In Malaysia alone, energy consumption has been increase gradually due to the growth of population. The growth of population lead to the increasing of energy demand in this country and have been estimated to reach 116 million tons of oil equivalents (mtoe) by this year. Energy provided in Malaysia is influenced by the main fossil fuel sources which included coal, natural gas and fuel oil. Buildings which including commercial, residential and industrial in our country utilises a total of 48% of the electricity that have been created (Hassan et al., 2014). The increasing of energy consumptions towards buildings from day to day create enforcement to this country in managing and reducing the energy consumption as much as possible in order to improve energy efficiency.

2. Literature review

Machine learning prediction methodology

Following the aforesaid problem, this research addresses its challenges by conducting prediction modeling through the evaluation of historical power data. This method utilises a data-driven approach as described by Corgnati et al. (2013) whereby the input (regressor variables) and output variables (response) are known. Based on this data, system parameters will be estimated and thus, a mathematical model could be generated. Several previous studies have analysed the data-driven machine learning approach. Fu et al. (2015) proposed using one of ML algorithms which is Support Vector Machine (SVM) to predict the load at a building's system-level (air conditioning, lighting, power, and others) based on weather predictions and hourly electricity load input. Overall, SVM method managed to predict the total electricity load with root mean square error (RMSE) of 15.2% and mean bias error (MBE) of 7.7%. Findings by Valgaev et al. (2016) proposed a power demand forecast using k-Nearest Neighbour (k-NN) model at a smart building as part of the Smart City Demo Aspern (SCDA) project. The k-NN forecasting method was approached using a set of historical observations (daily load curves) and their successors. The k-NN method is good at classifying data but limited in forecasting future value as it only identifies similar instances in large feature space. Therefore, it must be complemented with temporal information identification whereby the prediction will be made for the next 24 h during workdays.
Five methods of machine learning techniques were used for short-term load forecasting by El Khantach et al. (El Khantach et al., 2019) with an initial decomposition of the historical data done periodically into time series of each hour of the day, which finally constituted 24-time series that represented every past hour. The five machine learning methods used are multi-layer perceptron (MLP), support vector machine (SVM), radial basis function (RBF) regressor, REPTree, and Gaussian process. The experimentation was done based on data derived from the Moroccan electrical load data. The result showed that MLP method came out as the most accurate with MAPE percentage of 0.96 while SVM came second and although far from the result of MLP, it was still better than the rest. Although the prediction of energy consumption usually uses a classification-based machine learning method, prediction could also be made based on the regression method as studied by González-Briones et al. (2019). The research constructed a predictive model by analysing the historical data set using Linear Regression (LR), Support Vector Regression (SVR), Random Forest (RF), Decision Tree (DT) and k-Nearest Neighbour (k-NN). The parameters of the research used one day-before electricity consumption (kWh) as an additional attribute. The results showed that LR and SVR models had the best performance with 85.7% accuracy.

Management of missing data

Techniques in handling missing data have been vastly studied before and methodologies have been deduced. There are two types of methodology that are removing the portion of the data which has missing value and imputation method which is based on close estimation (Hegde et al., 2019). The first method which omit the missing part of data is not feasible as this causes valuable information to be removed (Manly and Wells, 2015). Without the data, a biased estimation would be made. Therefore, the imputation method is a preferable

technique. Newgard and Lewis (2015) presented several imputation techniques such as Mean Value Imputation, Last Observation Carried Forward, Maximum Likelihood Estimate (MLE) and Multiple Imputation (MI). The mean value imputation basically substitutes the missing data with the mean value of the dataset. However, this method is not suitable for data which is not strictly random as it will introduce inequality in the data (Kang, 2013). Another method presented was Last Observation Carried Forward, in which imputation is made for historical data that was collected through (Newgard and Lewis, 2015).
The more advanced methods presented were Multiple Imputation (MI) and Maximum Likelihood Estimate (MLE) (Newgard and Lewis, 2015). The Multiple Imputation method substitutes the missing data by gradually supplanting the missing data for every iteration made. This method utilises statistical analysis based on observed data to handle the uncertainty that is introduced by the missing portion. An example of a popular MI method is Multiple Imputation Using Chained Equations (MICE) (Azur et al., 2011). Maximum Likelihood Estimate conducts substitution through assumption made by initially identifying the parameters and boundaries based on the distribution of the data. The imputation would then be made based on the assumed parameters. This method of imputation was employed by Probabilistic Principal Component Analysis (PPCA). Both advanced imputation methods have been compared by Hegde et al. (2019) in which imputation method was made on sampled dataset consisting of 87 numeric-converted categorical variables and 29 continuous variables. The study used RMSE metrics to evaluate imputation technique performance. From the research, the PPCA method showed a much promising result compared to MICE, in which 65% of data variables were successfully imputed by PPCA and only 38% correct imputation by MICE. This was further supported by Schmitt et al. (Schmitt et al., 2015) wherein the research compared the performance of six imputation methods including PPCA and MICE on a real dataset of various sizes. The result showed that MICE managed to perform well in a small dataset, but in a large dataset case, the MICE method performed poorly.

Employment of cloud-based prediction modeling

There are various available cloud-based prediction modeling platforms which are able to support machine learning process with additional capabilities to analyse big data and streaming data. When selecting the preferable machine learning tool and platform, several important factors need to be considered such as ascendable, pace, scope, practicability, flexibility, and programming language (Landset et al., 2015). Machine learning platform and tool are observed to be frequently utilised with big data techniques for real-time analytics. Review of previous papers on utilising these ML platforms with big data analytics shows three top platforms that are widely used for research which are Apache Spark's Machine Learning Library (Spark MLlib), TensorFlow, and Microsoft Azure Machine Learning Studio (Azure ML). Apache Spark ML library provides a platform which generally uses Machine learning algorithms for regression, classification, and clustering (Quddus, 2018).
Literature by Pérez-Chacón et al. (2016) showed k-means algorithm in Spark MLlib was used to observe the electric energy consumption behaviour in a big time series-based data. The results showed that the software managed to discover a day-based consumption pattern with low computing power for big data sizing up to 3 years (2011, 2012 and 2013) for two buildings of a public university. Another type of platform is TensorFlow, which is an open-source library, that is focuses more on deep learning and reinforcement learning technique (Ramsundar and Zadeh, 2018). Under Apache 2.0 open-source licensing, TensorFlow development was then initiated by Google Brain team. The software's name, TensorFlow, basically explains its framework whereby it implements a data flow graph,

consists of “tensors” (data batch which will be processed) and “flows” (data motion in the system) (Abadi et al., 2016). On the desktop, it is able to utilise both CPUs and GPUs resources such as in the study by Cai (2019), where the research implemented Convolutional Neural Networks (CNN) in TensorFlow. The computational platform which consists of Intel Core i7 5820 K and Nvidia GeForce GTX Titan X are both utilised to provide a learning-based power and runtime modeling.
The machine learning approach at the enterprise level has also emerged suddenly due to the introduction of a scalable data framework for handling big data. These companies would usually go for Microsoft Azure Machine Learning Studio (AzureML) as it utilise cloud-based predictive analytics, thus requires less investment in hardware for conducting analysis (Mateev, 2019). Such example of a company is British Petroleum (BP) whereby Azure Artificial Intelligent is utilised to improve their safety performance and work efficiency in terms of exploring potential new energy by generating useful model within lesser time (Microsoft Customer Stories, 2019). In addition to its functionality, AzureML supports Python and R as its external programming (AzureML Team, 2016). Moreover, Azure ML platform can provide machine learning service from the start until the generation of a predictive model and is able to continue with the next step, whether publishing or deploying the model to a website or other platforms (Qolomany et al., 2019).

3. Methodology

Based on the reviewed studies regarding the critical areas, this research conducts energy consumption prediction using a dataset previously collected from a commercial building located in Klang Valley Malaysia, from June 2018 until December 2018. The commercial building is equipped with IoT meters that are connected to the power inlet socket at two major tenants of the building. Each tenant is divided into two areas consisting of two IoT meters named tenant A1, A2, B1 and B2. Collected data per minute that mapped into Tenaga National Berhad (TNB) requirements was saved in an open-source web server. Collected data can be extracted manually from the online platform in the form of a CSV file. The prediction method will employ 3 machine learning algorithms which are k-NN, SVM and ANN. Feature attributes for this prediction will use electrical power data consisting of power factor, voltage and current, in which the demand would be the targeted output. The prediction modeling will be conducted inside Microsoft Azure Machine Learning Studio (AzureML) utilising R programming language (Caret Package). Microsoft Azure has been chosen as a platform in this research based on the literature reviewed in the previous section. Before model training and testing, the raw data will initially be analysed and pre-processed to reduce the complexity of the model training and to manage any missing data. Finally, each model will be evaluated using validation metrics. Consequently, the energy consumption prediction framework will consist of four parts, which are:

- Step 1
  Normality testing of dataset
- Step 2
  Data pre-processing
- Step 3
  Model development (training)
- Step 4
  Model evaluation (test)

Step 1: normality testing of dataset

In this research data analysis, a normality testing of the dataset for each tenant was conducted to determine the dataset distribution. This process was orchestrated by identifying the skewness and kurtosis of the dataset. Normality testing is significant for model development as it usually assumes that the dataset was normally distributed. Based on the background research by Mishra et al. (2019), it was found that normality testing is ignorable if the sample size exceeds 100. However, understanding of the dataset distribution could provide a consequential analysis for the result of the prediction. On the ground of statistical analysis, skewness is defined as the measure of irregular probability distribution around the mean value whereas kurtosis is a quantification of the distribution peakness. The formula for skewness and kurtosis is as shown in equations (1), (2), respectively

3.3. Step 3: model development

$$\text{Skewness, } S = \sum_{i=1}^{N} (x_i - \mu)^3$$

$$\text{Kurtosis, } K = \sum_{i=1}^{N} (x_i - \mu)^4$$

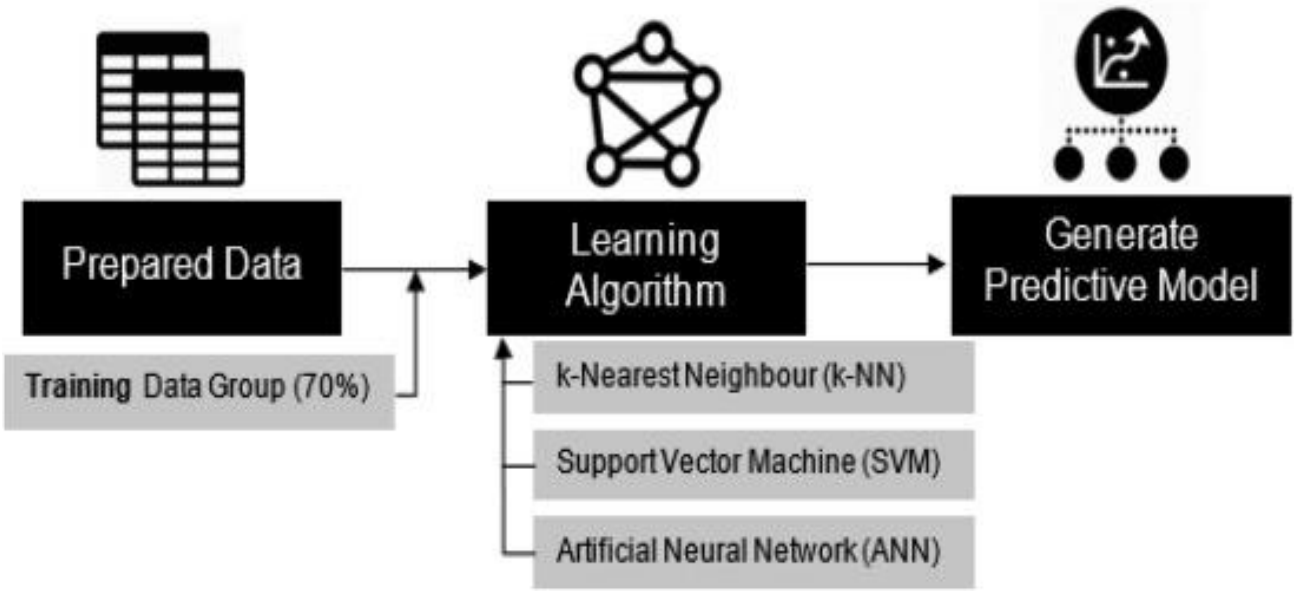
(training)

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.

k-Nearest Neigbour (k-NN)

The first predictive model to forecast energy consumption used the k-Nearest Neighbour (k-NN) method. This machine learning method is frequently used due to its simple criteria and its forecasting capability on intricate non-linear pattern (Valgaev et al., 2016). It manages to provide prediction by determining similar instances between data points in feature space (González-Briones et al., 2019). For this research, the method was used to predict maximum demand by using voltage, current and power factor as the features as the resultant multiplication of the values will output the electrical power usage (kW). Based on the Euclidean distance function in equation (4), k-NN method was trained repeatedly up to the maximum tuning parameter (k-value) which equals to 49. The resultant model from the training with the lowest RMSE value was chosen for prediction



k-Nearest Neigbour (k-NN)

The first predictive model to forecast energy consumption used the k-Nearest Neighbour (k-NN) method. This machine learning method is frequently used due to its simple criteria and its forecasting capability on intricate non-linear pattern (Valgaev et al., 2016). It manages to provide prediction by determining similar instances between data points in feature space (González-Briones et al., 2019). For this research, the method was used to predict maximum demand by using voltage, current and power factor as the features as the resultant multiplication of the values will output the electrical power usage (kW). Based on the Euclidean distance function in equation (4), k-NN method was trained repeatedly up to the maximum tuning parameter (k-value) which equals to 49. The resultant model from the training with the lowest RMSE value was chosen for prediction

$$\text{Euclidian distance function} = \sqrt{\sum_{i=1}^{k} (xi - yi)\,2} \quad (4)$$

Support Vector Machine (SVM)

In this research, the Support Vector Machine (SVM) was used with Radial Basis Function (RBF) as its kernel function. This methodology is usually known as a maximum margin classifier and is utilised to tackle problems regarding classification and regression for a large dataset (Ben-Hur et al., 2008). There are several kernel selections available for SVM method. In this study, Radial Basis Function (RBF) as shown in equation (5), was chosen due to the broad and non-linear characteristics of the dataset.(5)����(�,�′)=exp[−��−�′2]

$$K_{RBF}\left(x, x'\right) = \exp\left[-\frac{x - x^{'\,2}}{2\sigma^2}\right]$$

here γ is a gamma parameter to determine the spread distribution of the kernel and $x - x'$ is the Euclidean distance between the set of points.
Equation (5) has a corresponding definition as represented in equation

$$K_{RBF}\left(x, x'\right) = \exp\left[-\gamma x - x^{'\,2}\right]$$

Artificial Neural Network (ANN)

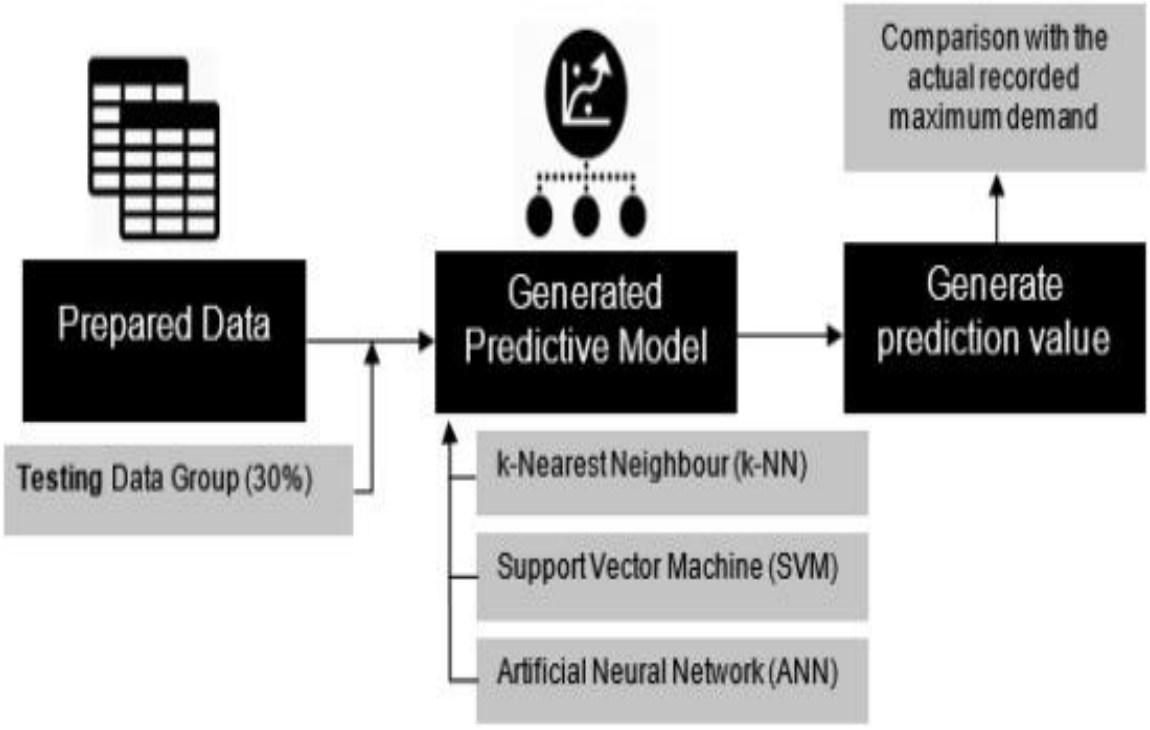
The third methodology in this research for energy consumption prediction was Artificial Neural Network (ANN). The advantages of using ANN such as its capability to learn complex behaviour, makes it widely used for predictions and pattern recognition (Karunathilake and Nagahamulla, 2017). ANN model structure consists of a formation of interconnected neurons that have three main layers; input layer, hidden layer, and output layer. By comparing the initial output with the desired output, adjustment of the synaptic weight of each link that connects between the neurons was made until the difference is minimal (minimising Sum Squared Error (SSE)); this would provide regularization for the model (Liu et al., 2019). The weight is the representation of the priority or importance of the neuron input. For this research, a Multilayer Perceptron Model (MLP) type of ANN structure with error back propagation learning algorithm was used for its network solution structure. In the hidden layer, a suitable non-linear transfer function was used to compute the information accepted by the input layer. The ANN model is as shown in equation

$$y_t = \alpha_0 + \sum_{j=1}^{n} \alpha_j f\left(\sum_{i=1}^{m} \beta_{ij} y_{t-i} + \beta_{0j}\right) + \varepsilon_t \qquad (7$$

where m is the number of input nodes, n is the number of hidden nodes, f is the Sigmoid Transfer function, $\{\alpha_j,\ j = 0,1,\ldots n\}$ is the vector of weights from the hidden layer to the output layer and $\{\beta_{ij},\ i = 0,1,\ldots m;\ j = 0,1,\ldots n\}$ is the weight from the input to the hidden nodes.

Step 4: model evaluation (test)

Before inputting the data to the machine learning algorithm, the data was partitioned into two groups whereby 70% of the dataset was used for training and the other 30% was partitioned as testing data groups. The training groups of data were used to train each machine learning algorithm and generate a predictive model that could output value that matches with the recorded maximum demand data while the rest of the data was held back to be used to test the trained predictive model. The process is as illustrated



4. Results and discussion

The results of the experimentation were discussed in sections based on the steps of the predictions' framework. The pre-processing method and imputation of missing data using modules provided by both Microsoft Azure Machine Learning Studio and Caret Package were discussed in terms of procedure, effects, and importance. The findings regarding energy consumption prediction were reviewed for each tenant and performance comparison was provided for the prediction result of Artificial Neural Network (ANN), Support Vector Machine (SVM) and k-Nearest Neighbour (k-NN). Holistically, utilisation of Azure ML Studio as a forecasting medium was inspected to identify its reliability and compatibility to predict energy consumption.

Normality testing of dataset

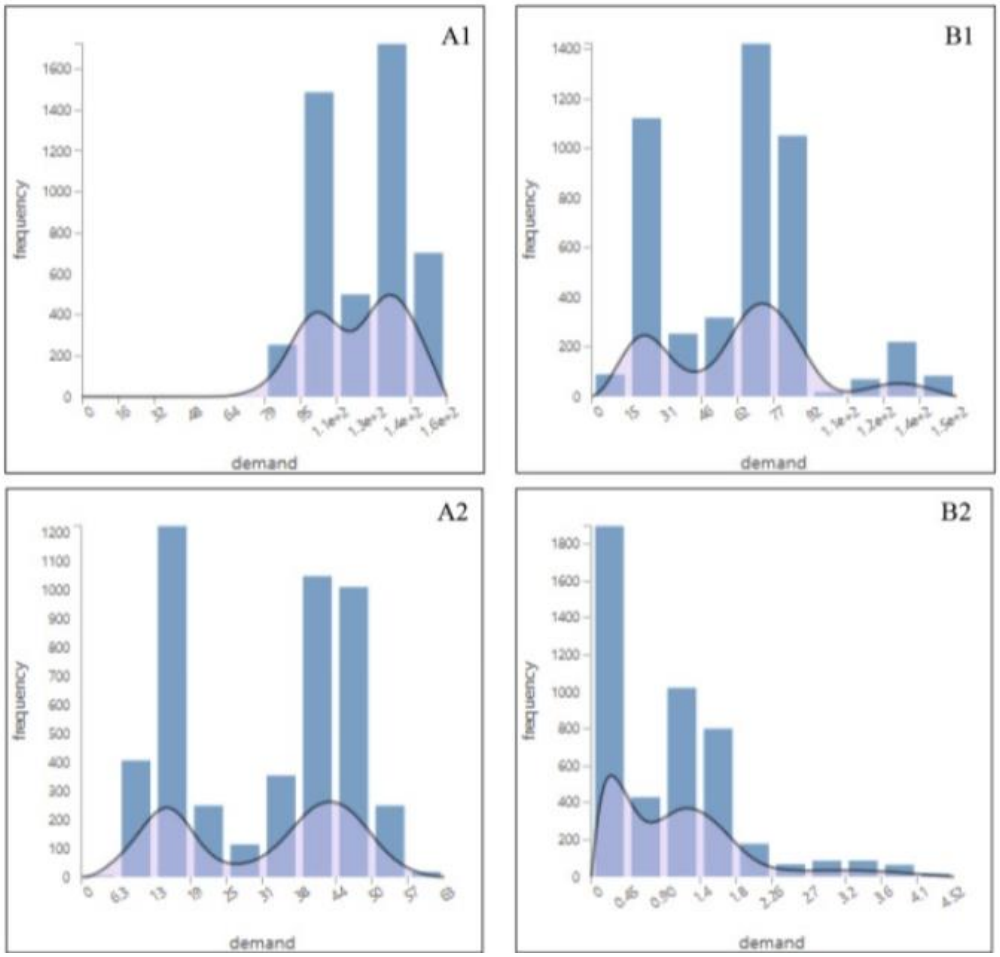
For the normality test of the energy demand data, the skewness and kurtosis values were calculated using the aggregate data for each tenant, starting from June 2018 until December 2018. This assessment was intended to analyse whether the shape of data affects the performance of the developed predictive model. The generated data was compiled in Table 1 and Table 2, for skewness and kurtosis, respectively. Additionally, Fig. 3 shows the form of the dataset for a graphical assessment of normality. The skewness and kurtosis values were computed for all attributes, including the demand.

Table 1. Skewness for each tenant using aggregate data.

Tenants	Skewness			
	Power Factor	Current	Voltage	Demand
A1	−0.457296	−0.17541	−0.116392	−0.279034
A2	−1.564851	−0.105746	−0.054989	−0.182159
B1	−1.877798	−0.666165	−0.310973	0.282481
B2	1.735751	0.202681	−0.714503	1.267578

Table 2. Kurtosis level for each tenant using aggregate data.

Tenants	Kurtosis			
	Power Factor	Current	Voltage	Demand
A1	6.026131	−1.648018	−0.45877	−1.053977
A2	2.839891	−1.763198	−0.768862	−1.584641
B1	2.062267	−1.011822	0.469322	−0.126043
B2	3.472321	−1.403968	4.623413	1.824909



Probability density for tenant A1, A2, B1 and B2.

Based on Table 1 and Fig. 3, Tenant A1, A2, and B1 dataset was approximately symmetry and skewed with bimodal shape density. Nevertheless, for Tenant A1, the density was skewed left, as the long tail pointed to the left whereas for Tenant B1, the density was skewed right. Tenant A2 shows asymmetry normal whereby the tails between each end were approximately balanced. Different from Tenant B2, the distribution was highly skewed as the skewness was more than 1 at 1.267578. The density plot of Tenant B2 distribution shows that the density was also bimodal with right-skewed.

In terms of kurtosis, from Table 2, Tenant A1, A2 and B1 had excess kurtosis less than 0. This means that the distributions were platykurtic. This characteristic was also observed based on Fig. 3 (A1, A2, B1) whereby the probability density plot has a broader tail, and the peak centre was wider. Contrary to Tenant B2, the excess kurtosis was higher than 0 at 1.824909. This indicates that Tenant B2 distribution was leptokurtic and had a higher variance. From this normality testing, Tenant A1 and A2 had an approximately normal distribution. However, Tenant A1 has a mean value less than the median. Tenant B1 had also an approximately normal distribution but had slightly higher skewness and kurtosis compared to Tenant in department A. Tenant B2 dataset was highly skewed and had a higher variance in comparison with the other tenants.

Imputation of missing data

The study of missing data was conducted inside Azure ML studio whereby Summarize Data module was utilised to determine the amount of missing data and to reveal the rows which have missing data. The diagnosis of missing data was also conducted via observation on the value of other attributes in the same row. Table 3 shows a summary of the analysis.

Table 3. Number of missing data for all tenants.

Tenants	Number of missing data				Total number of missing data
	P. Factor	Current	Voltage	Demand	
A1	0	0	0	1	1
A2	0	0	0	3	3
B1	0	0	0	1	1
B2	0	0	0	171	171

Briefly, the total number of data for A1 is 4666, A2 is 4666, B1 is 4648 and B2 is 4648. Based on Table 3, it was noticed that the dataset only had a missing value for demand, which was the targeted output. It was also observed that Tenant B2 had the highest number of missing data. The identification of the missing data mechanism was made by referring to the power formula and the method of data generation. The voltage and current values were captured by the voltage and current sensors. On the other hand, the values for power demand and power factor were generated bases on analysing the magnitude and waveform for both voltage and current. The electrical power formula also shows that the power value was the multiplication result of the three attributes. This indicates that demand and power factor variables are dependent on voltage and current value. Observation in Table 3 shows that although power factor, current and voltage attributes had value, the respective demand output was missing. This deduces that the missing value was Missing Completely at Random (MCAR), in which the missingness of demand data does not depend on the observed attributes. The missing value for this data was then imputed using the PPCA method utilising the Clean Missing Data module in Azure ML Studio. Although data MCAR was negligible, the dataset was not ignored as it would discard valuable information. Understanding the mechanism of the missing data has provided proper guidance for the imputation process of missing data. This prevents the missing data from being imputed with the wrong value, which could generate more outliers. Continuing with missing data imputation, Probabilistic Principal Component Analysis was chosen as the configuration for the imputation method utilising AzureML Clean Missing Data module. Fulfilling the objective of this research, the cleaning method was evaluated based on Raw Bias (RB), Coverage Rate (CR) and R-Squared criteria. Table 4 shows the result of the evaluation for each tenant.

Table 4. Evaluation of PPCA based imputation using AzureML Clean Missing Data Module.

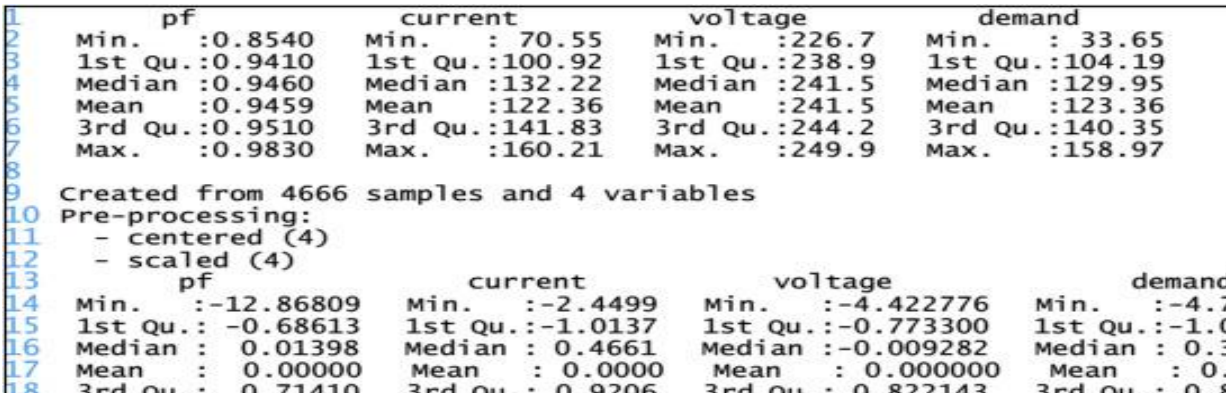
Tenants	Raw Bias	Coverage Rate	R-Squared
A1	−0.02813237	0.9003215	0.8997098
A2	−0.006479131	0.9011366	0.9039039
B1	0.1121446	0.9014418	0.9051205
B2	−0.000560815	0.9014965	0.90587942

From the raw biased result, PPCA managed to produce imputation value with a low difference between the mean of the estimated value and the mean of the actual value; this shows that the method was unbiased. Notably, the result for Tenant B2 shows a higher biased

in comparison with the result for the other tenants. The coverage rate result shows that 90% of the imputed values fall within the confidence interval. Although it was less than 95%, the coverage rate did not fall below 90%, which in that case will be denoted as poor (Demirtas et al., 2008). The R-Squared result shows the estimated value has a good fit for the actual value.

4.3. Data pre-processing

For data pre-processing, it was observed from Fig. 4 that the original dataset contained different scale ranges between power factor with current, voltage and demand. The power factor has a value ranging from 0 (min) until 1 (max). However, the current, voltage and demand all have a higher range of scale. After the standardised transformation, the value managed to be scaled within the same range for all attributes and centralised to achieve a mean of 0 and a standard deviation of 1. This is the same for all tenants.



```
       pf              current          voltage          demand
 Min.   :0.8540   Min.   : 70.55   Min.   :226.7   Min.   : 33.65
 1st Qu.:0.9410   1st Qu.:100.92   1st Qu.:238.9   1st Qu.:104.19
 Median :0.9460   Median :132.22   Median :241.5   Median :129.95
 Mean   :0.9459   Mean   :122.36   Mean   :241.5   Mean   :123.36
 3rd Qu.:0.9510   3rd Qu.:141.83   3rd Qu.:244.2   3rd Qu.:140.35
 Max.   :0.9830   Max.   :160.21   Max.   :249.9   Max.   :158.97

Created from 4666 samples and 4 variables
Pre-processing:
  - centered (4)
  - scaled (4)
       pf               current           voltage           demanc
 Min.   :-12.86809   Min.   :-2.4499   Min.   :-4.422776   Min.   :-4.7
 1st Qu.: -0.68613   1st Qu.:-1.0137   1st Qu.:-0.773300   1st Qu.:-1.(
 Median :  0.01398   Median : 0.4661   Median :-0.009282   Median : 0.3
 Mean   :  0.00000   Mean   : 0.0000   Mean   : 0.000000   Mean   : 0.
 3rd Qu.:  0.71410   3rd Qu.: 0.9206   3rd Qu.: 0.822143   3rd Qu.: 0.
```

Performance evaluation and comparison

Subsequently, after model training and testing, the predictive model generated was compared in terms of performance between algorithms for each tenant. Initially, the result of the testing was observed by comparing the performance of the methods for individual tenants. The comparison table is as shown in Table 5.

Table 5. Performance evaluation of test prediction for all tenants using trained SVM, k-NN and ANN model.

Tenant	Method	RMSE	NRMSE (%)	MAPE (%)
A1	k-NN	5.0025748	4.06	3.02
	SVM	4.7506789	3.85	2.76
	ANN	8.874015	7.19	5.02
A2	k-NN	3.6548885	11.46	9.98
	SVM	3.5898263	11.25	9.38
	ANN	4.540988	14.23	14.16
B1	k-NN	14.934312	23.87	15.43

Tenant	Method	RMSE	NRMSE (%)	MAPE (%)
	SVM	16.0690844	25.69	12.09
	ANN	20.63566	32.99	28.00
B2	k-NN	0.5439403	55.87	48.75
	SVM	0.5558279	57.09	43.97
	ANN	0.547152	56.20	60.62

From the performance evaluation in Table 5, the SVM method utilising Radial Basis Function kernel had the best performance for Tenant A1 and A2, in which the RMSE value was 4.7506789 and 3.5898263, respectively. Even though SVM had the best performance, the difference between RMSE value in comparison to k-NN method was minor. The MAPE result also indicates that SVM prediction had a lower absolute error percentage. For Tenant B1, k-NN was the best performing method in which the RMSE value was comparably smaller than the other method at 14.934312. However, its absolute error was higher than SVM absolute error. This event happened similarly to Tenant B2. The best performing algorithm was k-NN with 0.5439403 RMSE value, which was smaller compared to the RMSE result for SVM and ANN. However, the MAPE result for SVM was lower than the result for k-NN. As the forecast stated was in terms of the expectation of value, the square error method was a much better evaluation method (Tilmann, 2010). Thus, the k-NN method was denoted as the best performing algorithm for Tenant B1 and B2. Referring to the normalised RMSE result, it can be observed how each of the prediction methods performed differently under different datasets. The performance of every method deteriorated from Tenant A1 until Tenant A2, in which Tenant A2 had the worst performance for every method.

Under TNB tariff category, the tenants at both departmental lots A and B were categorised as Medium Voltage General Commercial (Tariff C1). This means that the monthly electricity charges were calculated by acquiring the maximum demand of the month and the kWh (Tenaga Nasional Berhad, 2006). For Tariff C1, the off and on-peak period was not applied to the billing process. Therefore, to predict energy billing thoroughly, the maximum energy demand and the kWh need to be determined. For maximum demand, forecasting an expectation of value can be done to predict the maximum demand of the month. However, for the energy consumption (kWh) prediction, the hourly predicted demand needs to be added up. As the average value shows the characteristics of the whole dataset, the average forecasted consumption was quantified and compared with the actual average consumption. The percentage of difference error between the forecasted and actual value would determine the best forecasting method. The comparison between actual and forecasted average consumption was tabulated and visualized in the form of a line graph, as shown in Fig. 5 until Fig. 8 in the Appendices.

Fig. 5 shows that the ANN forecasted average consumption had a stagnant peakness in which the forecasted average was almost the same for every month. The SVM forecasted average consumption, on the other hand, had better forecasted performance in which the line graph was much closer to the actual average consumption line graph. The k-NN forecast had a slightly larger error in comparison to the SVM forecasted result. Comparison between predicted and actual consumption for Tenant 2 in Fig. 6 shows that both predictions of average consumption made by k-NN and SVM method had better fit to the actual values than ANN method. In November, it can be observed that all methods forecasted a lower consumption for Tenant A2 than the actual consumption. In Fig. 7, the forecasted average consumption by k-NN and SVM was better than ANN. It was also perceived that all methods did not take into consideration the individual high consumption in the month of June. The

performance of all methods in Tenant B1 was approximately similar to the performance of predicted consumption in Tenant B2. Identification of the best method to determine the average consumption on a monthly basis was made by calculating the MAPE value for every month. This calculated value was tabulated in Table 6, whereas Fig. 9 in the Appendices shows a bar graph that compares the percentage of different errors.

Table 6. Mean Absolute Percentage Error (MAPE) of forecasted method for all tenants.

Tenant	k-NN	SVM	ANN
A1	0.40	0.241318507	1.108522675
A2	0.942855477	0.666018364	1.841600488
B1	8.596963137	8.174497001	18.03425668
B2	24.61323638	17.78423714	29.0736946

From Table 6 and Fig. 9, a conclusion can be made in which SVM forecasting is the best method to forecast monthly average consumption. The difference between the percentage of error by SVM method with that of k-NN was not significant. From the visual, the average consumption predicted by ANN method had the highest percentage in every tenant. This denotes that ANN method is inferior in comparison to both SVM and k-NN methods. Comparison of the model training time, as shown in Table 7, was conducted to determine the method with the lowest running time. From the tabulation, the SVM method took the longest time to train while k-NN method was the fastest to finish training. ANN method was faster compared to SVM but still took several hours to complete the training.

Table 7. Comparison of model training running time.

Tenant	k-NN	SVM	ANN
A1	37.916s	18 h 38 m 55.324s	4 h 39 m 30.035s
A2	28.978s	17 h 23 m 32.637s	5 h 14 m 22.311s
B1	34.350s	13 h 20 m 3.722s	5 h 13 m 9.418s
B2	34.5s	12 h 43 m 48.147s	6 h 34 m 48.972s

From the performance evaluation and comparison, SVM method was the best method to predict the individual peak energy demand for department lot A, while k-NN was the preferable choice for department lot B. In terms of average energy consumption, SVM method has proven to be the best method to predict the monthly mean value for energy consumption. However, high training time was required for SVM model training to achieve this high performance. This result further supports the “No Free Lunch” theorem by Wolpert (1996), which was discussed by Stenudd (2010). No Free Lunch stated that many scenarios would determine whether a machine learning method would perform much better than the other. In this research, the scenario would be the dataset distribution and the targeted output (max demand or average consumption). Due to the small difference in performance result between k-NN and SVM methods, it can be concluded that both of these methods were excellent prediction methods. The choosing element would be whether the users want a slightly better accurate result or a faster training method.

4.5. Effect of skewness and kurtosis level on non-parametric classifier

Observing the performance evaluation of each test method in Table 5 shows that Tenant A1, A2, and B1 result was normal as the RMSE value was not unreasonably small and the

absolute error between the forecasted demand to the actual demand was low. On the contrary, Tenant B2 prediction model for each method was considerably poor even by referring the normalised RMSE. From data analysis, it was determined that Tenant B2 dataset was highly skewed, and its kurtosis was leptokurtic, which means presence of high variance in the dataset. Correlating both observations, it can determine that the skewness and kurtosis of a dataset have an effect, especially towards a non-parametric classifier. A non-parametric classifier is an algorithm which falls under probabilistic density supervised classification (Kumar and Sahoo, 2012). This classification model was used if the density function was unknown and does not have a fixed size of parameters. Other than that, the model does not have an initial assumption of the probability density of the dataset (Sampat et al., 2005). During the training and learning process of these algorithms, the parameters increase with the training set. In this research, the model that was used, which are k-Nearest Neighbour, Support Vector Machine with Radial Basis Function kernel, and Artificial Neural Network with Multilayer Perceptron model, is a non-parametric classifier. SVM was initially considered as parametric, but due to the introduction of RBF kernel, the model becomes nonparametric as RBF kernel matrix computes the distance between two pairs of data points (Vasile and Camps, 2013). Skewed and positive/negative kurtosis signifies that the dataset was not normally distributed. Several works of literature described how these properties influence the performance of a non-parametric classifier. Kurtosis level for a dataset has a significant effect on the performance of a nonparametric classifier. This was as indicated by Larasati et al. (2018) in which a leptokurtic type of distribution have a considerably higher chances to be misclassified as it have a higher dispersion of data from the mean value. Another study from the same researcher established the relationship between a skewed dataset and the accuracy of a non-parametric classifier (Larasati et al., 2019). The study concluded that a skewed data does affect the performance of the predictive model, especially the research focussed model, which was artificial neural network.

4.6. Analysis of Microsoft Asure ML studio environment

This research has utilised Microsoft Azure Machine Learning Studio, which is a web service solution for the development of prediction model. Starting from data analysis until performance evaluation, Asure ML has been successfully employed for the implementation of energy demand forecasting. In this section, the employment of Azure ML was analysed in terms of 3 criteria which are: a) Distinguishing Features, II) System Overview, and III) Experimentation Overview. Thoroughly, this analysis would provide an insight on how reliable and capable Azure ML studio for developing a prediction model.

One of the distinguishing features of Azure ML was its ability to manoeuvre through a visualization workflow. The workflow that was conducted inside the environment was manipulated through a graphical drag and drop procedure. Other than that, parsing data for experiment was simply done by joining of modules. Additionally, the platform also supports script packages and algorithms written in external programming language, particularly R programming. In this research, R programming was heavily utilised to conduct prediction modeling such as data pre-processing and model training. The versioning feature of AsureML had helped to reduce the time for experimentation, in which each version of individual experiment information, for instance, parameter settings, was cached.

DATA SET;

Household Electric Power Consumption

Given the economic growth many countries have experienced over the years, the average household has alos been able to afford more and more electrical usage. This notebook will attempt to complete two tasks analytical and predictive to shed light on the driving elements behind electrical consumption. In doing so, we will follow LIME (Local Interpretable Model-agnostic Explanations) principles making it accessible and user-friendly to most readers.

Table of Contents

- Final Model
- AUC - ROC Curve

In [1]:
```
import random
random.seed(123)

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn

from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.svm import SVR
from sklearn.preprocessing import PolynomialFeatures
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
import sklearn.metrics as skm
from sklearn.model_selection import train_test_split
import operator as op

from sklearn.impute import SimpleImputer

import seaborn as sns
sns.set(rc={'figure.figsize': (12,8)})

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/electric-power-consumption-data-set/household_power_consumption.
txt
```

1. Data Importing

In [2]:
```
df = pd.read_csv('/kaggle/input/electric-power-consumption-data-set/household_powe
r_consumption.txt',sep=";")
df.head()
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3147:
DtypeWarning: Columns (2,3,4,5,6,7) have mixed types.Specify dtype option on i
mport or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
0	16/12/2006	17:24:00	4.216	0.418	234.84	18.4	0.0	1.0	17.0
1	16/12/2	17:25	5.360	0.436	233.	23.0	0.0	1.0	16.0

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
	006	:00			63				
2	16/12/2006	17:26:00	5.374	0.498	233.29	23.0	0.0	2.0	17.0
3	16/12/2006	17:27:00	5.388	0.502	233.74	23.0	0.0	1.0	17.0
4	16/12/2006	17:28:00	3.666	0.528	235.68	15.8	0.0	1.0	17.0

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075259 entries, 0 to 2075258
Data columns (total 9 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   Date                   object 
 1   Time                   object 
 2   Global_active_power    object 
 3   Global_reactive_power  object 
 4   Voltage                object 
 5   Global_intensity       object 
 6   Sub_metering_1         object 
 7   Sub_metering_2         object 
 8   Sub_metering_3         float64
dtypes: float64(1), object(8)
memory usage: 142.5+ MB
```

In [4]:
```
# Count the number of null values
df.isnull().sum()
```

Out[4]:
```
Date                         0
Time                         0
Global_active_power          0
Global_reactive_power        0
Voltage                      0
Global_intensity             0
Sub_metering_1               0
Sub_metering_2               0
Sub_metering_3           25979
dtype: int64
```

In [5]:

```
df.isnull().any(axis = 1).sum()
```

Out[5]:
```
25979
```

In [6]:
```
m, n = df.shape
df_per = (df.isnull().sum().sum())/m
col_pers = {}
for i in df.columns:
    col_pers[i] = (df[i].isnull().sum())/m

print(df_per)
print(col_pers)
```

```
0.012518437457686004
{'Date': 0.0, 'Time': 0.0, 'Global_active_power': 0.0, 'Global_reactive_power'
: 0.0, 'Voltage': 0.0, 'Global_intensity': 0.0, 'Sub_metering_1': 0.0, 'Sub_me
tering_2': 0.0, 'Sub_metering_3': 0.012518437457686004}
```

2. Data Pre-Processing

In [7]:
```
# Feature Modification
df['Date'] = df['Date'].astype(str)
df['Time'] = df['Time'].astype(str)
df.replace(['?', 'nan', np.nan], -1, inplace=True)
num_vars= ['Global_active_power', 'Global_reactive_power', 'Voltage',
           'Global_intensity', 'Sub_metering_1', 'Sub_metering_2', 'Sub_metering_3
']
for i in num_vars:
    df[i] = pd.to_numeric(df[i])
imp = SimpleImputer(missing_values=-1, strategy='mean')
df[num_vars] = imp.fit_transform(df[num_vars])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075259 entries, 0 to 2075258
Data columns (total 9 columns):
 #   Column                 Dtype
---  ------                 -----
 0   Date                   object
 1   Time                   object
 2   Global_active_power    float64
 3   Global_reactive_power  float64
 4   Voltage                float64
 5   Global_intensity       float64
 6   Sub_metering_1         float64
 7   Sub_metering_2         float64
 8   Sub_metering_3         float64
dtypes: float64(7), object(2)
memory usage: 142.5+ MB
```

In [8]:
```
# Target Variable
eq1 = (df['Global_active_power']*1000/60)
eq2 = df['Sub_metering_1'] + df['Sub_metering_2'] + df['Sub_metering_3']
df['power_consumption'] = eq1 - eq2
df.head()
```

Out[8]:

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	power_consumption
0	16/12/2006	17:24:00	4.216	0.418	234.84	18.4	0.0	1.0	17.0	52.266667
1	16/12/2006	17:25:00	5.360	0.436	233.63	23.0	0.0	1.0	16.0	72.333333
2	16/12/2006	17:26:00	5.374	0.498	233.29	23.0	0.0	2.0	17.0	70.566667
3	16/12/2006	17:27:00	5.388	0.502	233.74	23.0	0.0	1.0	17.0	71.800000
4	16/12/2006	17:28:00	3.666	0.528	235.68	15.8	0.0	1.0	17.0	43.100000

Exploratory Data Analysis

In [9]:
```
# Distribution of the target variables
sns.histplot(data=df, x='power_consumption', bins=15, kde=True)
plt.show()
```

In [10]:
```
corr = np.corrcoef(df.corr())
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(df.corr(), annot=True, mask=mask)
plt.show()
```

Model Building & Evaluation

In [11]:
```
models = {}
df1 = df
df1.head()
```

Out[11]:

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	power_consumption

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	power_consumption
0	16/12/2006	17:24:00	4.216	0.418	234.84	18.4	0.0	1.0	17.0	52.266667
1	16/12/2006	17:25:00	5.360	0.436	233.63	23.0	0.0	1.0	16.0	72.333333
2	16/12/2006	17:26:00	5.374	0.498	233.29	23.0	0.0	2.0	17.0	70.566667
3	16/12/2006	17:27:00	5.388	0.502	233.74	23.0	0.0	1.0	17.0	71.800000
4	16/12/2006	17:28:00	3.666	0.528	235.68	15.8	0.0	1.0	17.0	43.100000

1. Multiple Linear Regression

In [12]:

```
class linmodel():
    def __init__(self, df, target):
        self.df = df
        self.target = target

    def pre_processing(self):
        cat = ['Date', 'Time', 'power_consumption']
        X = self.df.drop(cat+[self.target], axis=1).values
        Y = self.df[self.target].values

        self.X_train, self.X_test, self.Y_train, self.Y_test = train_test_split(X,
Y,
                                                                test_size = 0.2,
                                                                random_state = 2)
        return self

    def fit_pred_acc(self):
        reg = LinearRegression()
        reg.fit(self.X_train, self.Y_train)
        pred = reg.predict(self.X_test)
        mae = round(skm.mean_absolute_error(self.Y_test, pred), 2)
        rmse = round(skm.mean_squared_error(self.Y_test, pred, squared=False), 2)
        r2_score = round(skm.r2_score(self.Y_test, pred), 4)
        ev = round(skm.explained_variance_score(self.Y_test, pred), 4)

        return [mae, rmse, r2_score, ev]
```

```
lin = linmodel(df, 'Global_active_power')
lin = lin.pre_processing()
models["Mult. Reg"] = lin.fit_pred_acc()
```

2. Shrinkage Technique: Ridge

```
In [13]:
class rdgmodel():
    def __init__(self, df, target):
        self.df = df
        self.target = target

    def pre_processing(self):
        cat = ['Date', 'Time', 'power_consumption']
        X = self.df.drop(cat+[self.target], axis=1).values
        Y = self.df[self.target].values

        self.X_train, self.X_test, self.Y_train, self.Y_test = train_test_split(X,
Y,
                                                                 test_size = 0.2,
                                                                 random_state = 2)
        return self

    def fit_pred_acc(self):
        reg = Ridge(alpha=0.0001, normalize=True)
        reg.fit(self.X_train, self.Y_train)
        pred = reg.predict(self.X_test)
        mae = round(skm.mean_absolute_error(self.Y_test, pred), 2)
        rmse = round(skm.mean_squared_error(self.Y_test, pred, squared=False), 2)
        r2_score = round(skm.r2_score(self.Y_test, pred), 4)
        ev = round(skm.explained_variance_score(self.Y_test, pred), 4)

        return [mae, rmse, r2_score, ev]

rdg = rdgmodel(df, 'Global_active_power')
rdg = rdg.pre_processing()
models["Ridge Reg"] = rdg.fit_pred_acc()
```

3. Shrinkage Technique: Lasso

```
In [14]:
class lasmodel():
    def __init__(self, df, target):
        self.df = df
        self.target = target

    def pre_processing(self):
        cat = ['Date', 'Time', 'power_consumption']
        X = self.df.drop(cat+[self.target], axis=1).values
        Y = self.df[self.target].values

        self.X_train, self.X_test, self.Y_train, self.Y_test = train_test_split(X,
Y,
                                                                 test_size = 0.3,
                                                                 random_state = 72)
        return self

    def fit_pred_acc(self):
        reg = Lasso()
```

```
        reg.fit(self.X_train, self.Y_train)
        pred = reg.predict(self.X_test)
        mae = round(skm.mean_absolute_error(self.Y_test, pred), 2)
        rmse = round(skm.mean_squared_error(self.Y_test, pred, squared=False), 2)
        r2_score = round(skm.r2_score(self.Y_test, pred), 4)
        ev = round(skm.explained_variance_score(self.Y_test, pred), 4)

        return [mae, rmse, r2_score, ev]

las = lasmodel(df, 'Global_active_power')
las = las.pre_processing()
models["Lasso Reg"] = las.fit_pred_acc()
```

4. Polynomial Multiple Regression

```
In [15]:
class polymodel():
    def __init__(self, df, target):
        self.df = df
        self.target = target

    def pre_processing(self):
        cat = ['Date', 'Time', 'power_consumption']
        X = self.df.drop(cat+[self.target], axis=1).values
        Y = self.df[self.target].values

        self.X_train, self.X_test, self.Y_train, self.Y_test = train_test_split(X,
Y,
                                                            test_size = 0.2,
                                                            random_state = 42)
        return self

    def fit_pred_acc(self):
        reg = LinearRegression(normalize=True)
        pol_feat = PolynomialFeatures(2)
        X_train_transf = pol_feat.fit_transform(self.X_train)
        X_test_transf = pol_feat.fit_transform(self.X_test)
        model = reg.fit(X_train_transf, self.Y_train)
        pred = model.predict(X_test_transf)
        mae = round(skm.mean_absolute_error(self.Y_test, pred), 2)
        rmse = round(skm.mean_squared_error(self.Y_test, pred, squared=False), 2)
        r2_score = round(skm.r2_score(self.Y_test, pred), 4)
        ev = round(skm.explained_variance_score(self.Y_test, pred), 4)

        return [mae, rmse, r2_score, ev]

poly = polymodel(df, 'Global_active_power')
poly = poly.pre_processing()
models["Poly Reg"] = poly.fit_pred_acc()
```

```
In [16]:
models_df = pd.DataFrame.from_dict(models, orient='index',
                    columns=['MAE', 'RMSE', 'R_sq', 'Expl. Var.'])
models_df
```

Out[16]:

	MAE	RMSE	R_sq	Expl. Var.

	MAE	RMSE	R_sq	Expl. Var.
Mult. Reg	0.03	0.04	0.9985	0.9985
Ridge Reg	0.03	0.04	0.9985	0.9985
Lasso Reg	0.17	0.23	0.9509	0.9509
Poly Reg	0.02	0.04	0.9988	0.9988