Analytics and Forecasting Energy Consumption Using Smart Meters

Team 014: Transatlantic Synergy
Andre Klein, Jenea Spinks, Tyler Nguyen, Austin Ikedi Okechukwu

INTRODUCTION

Using the ACORN smart meter dataset, the project's goal is to analyze London's energy consumption patterns, and the results will be visualized on an interactive website. To do this, clustering techniques like KMeans will be used to identify salient features (such as temperature, pressure, UV index, etc.) in our dataset. Our second objective is to predict utilizing comparative time series analysis techniques (ARIMA, SARIMA, and STL) based on prominent characteristics found through clustering.

PROBLEM DEFINITION & LITERATURE SURVEY

The government would like to study and understand the energy consumption of every household in England, Wales, and Scotland by installing smart meters. In London, 5567 households volunteered to install smart meters to gather energy consumption data. From previous studies, the raw data had been reprocessed and clustering analyses (e.g., KMeans) had been completed to categorize consumers and reveal consumption patterns [1], [2], [3]. Neural network models, particularly RNNs, excel in load forecasting, as shown in a study of 920 Irish customers [4] and in the study of traditional "forward" modeling and "inverse" modeling [5]. Other studies suggest that expert systems using SVR, neural networks, and fuzzy sets outperform decision trees and linear regression in power load forecasting [6]. However, current practices face limitations, including the need for meaningful feature extraction before clustering [7] and challenges with scalability due to the computational cost of techniques like KMeans [8]. Furthermore, load forecasting accuracy is hindered by factors like volatility and irregular fluctuations in household loads, alongside varying impacts from weather conditions, weekdays, holidays, and other factors [9]. Finally, STL methods applied here "can benefit statistical methods by providing a more robust decomposition procedure." [16]

PROPOSED METHOD

KMeans analysis is used to examine datasets. It does this by detecting which ACORN groups belong together by clustering comparable data points. KMeans can help identify distinct customer categories by grouping the smart meters according to demographics, patterns of power use, or other pertinent criteria. Personalized advice, customized pricing plans, and targeted marketing campaigns may all be implemented with this segmentation. Additionally, utility companies can forecast peak demand times, operate their energy distribution networks more effectively, and allocate resources more efficiently by comprehending various load profiles depending on the clusters. After analyzing these clusters, energy companies and legislators may pinpoint high-energy consumption locations and carry out focused actions to encourage energy conservation and efficiency. Our use of time-series analysis techniques such as ARIMA will enable us to provide a distinct insight since ARIMA models can capture both short-term fluctuations and long-term trends in time series data [11], [12]. Data on electricity consumption frequently shows both general trends over time (such as rising usage because of population increase or economic development) and distinct seasonal patterns (such as higher usage in the summer owing to air conditioning). These trends may be automatically considered by ARIMA models, which can then modify their projections appropriately. Customers and energy firms can connect power consumption to home activities by using ARIMA's forecasting of consumption trends and patterns. Users will be able to interact with and study the forecast data trend, which will be plotted on a separate page in accordance with the original dataset. This will allow utilities, energy providers, and policymakers to plan and optimize resource allocation, infrastructure investment, and energy management strategies. We chose to use ARIMA as the method for forecasting because it is the model that is built specifically for time series data, which is what our chosen dataset is.

Another benefit of this method is that the runtime does not scale exponentially with the number of features compared to distance-based machine learning techniques such as KNN and SVM; a distinctive advantage of our technique over other common practices. Initially, we attempted to run with non-time series methods such as Decision Trees and KNN and our accuracy was quite low, which meant that our data is sensitive to time. For KMeans, however, we chose to perform the analysis utilizing it because we would like to understand the dataset features. In the final phase, we went further beyond basic ARIMA and utilized seasonal trends in the forecasting using SARIMA and STL-ARIMA.

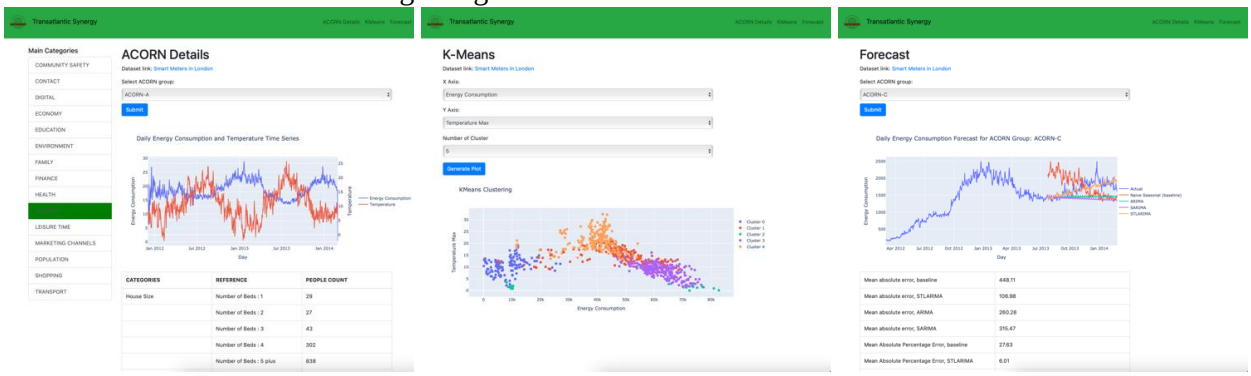

Figure 1: The Website with Different Pages Dedicated to the Analysis Have Done for the Project

A website that allows user interaction when showing the results of the analyses that we have done using different visualization tools such as Flask, Pandas, and Plotly [10]. The first page, ACORN Details, shows the original dataset for each ACORN group based on user selection from the dropdown menu. It displays the time series data of energy consumption along group characteristics such as population, economy, etc., called Main Categories, which a user can select as desired. Furthermore, it is useful to be able to view trendlines for comparisons of consumption involving seasons/days. It will be successful because it gives utility providers and customers detailed insight into peak times and consumption patterns, empowering them to make wise decisions about how much energy they use and spot areas where they can cut back. The second page, KMeans, displays the clustering plots based on our findings. Users can choose their own combinations of different options with the number of clusters to see how the data is grouped up, hence understand the relationships between different components. Lastly, the Forecast page shows the forecasting results of a chosen ACORN group from our time series analysis and the error metrics to show how accurate our analysis is. All plots can be zoomed in/out, moved around, and let the user select a specific section on the plot.

EXPERIMENTS/EVALUATION

Data Cleaning and Goals

We used Pandas to merge our daily_dataset per block into one data frame then merged that data frame with csv files such as information per household, details per ACORN, holidays, weather, etc. When we merged with weather data, we made sure to use the time the reading was taken instead of the day the reading was allocated to since the time of the reading is more indicative. We decided that our resolution would be daily instead of hourly and accordingly parsed date information. During the data frame merge process, we noticed there were days where there were no weather measurements; we discarded those days. Similarly, we discarded days with weather readings but no meter readings. Looking at the figure below we can see that meters were incrementally installed between Q4 2011 and Q3 2012. We discarded a few months that did not have sufficient readings and we started the dataset on March 1st, 2012. We could have started later but wanted to make sure that each month is represented twice. In addition, we discarded days that had less than 48 readings since ideally, we should have 48 readings per day (normally readings are taken every

30 min). For holidays, we aggregated days as either having holidays or not (binary representation). For ACORN groups, we eliminated uncategorized areas and kept affluent, comfortable, and adversity ACORN groups and converted to numeric representations. For blocks, we also converted string to numeric representations. We did the same for weather patterns until everything in the data frame was numeric other than date which was used to index on. We aggregated all readings per day in the previous phase. In this phase, we aggregated by block so that we have a better insight and finer resolution in our experiment. Previously, we had implemented ARIMA successfully. In this phase, we noticed that our data possesses seasonal patterns as shown in the STL decomposition below (setting season length = 30 steps/days). As can be seen, there are seasonal patterns that are clearly not mere noise. In general, the benefit of a technique such as SARIMA or STL-ARIMA is that ARIMA handles the residue while the seasonal parts of the algorithm handle the general trend. Given the aforementioned discovery involving seasonality, we wanted to identify the most suitable seasonal technique based on our dataset.

We utilized KMeans clustering to potentially identify distinct patterns and groups of similar data points. By clustering the data, we aimed to discover underlying structures and potentially uncover correlations or dependencies between weather conditions and energy usage which can be useful in segmenting the data to produce more useful and tailored forecasts

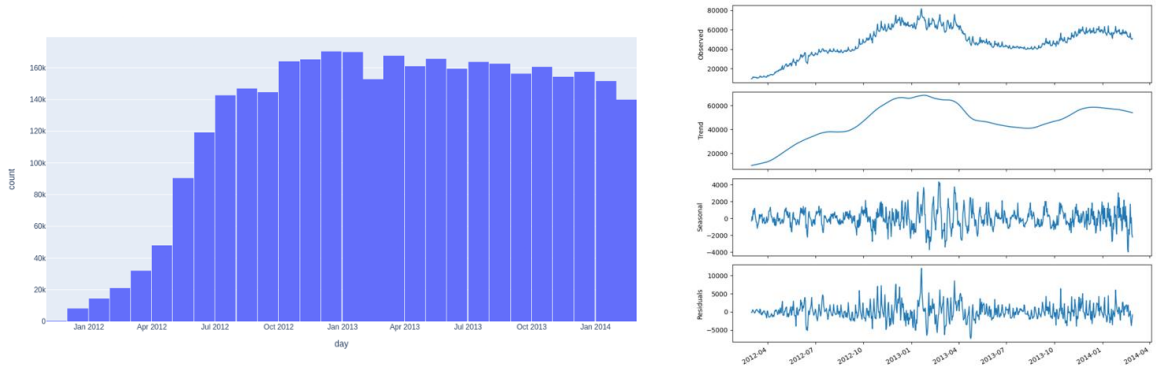

Figure 2: On the left-hand side, we have Total Data Count for each month to determine the sufficient reading period. On the right-hand side, we have STL-Decomposition to examine seasonal trends.

Clustering

Before clustering, we addressed the presence of both continuous and categorical attributes in our dataset. Firstly, we identified the number of categories within each variable, and subsequently applied an encoding technique based on this information. The categorical variables chosen for our analysis were dual category features and were transformed into a binary format. Furthermore, given that our dataset contained features with varying scales, we recognized the need for feature scaling to prevent any individual feature from dominating the clustering process. To achieve this, we employed the StandardScaler, which normalized the features. During our data exploration phase, which involved correlation analysis and statistical visualizations like boxplots, we observed highly correlated features. Consequently, we eliminated such features from consideration. For instance, we discovered that the maximum temperature and apparent maximum temperature features exhibited nearly full correlation, leading us to select only one of them for our analysis. By addressing these preprocessing steps and identifying and removing redundant features, we aimed to ensure the validity and effectiveness of our KMeans clustering analysis. After preprocessing the data, we proceeded to apply KMeans clustering to derive meaningful insights. During the experimentation phase, we examined a range of cluster quantities, spanning from two to fifteen. To determine the optimal number of clusters, we assessed the silhouette score for each configuration and selected the one with the

highest score. Once the optimal cluster count was identified, we conducted KMeans clustering on both the complete dataset and a dataset reduced through Principal Component Analysis (PCA). This allowed us to conduct a comparative analysis between the two approaches. To determine the appropriate number of principal components for the PCA reduction, we plotted a scree plot. This plot illustrates the explained variance associated with each principal component. We selected the number of principal components at the point where the plot displayed a leveling off, indicating the point of diminishing returns. For the PCA-reduced dataset, we chose the first five principal components. The clustering results are shown below:

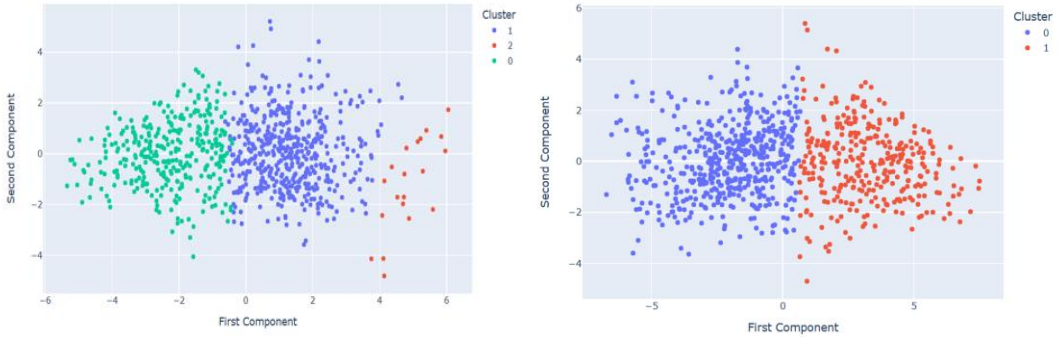

Figure 3: Clustering Results on Non-reduced Dataset (Left) And Pca-reduced Dataset (Right)

When conducting clustering on the non-reduced dataset, we determined that the optimal number of clusters was three, resulting in a silhouette score of 0.258. However, after reducing the dimensionality of the data, the optimal number of clusters decreased to two, accompanied by an improved silhouette score of 0.319. This increase in the silhouette score indicates enhanced compactness within the clusters and suggests slightly improved clustering performance. We conducted a validation and comparison of our KMeans clustering outcomes with those obtained from the KPrototypes model. In our analysis, we utilized the same range of clusters (2 to 15) and employed identical features. However, rather than encoding the categorical variables, we incorporated them into the KPrototypes model in their original form given its ability to handle a mixture of datatypes. The dimensionally reduced dataset obtained through PCA yielded the best clustering results with 2 clusters. One cluster contained higher temperatures, lower humidity, more cloud cover, lower UV Index datapoints and the other, the opposite highlighting seasonality. Notably, the clusters closely resemble the outcomes of the KPrototypes model which had a score of 0.29. The similarity in silhouette scores suggests that both KMeans and KPrototypes exhibit comparable performance, albeit with distinct methodologies for handling variable types.

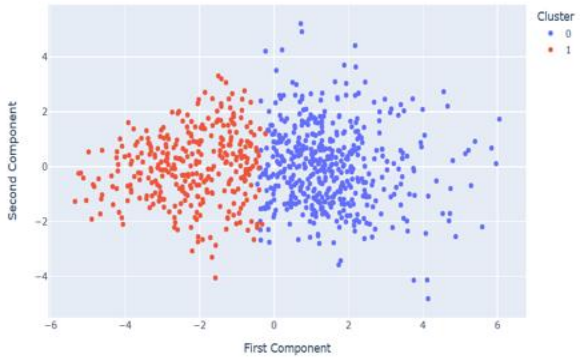

Figure 4: K-Prototypes Clustering Results

Autoregressive Integrated Moving Average (ARIMA)

The first thing we did to implement ARIMA is to identify the integrative term d (also referred to as i). The d term can be found using ADF (Augmented Dickey-Fuller test) statistic which is a measure of stationarity; we differentiate until we get stationary data with high confidence. In discrete data, this translates to differencing (np.diff). Without differencing, ADF statistic was –2.5 but had a p-value of 0.104 which indicates low confidence in stationarity. After differencing once, ADF statistic was –5.04 and the p-value was 1.84×10^{-5} which indicates high confidence in stationarity. Therefore, our d term is 1. What is left is identifying the terms for auto-regression and moving average (AR & MA terms p and q). We experimented with different p and q values to find whichever one gives the lowest AIC (Akaike information criterion) score which indicates how well the terms minimize the error. The lowest AIC score resulted from p and q equal to 7. Next, we performed Lijung-box test on the residuals to see if our residuals have any significant autocorrelation or if they are purely stochastic (follow a normal distribution). See figure below for test results. As you can see, standard residuals look like pure noise at p and q set to 7. The histogram plus estimated density follows a normal distribution. Looking at the quantiles as well, the data is close to the red line indicating it is following a normal distribution. Finally, our correlogram shows values within the blue boundary for all x values greater than 0 which indicates that there is nothing statistically significant extending beyond normal noise. After the implementation of ARIMA, we noticed that ARIMA is not capturing seasonal trends clearly. As can be seen in the figure below, the black plot (ARIMA) does not possess any significant trend that follows the general trend of the graph. Therefore, the next step was to implement SARIMA to account for seasonality. We attempted different lengths of seasons for SARIMA. However, considering the size of our dataset, anything beyond s (or m) equal to 12 was computationally impossible given our resources. Here, s is the length of the season (i.e., number of days/steps in a season). We had to limit s to 7 which affected our ability to capture trends pertaining to weather patterns. An additional challenge is that our dataset only captures two years of data which makes it harder to capture seasonality pertaining to weather seasons. When tuning SARIMA, we had to limit ourselves to a smaller grid search since the number of parameters scales with P and Q. We settled on P and Q equal to 1. P and Q are also AR and MA terms (see above), but they pertain to general trends vs short-term trends in ARIMA. Unfortunately, SARIMA did not entail a much better prediction due to the aforesaid challenges. We decided to explore things further and were able to identify a tool in statsmodels called STLForecast. STLForecast is a hybrid model of Seasonal-Trend Decomposition (ST) using Locally Estimated Scatterplot Smoothing (L) and ARIMA. The way it works is by decomposing our data into seasonal trends and residuals. It operates using STL on the seasonal part while using ARIMA to operate on the residuals. While STLForecast was much faster than SARIMA, there was still a challenge of identifying a sufficient length of a season (period parameter). We settled on 30 to capture monthly patterns. Unfortunately, we do not have enough years in the dataset to capture annual patterns. Our performance was still satisfactory (see figure below). Finally, we implemented a baseline that is naive-seasonal (repeating last portion of our data) to compare with ARIMA, SARIMA, and STL-ARIMA. Forecasting on the aggregation of blocks, naive-seasonal mean absolute error was 16082.5, ARIMA was 9265.0, SARIMA was 16766.6, and STL-ARIMA was 4033.8. STL-ARIMA performed much better on individual blocks/ACORNs too. In figure 1 above, STL-ARIMA (orange) followed the actual data trend much better than SARIMA (purple). For most blocks, STL-ARIMA had much lower MAE than other algorithms. For example, in ACORN-C naive-seasonal was 448.1, STLARIMA was 107.0, ARIMA was 260.3, and SARIMA was 315.5.

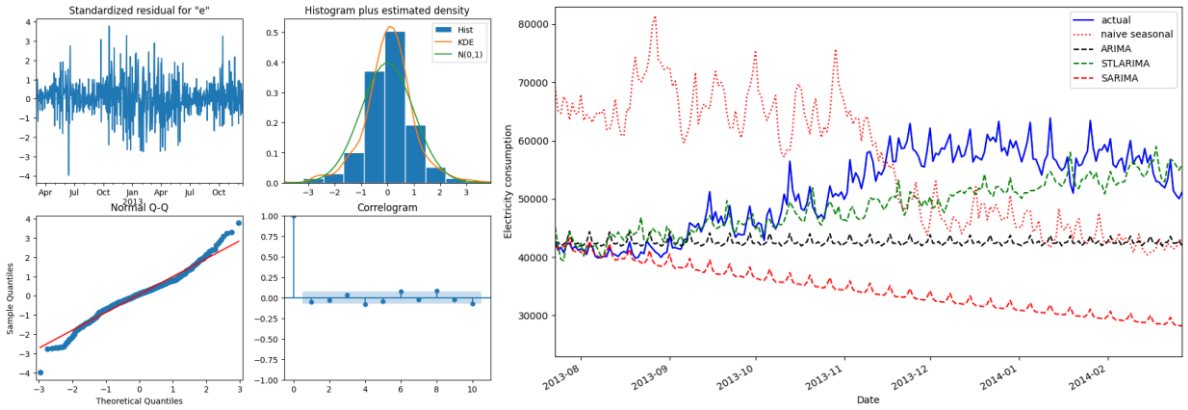

Figure 5: On the left-hand side, we can see results of our Lijung-box test. On the right-hand side, we can see forecast on aggregates using ARIMA, SARIMA, Naive Seasonal (baseline), and STL-Forecast.

Conclusions and Discussion

This project was the first time we had the opportunity to work on time series and had a significant pedagogical outcome. We learned about how time series is impacted by big data and the limitations of applying statistical methods on time series. The KMeans clustering method yielded satisfactory results by effectively segregating the data, although it necessitated the encoding of categorical features. To address this limitation, we employed the K-Prototypes algorithm, which handled mixed data types (categorical and continuous) directly, eliminating the need for data encoding. This alternative approach yielded comparable outcomes to the KMeans model with the best being two clusters for each model. However, the presence of fluctuations and temporal dependencies within the data can affect the clustering results. As a result, it would be interesting to see the effect of segmenting the data by seasons and performing clustering on each subset individually. Additionally, an alternative technique such as a variant of time series k-means with the dynamic time warping metric could have been explored for clustering household consumption data, to consider the data's temporal nature. For forecasting, we learned that ARIMA is useful for short-term predictions that do not involve seasonal trends. Additionally, SARIMA has limitations when it comes to longer seasons and tuning can become computationally expensive. We ran into issues with convergence with some ACORNS even at s = 12 despite many iterations. SARIMA could have been more useful if we did monthly instead of daily aggregation but that would be a coarse resolution for our purposes. Lastly, we learned about a useful tool called STLForecast a hybrid model of STL decomposition and ARIMA. STLForecast is fast and powerful but is also data hungry. If we were to choose a season length of one year, there is simply not enough duration in our dataset to satisfy that and we had to settle for a season length of one month, which still produced satisfactory results. Given that a substantial amount of time was spent tuning and experimenting with our time series models (a week on a powerful machine), we did not get a chance to experiment with deep learning. This would be an exciting thing to try in the future for this dataset. We did attempt some ML models such as KNN and decision trees but those produced poor results. All team members have contributed a similar amount of effort to this project. To sum up, STLForecast was the fastest and most appropriate approach for our data. This is evident by MAE calculations for each ACORN. Since energy consumption correlates with weather, ideally season length would have been set to one year.

CITATIONS

[1] Okereke, G. E., Bali, M. C., Okwueze, C. N., Ukekwe, E. C., Echezona, S. C., & Ugwu, C. I. (2023). KMeans clustering of electricity consumers using time-domain features from smart meter data. *Journal of Electrical Systems and Information Technology*, *10*(1), 2–18. https://doi.org/10.1186/s43067-023-00068-3

[2] Haben, S., Singleton, C., & Grindrod, P. (2016). Analysis and Clustering of Residential Customers Energy Behavioral Demand Using Smart Meter Data. *IEEE Transactions on Smart Grid*, 7(1), 136–144. - https://doi.org/10.1109/TSG.2015.2409786

[3] Alquthami, T., AlAmoudi, A., Alsubaie, A. M., Jaber, A. B., Alshlwan, N., Anwar, M., & Al Husaien, S. (2020). Analytics framework for optimal smart meters data processing. *Electrical Engineering*, 102(3), 1241–1251. https://doi.org/10.1007/s00202-020-00949-0

[4] Cheng, L., Zang, H., Xu, Y., Wei, Z., & Sun, G. (2021). Probabilistic Residential Load Forecasting Based on Micrometeorological Data and Customer Consumption Pattern. *IEEE Transactions on Power Systems*, *36*(4), 3762–3775. https://doi.org/10.1109/TPWRS.2021.3051684

[5] Edwards, R. E., New, J., & Parker, L. E. (2012). Predicting future hourly residential electrical consumption: A machine learning case study. *Energy and Buildings*, 49, 591–603. https://doi.org/10.1016/j.enbuild.2012.03.010

[6] Jahan, I. S., Snasel, V., & Misak, S. (2020). Intelligent Systems for Power Load Forecasting: A Study Review. *Energies (Basel)*, 13(22), 6105-. https://doi.org/10.3390/en13226105

Wang, Y., Chen, Q., Hong, T., & Kang, C. (2019).

[7] Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges. *IEEE Transactions on Smart Grid*, *10*(3), 3125–3148. https://doi.org/10.1109/TSG.2018.2818167

[8] Anand, H., Nateghi, R., & Alemazkoor, N. (2023). Bottom-up forecasting: Applications and limitations in load forecasting using smart-meter data. *Data-Centric Engineering (Online)*, 4. https://doi.org/10.1017/dce.2023.10

[9] Azeem, A., Ismail, I., Jameel, S. M., Romlie, F., Danyaro, K. U., & Shukla, S. (2022). Deterioration of Electrical Load Forecasting Models in a Smart Grid Environment. *Sensors (Basel, Switzerland)*, 22(12), 4363-. https://doi.org/10.3390/s22124363

[10] Hoque, E., & Agrawala, M. (2020). Searching the Visual Style and Structure of D3 Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, *26*(1), 1236–1245. https://doi.org/10.1109/TVCG.2019.2934431

[11] Géron, Aurélien. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.)*. O'Reilly Media, Inc.

[12] Gridin, I. (2021). *Time Series Forecasting using Deep Learning: Combining PyTorch, RNN, TCN, and Deep Neural Network Models to Provide Production-Ready Prediction Solutions / (1st ed.)*. BPB Publications.

[13] Koroleva, K., Melenhorst, M., Novak, J., Herrera Gonzalez, S. L., Fraternali, P., & Rizzoli, A. E. (2019). Designing an integrated socio-technical behavior change system for energy saving. *Energy Informatics*, 2(Suppl 1), 1–20. https://doi.org/10.1186/s42162-019-0088-9

[14] Bradley, P., Coke, A., & Leach, M. (2016). Financial incentive approaches for reducing peak electricity demand, experience from pilot trials with a UK energy provider. Energy Policy, 98, 108–120. https://doi.org/10.1016/j.enpol.2016.07.022

[15] Arora, S., & Taylor, J. W. (2016). Forecasting electricity smart meter data using conditional kernel density estimation. *Omega (Oxford)*, 59, 47–59. https://doi.org/10.1016/j.omega.2014.08.008

[16] Jabloun, M., Ouyang, Z., & Ravier, P. (2021). STL Decomposition of Time Series Can Benefit Forecasting Done by Statistical Methods but Not by Machine Learning Ones. Engineering Proceedings, 5(1). https://doi.org/10.3390/engproc2021005042