Dark Sky: Using Smart Meter Data to Reclaim London's Night

Marie Aimee Karumuhinzi, Nicole Hessner, Alessandro Loi, Tyler Adams
DS625 – Big Data Architectures/Systems,
Master of Science Computer Science, Master of Science Data Science
School of Technology & Computing, City University of Seattle

karumuhinzimarieaim@cityuniversity.edu, adamstyler@cityuniversity.edu, hessnernicole@cityuniversity.edu, loialessandro@cityuniversity.edu

Abstract

Minimizing exposure to light during nighttime has many known health and ecological benefits. Using Big Data and Machine Learning tools and techniques in Apache Spark to analyze Smart Meter and Dark Sky data from London, England, we will determine if there are any actionable correlations between Dark Sky hours and energy usage. This will require correcting for major electricity outage events and weather patterns, including both cloud cover and temperature, and comparing against sunrise and sunset times.

Keywords: Machine Learning, Energy Management, Smart Meters, Apache Spark, Dark Sky, Big Data

1. INTRODUCTION

The importance of preserving natural darkness and minimizing artificial light during nighttime has gained significant attention in recent years due to its profound implications for human health and the ecological balance of our planet. Artificial light at night, also known as light pollution, has been linked to various adverse effects on human well-being, including disruptions in sleep patterns, increased risk of certain diseases, and ecological disturbances. Recognizing the significance of these issues, this research paper investigates the relationship between Dark Sky hours, energy usage, and weather patterns in the context of London, England, with the aim of reclaiming and promoting the restoration of natural darkness.

To accomplish this objective, we leverage the potential of Big Data and Machine Learning tools, specifically utilizing Apache Spark, to analyze Smart Meter and Dark Sky data. The Smart Meter data, sourced from the Smart Meters in London dataset, provides real-time information on energy consumption trends in the city, while Dark Sky data offers insights into the extent of natural darkness. By examining the correlations between Dark Sky hours, energy usage, and weather conditions, we can uncover actionable insights that can inform awareness campaigns and educational initiatives to enhance Dark Sky preservation efforts in large metropolitan areas like London.

The dataset used in this study involves the integration and preprocessing of various datasets, including daily and hourly energy usage, UK bank holidays, Acorn demographic classification, and weather data. However, we will be focusing on the subset of the dataset focused on energy and weather, which includes nighttime visibility. To handle the substantial volume of data, we employ Apache Spark, a distributed data processing framework capable of parallelizing computations. We will employ PySpark Machine Learning's LinearRegression to determine if there are significant patterns between weather features and half-hourly Dark Sky hours, identifying the weather variables that exhibit the strongest correlation with Dark Sky conditions.

The outcomes of this research hold significant practical implications. First and foremost, the study aims to provide insights to electricity providers, enabling them to anticipate and

prepare for potential high energy usage periods based on correlations between Dark Sky hours and energy demand. Additionally, the integration of smart meters, along with advanced data analytics and forecasting techniques, can contribute to improved energy efficiency, strengthening the reliability and sustainability of existing infrastructure. Furthermore, the centralized availability of smart meter data facilitates the discovery of previously unrealized correlations and empowers customers to implement more efficient building automation strategies based on data-driven insights.

This paper contributes to the growing body of research that underscores the value of real-time data analytics and machine learning algorithms in optimizing energy consumption and enhancing the effectiveness of public infrastructure. By leveraging the power of Apache Spark and distributed data processing architectures, we aim to harness the potential of smart meter data to achieve greater energy efficiency, conservation of natural resources, and the preservation of Dark Sky hours in urban environments.

2. LITERATURE REVIEW

Research continues to outline a large potential for increasing energy efficiency and forecast accuracy by using real-time data with Machine Learning or Deep learning techniques. One article titled Accurate prediction of hourly energy consumption (Hoai et al, 2021), aimed to use real-time weather data to make forecasts using a deep neural network. Such predictions were imperative due to the microgrid electrical infrastructure adopted in Victoria, Australia where this study was based. Distributing electrical demand across multiple microgrids allowed for both redundancy and independent levels of energy storage depending on the area. The infrastructure described above is commonly controlled by an energy management system (EMS) that makes automatic, real-time decisions depending on energy demand and predictions. This is just one example of how distributed data systems such as Apache Spark can make data more accessible to these systems that control critical infrastructure. The need for building-level forecasting is highlighted in the article "…studies are carried out to evolve the current machine learning techniques to learn the uncertainly at the building level directly due to the many influencing factors" (Hoai et al, 2021). Utilization of smart meters would be required to provide the level of details that these EMS systems require to make their decisions. Not only will utilizing real-time data and machine learning algorithms help reduce energy consumption, but it would also strengthen the reliability of existing infrastructure.

Apache Spark and other distributed data systems dramatically improve benchmark results for big data energy forecasting. Another article titled Concept and benchmark results for big data energy forecasting based on Apache spark (Ordiano, 2018), aims to show the benefits of these distributed systems using a simulated energy dataset representing a smart-grid infrastructure. Their benchmarks used a spark cluster versus the R programming language, which was ran on a single computer. Various amounts of data were used in the experiment including 1 day, 1 week, 1 month, 6 months, 1 year, 5 years and 10 years of energy data, all containing entries related to energy consumptions in one-second intervals. Benchmark results showed that R performed well with smaller amounts of data, however when data equal to or larger than 6 months was utilized, spark clusters outperformed R in both training and evaluation of the forecasting model (Ordiano, 2018). After 5 years of data, the machine running R was unable to complete the tests due to memory limitations. The shear amount of data required to monitor and make forecasts is extensive and would be far beyond the capabilities of one machine. Looking at the current trends related to the subject of data collection and forecasting, it is clear that utilization of distributed systems such as Spark, along with predictive algorithms have the potential to greatly improve current energy systems.

The article (Sarswatula et al., 2022) aimed to provide applications of statistics and machine learning using data collected by the United States Department of Energy Industrial Assessment Centers program in metal, electrical, food and plastic manufacturing to draw insights from energy data. These companies are among the major energy consuming industries in the US and understanding this data is vital to provide recommendations for improving energy efficiency. The dataset was composed of over 15,000 samples collected from 1981 to 2013. An exploratory data analysis was performed to clean the data, visualize trends, and draw insights. Energy consumption predictive models were built using machine learning techniques such as Multiple Liner Regression, Random Forest Regressor, Decision Tree Regressor and Extreme Gradient Boost Regressor. In addition,

classification models using Support Vector Machines, Random Forest, K-Nearest Neighbors and Deep Learning were also considered. The results indicated that Random Forest Regressor is the best technique with an R^2 of 0.869, deep learning performed competitively with an accuracy of .88 in training and testing after 10 epochs, and Random Forest Classifier was the best technique with precision, recall, F1 score and accuracy score of 0.818, 0.884, 0.844 and 0.883, respectively. These results indicate that machine learning is capable of predicting energy consumption and identifying opportunities for improving energy efficiency.

De Mattos Neto et al. (2021) studied the usage of Extreme Learning Machine Ensemble to forecast smart meters energy consumption. Using smart meters in monitoring energy consumption is essential for planning and managing power generation systems. Being able to forecast energy consumption is important for decision making as it can predict forthcoming demands to energy providers. Since energy consumption time series may have linear and non-linear patterns, models may not be able to achieve the best accuracy prediction. Thus, using ensemble learning technique stood out due to its high capability to increase prediction power of stand-alone forecasting models. The proposed neural based ensembles for energy consumption forecasting were conducted using a series from residential building containing installed smart grid network. The results indicated that extreme learning machine based on ensemble outperformed other proposals. The model was simple, fast learner and had great ability of generalization in comparison to other artificial neural networks.

3. METHODOLOGY

The dataset that we are using is the Dark Sky dataset from Kaggle. This dataset consists of several different collections of data: daily energy usage, hourly energy usage, UK bank holidays, hourly dark sky data, daily dark sky data, Acorn demographic classification data, half hourly and hourly weather data, and customer information data. Since this dataset contains several gigabytes of data, we will utilize a Spark Session to make use of parallelization. First, the datasets will be prepared by removing all of the UK bank holiday dates from the datasets, since energy usage patterns will likely be different than the typical usage patterns, and culturally significant holidays will likely continue to be celebrated similarly in the future, irrespective of public attitudes toward energy conservation and dark sky hours. Then, we will use PySpark Machine Learning to look for significant patterns between weather features and half-hourly dark sky hours and determine which weather variables have the strongest correlation with dark sky, if any. After determining what weather variables impact dark sky, we can normalize the energy data against those variables using an appropriate regression and determine if there are any statistically significant correlations between energy usage and dark sky hours.

To pre-process the data, rows with null values were removed, and a Date column was added to the daily energy and hourly weather datasets, as shown in Figure 1. Figure 2 shows that the hourly weather data was filtered to only include partly-cloudy-night and clear-night descriptions, to make sure we were only looking at visibilities during nighttime, when fog and total cloud cover would not obscure the sky. Additionally, several holidays were manually added to the UK Bank Holidays table, since our energy and weather datasets covered 2011, but the UK Bank Holidays table did not. After aggregating the average of continuous variables in the hourly weather table, grouped by Date, shown in figure 3. the weather and energy table were joined with an inner join. A left anti-join with the UK Bank Holidays table removed all of the bank holidays.

```
from pyspark.sql.functions import to_date

# create date column in all of the datasets
dailyEnergyDf = dailyEnergyDf.withColumn("Date", to_date(col("day")))
weatherHourlyDf = weatherHourlyDf.withColumn("Date", to_date(col("time")))
holidaysDf = holidaysDf.withColumn("Date", to_date(col("Bank holidays")))
```

Figure 1 Add date column.

```
# filter to only show data for nighttime
from pyspark.sql.functions import avg

filteredweatherHourlyDf = filteredweatherHourlyDf.filter(col("icon").contains("night"))

filteredweatherHourlyDf.show()
```

Figure 2 Filter for only night time.

For a future analysis, it would be interesting to examine energy usage and dark sky hours versus the demographic data, which could help create more targeted and effective dark sky marketing.

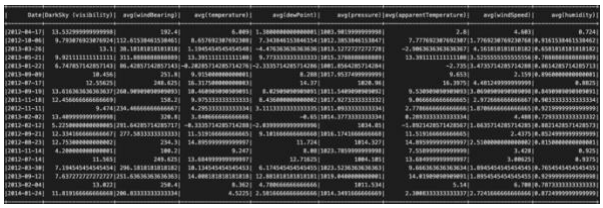

Figure 3 Table with averages of nighttime weather data grouped by date.

```
# Join the filtered and aggregated energy and weather tables
weatherJoinEnergy = aggWeatherHourlyDf.join(filtereddailyEnergyDf, aggWeatherHourlyDf.Date == filtereddailyEnergyDf.Date, "inner"
                                            ).select("DarkSky (visibility)",
                                                     "avg(windBearing)",
                                                     "avg(temperature)",
                                                     "avg(dewPoint)",
                                                     "avg(pressure)",
                                                     "avg(apparentTemperature)",
                                                     "avg(windSpeed)",
                                                     "avg(humidity)",
                                                     "energy_median",
                                                     "energy_mean",
                                                     "energy_max",
                                                     "energy_std",
                                                     "energy_sum",
                                                     "energy_min",
                                                     )

weatherJoinEnergy = weatherJoinEnergy.na.drop()

weatherJoinEnergy.show()
```

Figure 4 Join weather and energy tables and drop null values.

After pre-processing, we used the VecftorAssembler from PySpark Machine Learning to convert all of the columns into a list of Independent Variables. The variables used in the VectorAssembler are shown in Figure 5. The

```
from pyspark.ml.feature import VectorAssembler

# converting columns into independent features (variable)
# exclude string data type
featureassembler = VectorAssembler(inputCols=["avg(windBearing)",
                                              "avg(temperature)",
                                              "avg(dewPoint)",
                                              "avg(pressure)",
                                              "avg(apparentTemperature)",
                                              "avg(windSpeed)",
                                              "avg(humidity)",
                                              "energy_median",
                                              "energy_mean",
                                              "energy_max",
                                              "energy_std",
                                              "energy_sum",
                                              "energy_min"],
                                   outputCol="Independent Features")
```

Figure 5 Vector Assembler.

finalized data with the Independent Features and DarkSky (visibility) columns are shown in Figure 6. Once we had the Independent Features, we used a LinearRegression to train model on a 70:30 training:test split. We repeated this process for several other sets of features to see if a smaller subset of energy and weather variables would improve the accuracy: a subset of weather features (Figure 7), all weather features (Figure 8), and all energy features (Figure 9).

Independent Features	DarkSky (visibility)
[227.0,5.33625,2....	13.435000000000002
[256.5,4.68812499...	12.920000000000002
[240.352941176470...	12.968823529411766
[204.117647058823...	11.94764705882353
[235.615384615384...	12.796923076923076
[231.555555555555...	13.064444444444442
[217.470588235294...	13.297058823529412
[192.5,5.50857142...	12.532857142857141
[213.352941176470...	11.042941176470588
[289.470588235294...	11.22235294117647
[295.235294117647...	13.042352941176471
[242.647058823529...	10.803529411764705
[265.058823529411...	13.011176470588234
[247.117647058823...	9.331764705882353
[253.058823529411...	13.05764705882353
[261.375,8.848125...	12.755625000000002
[256.6875,5.67687...	13.4225
[204.7,9.3,5.5319...	13.366999999999999
[232.176470588235...	12.6
[263.705882352941...	13.081764705882351

Figure 6 Finalized Independent Features and DarkSky (visibility).

```
from pyspark.ml.feature import VectorAssembler

# converting columns into independent features (variable)
# exclude string data type
weatherfeatureassembler = VectorAssembler(inputCols=["avg(temperature)",
                                                     "avg(dewPoint)",
                                                     "avg(pressure)",
                                                     "avg(humidity)"],
                                          outputCol="Independent Features")
```

Figure 7. Finalized Independent Features and DarkSky (visibility).

```
from pyspark.ml.feature import VectorAssembler

# converting columns into independent features (variable)
# exclude string data type
featureassembler = VectorAssembler(inputCols=["avg(windBearing)",
                                              "avg(temperature)",
                                              "avg(dewPoint)",
                                              "avg(pressure)",
                                              "avg(apparentTemperature)",
                                              "avg(windSpeed)",
                                              "avg(humidity)"],
                                   outputCol="Independent Features")
```

Figure 8 All weather features.

```
from pyspark.ml.feature import VectorAssembler

# converting columns into independent features (variable)
# exclude string data type
energyfeatureassembler = VectorAssembler(inputCols=["energy_median",
                                                    "energy_mean",
                                                    "energy_max",
                                                    "energy_std",
                                                    "energy_sum",
                                                    "energy_min"],
                                         outputCol="Independent Features")
```

Figure 9 All energy features.

4. RESULTS

Various features existed in the dataset that could potentially be predictors for dark sky. To determine which features were most likely to produce correlations, three different models were trained, each model using a different combination of feature characteristics. To determine if energy was correlated with dark sky visibility, the first model was trained using Linear Regression and included features solely related to energy including: energy_median, energy_mean, energy_max, energy_std, energy_sum, and energy_min for each day. The findings of this model resulted in an r-squared result of only 0.0033 (Figure 10), indicating that there is no correlation between energy consumption and the level of dark sky visibility.

To further test the data, a second model was created that included various weather attributes for the day. Features for this model consisted of average temperature, dewpoint, pressure, wind speed and humidity for each day. A linear regression model was trained on this weather-related data and compared the level of dark sky visibility. Results showed an R squared outcome of 0.0322 (Figure 11), indicating minimal correlation between weather and dark sky visibility. However, reducing the number of features to include average temperature, dew point, pressure and humidity resulted in much higher correlation. Using the attributes above, the linear regression model predicted a 0.3849 R square value (Figure 12), suggesting that weather may be more correlated to dark sky visibility than energy attributes alone.

Lastly, the data points for both weather and energy were merged into a large dataset and trained again using Linear Regression. The resulting outcome included an R squared value of 0.4925 which indicates a moderate correlation, but not significant (Figure 13). Considering the increase of data features in this model, overfitting becomes a concern and may result in reduced accuracy when using the model with never-before-seen data. Given the results, it can be concluded that various attributes not tested in this research may be responsible for dark sky visibility. Certain features for predicting dark sky visibility in future research might include the season, elevation, or earth's current phase of orbit.

```
# evaluating the trained model with train data
train_results = trained_model.evaluate(train_data)

# Calculating Rsquared and printing value
print("The value of Rsquared is:", train_results.r2)
print("The model accuracy is {0:.0f}% with train data".format(train_results.r2*100))
```

```
[Stage 586:>                                                        (0 + 8) / 8]
The value of Rsquared is: 0.003394646251341449
The model accuracy is 0% with train data
```

Figure 10 0% accuracy for energy model.

```
# evaluating the trained model with train data
train_results = trained_model.evaluate(train_data)

# Calculating Rsquared and printing value
print("The value of Rsquared is:", train_results.r2)
print("The model accuracy is {0:.0f}% with train data".format(train_results.r2*100))
```

```
[Stage 636:>                                                        (0 + 8) / 8]
The value of Rsquared is: 0.03223064821031474
The model accuracy is 3% with train data
```

Figure 11 Partial weather model accuracy.

```
# Calculating Rsquared and printing value
print("The value of Rsquared is:", train_results.r2)
print("The model accuracy is {0:.0f}% with train data".format(train_results.r2*100))
```

```
The value of Rsquared is: 0.4925177622457696
The model accuracy is 49% with train data
```

Figure 12 Full weather model.

```
# evaluating the trained model with train data
train_results = trained_model.evaluate(train_data)

# Calculating Rsquared and printing value
print("The value of Rsquared is:", train_results.r2)
print("The model accuracy is {0:.0f}% with train data".format(train_results.r2*100))
```

```
[Stage 548:>                                                        (0 + 8) / 8]
The value of Rsquared is: 0.38493383057940067
The model accuracy is 38% with train data
```

Figure 13 All features model

5. CONCLUSION

Preserving dark sky hours proves to be an important attribute of our natural environment. Ensuring appropriate ecological cycles within our environment is imperative to the well-being of many animals living under the night sky. Although the importance of preserving dark sky hours is clear, the cause of a sudden reduction in night sky hours still requires additional research. This research aims to address some of the obvious causes of diminishing dark hours: weather and the effects of human energy consumption. The dataset used in this study represented accurate data in a relatively short period regarding weather and human energy consumption levels. The correlation between the data and night sky was shown to be insignificant, however certain features did predict night sky hours better than others such as temperature, humidity, dew point, and pressure. The level of human energy consumption was shown to have no correlation to dark sky hours. This research study dataset could be a limiting factor of why human energy consumption had no correlation to dark sky hours. Further research may be required to determine if human energy consumption does in fact have a long-term effect on dark sky hours. Furthermore, determining the direct cause of reduction in dark sky hours should continue to be a priority among researchers.

6. REFERENCES

D., J.-M. (2022, May 23). *Smart Meters in London*. Kaggle. https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london?select=darksky_parameters_documentation.html

González Ordiano, J. Á., Bartschat, A., Ludwig, N., Braun, E., Waczowicz, S., Renkamp, N., Nico, P., Düpmeier, C., Mikut, R., & Hagenmeyer, V. (2018). Concept and benchmark results for Big Data energy forecasting based on Apache Spark. Journal of Big Data, 5(1), 1-11. https://doi.org/10.1186/s40537-018-0119-6

Le Hoai, M. T., Chow, K. H. K., Luevisadpaibul, R., Thirunavukkarasu, G. S., Seyedmahmoudian, M., Horan, B., Mekhilef, S., & Stojcevski, A. (2021). Accurate Prediction of Hourly Energy Consumption in a Residential Building Based on the Occupancy Rate Using Machine Learning Approaches. Applied Sciences, 11(5), 2229. https://doi.org/10.3390/app11052229

Sarswatula, Pugh, T., & Prabhu, V. (2022). Modeling Energy Consumption Using Machine Learning. Frontiers in Manufacturing Technology (Lausanne), 2. https://doi.org/10.3389/fmtec.2022.855208

De Mattos Neto, P. S. G., De Oliveira, J. L. C., Bassetto, P., Siqueira, H., Barbosa, L., Alves, E. P., De Mattos Neto, P. S. G., Rissi, G. F., & Li, F. (2021). Energy Consumption Forecasting for Smart Meters Using Extreme Learning Machine Ensemble. Sensors, 21(23), 8096. https://doi.org/10.3390/s21238096

7. WORKLOAD ASSIGNMENTS

For this project, Nikki was the primary modeler, Marie was responsible for recording and finalizing the presentation, and Tyler was responsible for finalizing the paper. All of us had some input and review responsibilities for each other's tasks.