Xanda Limited

Synthetic EAC’s   
  
**Model Design Principles and Analytics v1.0**

**Privileged and Confidential**

**Hang Ruan**



**I. Introduction**

**1.1 Background**

Energy businesses and industries are aligning their energy strategies with national and global environmental policies. They have pledged to reduce carbon emissions to achieve Net Zero in an effort to address the energy crisis and climate change. Operating at the heart of the UK energy sector, ElectraLink's[[1]](#footnote-1) initiatives have played a pivotal role in integrating environmental sustainability actions into their business. They achieved Net Zero in 2022. Since then, they have continued with further plans for continuous improvement and have assisted the broader society in achieving Net Zero on a large scale. As part of this effort, smart meters, data analytics platforms, and new customer services have been developed. These tools provide a more comprehensive nationwide view of energy patterns, helping to better understand customer behaviors and decision-making processes.

While traditional electricity and gas meters provide only sporadic readings, smart meters capture half-hourly (HH) readings. The current industry system in the UK is not fully equipped to handle HH settlements for all energy consumers. As a result, a profiling process is needed to divide user readings between two dates. This is defined as Meter Advance (MA), which can be broken down into HH segments. MA can also be annualized to Annual Advance (AA) to represent the amount of energy a user consumes from April to March each year. In this context, a "user" refers to a Meter Point Reference Number (MPAN), a unique identifier for all UK energy users or properties.

To ensure customers receive accurate initial bills and realize effective cost savings, the energy to be purchased in the wholesale market and the total potential savings from tariff switching[[2]](#footnote-2) are considered. The Estimated Annual Consumption (EAC), defined by energy usage for dates since the most recent reading, serves as an effective estimate for future annual energy consumption. This estimate is processed in real-time.

**1.2 Challenges**

The market has devised various methods to approximate the EAC. These methods include requesting the monthly DD value, conducting surveys on the number of rooms and occupants, or even using the generic Ofgem standards of high, medium, and low consumption. However, these methods often suffer from significant inaccuracies and demand considerable time from the customer, leading to increased customer drop-offs. ElectraLink possesses the majority of recent EACs for meter readings transmitted from the D0019 flows via Dynamically TeleSwitched (DTS) meters.

The challenge lies in devising an effective method to generate more accurate EACs for all customers and to precisely estimate a customer's ongoing EAC whenever they submit a reading through the API. This encompasses scenarios where:

• A user with at least one previous EAC in the database can obtain a new EAC estimate upon submitting a reading.

• A user without any prior EAC in the database can still receive a new EAC estimate after submitting a reading. If this user has at least two previous readings stored, they will be added to the database.

• All new user reading submissions will be incorporated to update the database.

• The EAC database will be updated with new EAC estimates on a daily or weekly basis.

**1.3 Solution Development**

To tackle the persistent issue of imprecise EAC estimations and ensure customers consistently receive accurate EAC predictions at scale, we have turned to machine learning (ML) techniques. These techniques are specifically designed, trained, tested, and deployed as the optimal solution. ML, a primary subset of Artificial Intelligence (AI), is currently transforming business sectors and industries globally. It's a scientific discipline that leverages vast amounts of data, relying on a myriad of algorithms and statistical concepts, and typically involves an iterative process.

In this study, we will employ a renowned and competitive open-source supervised ML model called XGBoost[[3]](#footnote-3), which is backed by prior research[[4]](#footnote-4). XGBoost is a gradient-boosted tree ensemble model adept at processing extensive data sets of varied types, even those with missing values, without sacrificing accuracy. We will use this algorithm to accurately synthesize EAC values from the data provided by ElectraLink during training. Subsequently, we will deploy it as a model at an endpoint to serve the API for customers in the production process.

It's worth noting that we will also evaluate, compare, and implement the existing industry-standard approach to calculate EAC using a straightforward equation provided by ElectraLink, alongside the ML model.

In the subsequent sections of this paper, Chapter II will delve into the available datasets. Chapter III will discuss both the industry-standard method and the ML model. Testing outcomes and quality assurance will be presented in Chapter IV, with a summary provided in Chapter V.

**II. Datasets**

**2.1 Available Datasets**

For the Synthetic EAC project, ElectraLink has provided a database comprising six datasets, all managed under their AWS platform. This database is primed for queries on Athena, a robust and scalable AWS SQL service, with the actual data housed in S3. The datasets are:

* eac\_aas
* eac\_d39\_dpc
* eac\_d\_mpan\_gsp\_group
* eac\_d\_mpan\_profileclass\_stg
* gsp\_average\_eac
* meter\_readings

These datasets encompass crucial details such as meter readings, dates, meter attributes, geographical attributes, and the daily profile coefficient (DPC) for the majority of MPANs in the UK. The number of rows in these datasets ranges from millions to billions.

However, the original datasets required some initial data cleaning and wrangling since they were raw data with missing values across various fields. The essential information for the MPANs was scattered across these datasets. As a result, additional queries were executed to amalgamate them, using aggregation and restructuring operations under specific conditions to consolidate all pertinent information into one table. Leveraging the capabilities of cluster computing and the subquery features offered by Athena, we derived a final training dataset. This was achieved after overcoming challenges related to account access and query timeout constraints. The outcome of each subquery has been preserved as a separate table for tracking and recall purposes.

**2.2 The Training Dataset**

The training table, labeled "full\_eac\_table" within the database, consists of approximately 380 million rows. It encompasses around 22.5 million distinct "mpan" entries and their meter reading specifics. These specifics include:

* read\_date
* next\_read\_date
* effective\_from\_settlement\_date
* effective\_to\_settlement\_date
* sum\_dpc: This represents the summation of all DPC values between "read\_date" and "next\_read\_date".
* cal\_aa: Calculated using the industry formula cal\_aa = (next\_read – current\_read) / sum\_dpc.
* previous\_eac: This denotes the preceding EAC value for the specified MPAN on the given read date.
* current\_eac: This signifies the current EAC value for the specified MPAN on the given read date.

A snapshot of the training dataset is presented in Table 1; the schema details are outlined in Table 2.

**Table 1. Preview of the training dataset**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| mpan | read\_date | next\_read\_date | effective\_from\_settlement\_date | effective\_to\_settlement\_date | sum\_dpc | cal\_aa | previous\_eac | current\_eac |
| 4400030074526 | 2022-09-25 00:00:00.000 | 2022-10-01 00:00:00.000 | 2022-09-25 | 2022-09-30 | 0.0150607915042 | 1354.5104846787804 | 2112.6 | 2044.6 |
| 4400030074526 | 2022-10-01 00:00:00.000 | 2022-10-25 00:00:00.000 | 2022-10-01 | 2022-10-24 | 0.0616157247018 | 1598.6178930250007 | 2044.6 | 2023.8 |
| 4400030074526 | 2022-10-25 00:00:00.000 | 2022-10-26 00:00:00.000 | 2022-10-25 | 2022-10-25 | 0.0026340619692 | 607.4268634180773 | 2023.8 | 1971.4 |
| 4400030074526 | 2023-01-25 00:00:00.000 | 2023-01-26 00:00:00.000 | 2023-01-25 | 2023-01-25 | 0.0032508609005 | 3906.6574635803922 | 1284.2 | 1110.8 |
| 4400030074526 | 2023-01-26 00:00:00.000 | 2023-02-20 00:00:00.000 | 2023-01-26 | 9999-12-31 | 0.07909685298320002 | 5270.753314149534 | 1110.8 | 1129.0 |
| 4400030074526 | 2022-10-26 00:00:00.000 | 2022-11-25 00:00:00.000 | 2022-10-26 | 2022-11-24 | 0.08753744512039997 | 146.22313893665435 | 1971.4 | 1964.2 |
| 4400030074526 | 2022-11-25 00:00:00.000 | 2022-12-25 00:00:00.000 | 2022-11-25 | 2022-12-24 | 0.1010917752886 | 111.77961775565225 | 1964.2 | 1645.9 |
| 4400030074526 | 2022-12-25 00:00:00.000 | 2023-01-01 00:00:00.000 | 2022-12-25 | 2022-12-31 | 0.021960025616200002 | 163.93423500126818 | 1645.9 | 1335.7 |
| 4400030074526 | 2023-01-01 00:00:00.000 | 2023-01-25 00:00:00.000 | 2023-01-01 | 2023-01-24 | 0.0775493805336 | 166.3456227662681 | 1335.7 | 1284.2 |
| 4400030074578 | 2021-04-01 00:00:00.000 | 2021-04-29 00:00:00.000 | 2021-04-01 | 2021-04-28 | 0.07753470073970001 | 3753.15822752637 | 3212.1 | 3210.4 |

**Table 2. Training dataset scheme**

|  |  |
| --- | --- |
| columns | data format |
| mpan | *bigint* |
| read\_date | *timestamp* |
| next\_read\_date | *timestamp* |
| effective\_from\_settlement\_date | *date* |
| effective\_to\_settlement\_date | *date* |
| sum\_dpc | *double* |
| cal\_aa | *double* |
| previous\_eac | *decimal(16,1)* |
| current\_eac | *decimal(16,1)* |

Note the DPC values are distributed across different combinations of GSP Group ID (GSP), Profile Class (PC), Standard Settlement Configuration (SSC) and Time Pattern Regime (TPR) for each MPAN and also changes every day. The time range for the given training dataset is from 2021-01-01 to 2023-02-21. The goal is to accurately estimate the “current\_eac” and conform an algorithm or model that is fully or partially data-driven and can be stored as an asset to make accurate EAC estimations on new data feeds from customers at any time after 2023-02-21.

Counting for calculating new EAC beyond the given time range of the existing dataset, it is important that during the initial ETL process when acquiring data with relevant fields, that is to initially link, if possible, all MPANs with their sum DPC values from the date of their last available EAC to the date of the new EAC required to be estimated. Since the new EAC calculation date is unknown and depends on the user demand, for each MPAN there should be a field named sum\_dpc accounting for frequent updates of adding new DPC values of later dates into the sum\_dpc field. Hence the ideal data fields for the initial ETL will be the following:

* mpan
* previous\_read\_date: the date of the previous meter reading
* previous\_meter\_read: the meter reading on previous\_read\_date
* previous\_eac: the previous EAC estimate on the same day of the previous meter reading date
* new\_read\_date: the date of the new/latest meter reading date
* new\_meter\_reading: the meter reading on new\_read\_date
* sum\_dpc: the sum value of all DPC values from previous\_read\_date to new\_read\_date

Alternatively, if the calculation of AA can take place in the ETL as well, then the expected fields will be:

* mpan
* previous\_read\_date
* previous\_eac
* new\_read\_date
* sum\_dpc
* calculated\_AA: (new\_meter\_reading - previous\_meter\_read)/sum\_dpc

The initial linkage of MPAN and DPC is a key process that underpins the data pipeline. Ensuring it is accurately set up at the backend will minimize possible delays in the rest of the pipeline and maximizes data efficiency and reduces unnecessary further data processing for the model pipeline.

**III. Models**

**3.1 The Industry Version Approach**

An industry version approach[[5]](#footnote-5) has been provided, possibly as a benchmark to estimate EACs. It is described as in a simple equation as

,

for which the key arguments AA, sum\_dpc and previous\_eac are already introduced in the last Chapter. The interpretation is straightforward: if the sum of the DPCs for the advance is > 0.5, the AA is used as the EAC. Otherwise, a proportion of the previous EAC is used to account for the whole year (effectively a volume weighted average)5. The biggest advantage of this approach is that it simplifies the process into straightforward calculation and easy to explain to customers to have them accepting the estimation confidently.

The drawback of this approach is that it lacks accuracy and does not do well for customers with irregular consumption behaviors. In ElectraLink’s note5, it admits EAC synthesis proves to be more complicated: lack of previous EAC (which could be pre-estimated from readings and DPCs); having to make sure sum\_dpc > 0.5 to maintain some level of accuracy that leads to some manual checks. For example, if a customer needs a EAC quote soon enough after last quote so that sum\_dpc << 0.5, it will require to use the previous EAC from last quote (or the further previous EAC from the current quote), a change of equation or manual adjustment to use a different previous EAC is needed.

**3.2 Other Considerations**

One option is to construct a time series forecasting model for each MPAN, utilizing statistical models like the Autoregressive Integrated Moving Average (ARIMA) to predict its future EAC directly. This approach would be less influenced by correlations with other parameters. However, given the vast number of MPANs, it's impractical to create a unique model for each. More critically, some MPANs have only a few readings, making the use of time series statistical models illogical. A time series model also necessitates the incorporation of new actual data into the database to ensure the historical data used for predictions remains accurate; otherwise, a decline in performance can be anticipated. Parallelism-capable models, such as Vector Autoregression (VAR), can circumvent some model and processing limitations but will still face data constraints for certain MPANs.

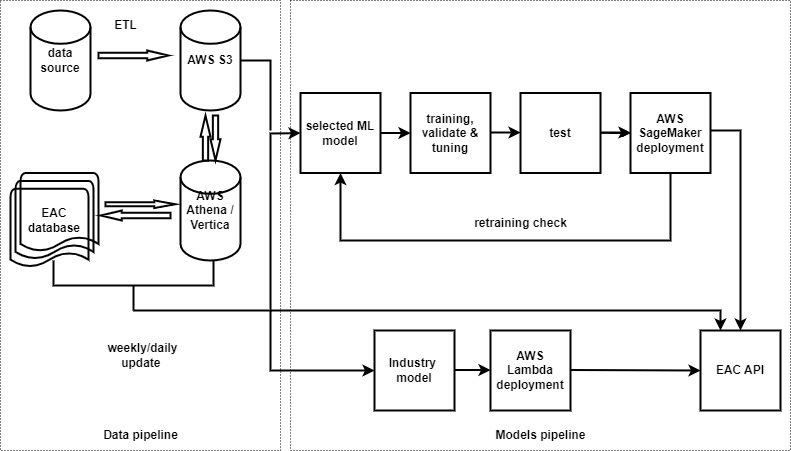
It's worth considering leveraging other information in the datasets, including the DPCs and meter readings, similar to the industry version approach. However, rather than relying solely on this approach, we can depend on an ML model to discern deeper correlations between these data points. This would enhance accuracy without concern over whether the sum\_dpc for a given MPAN exceeds 0.5. Consequently, we can employ the same three inputs from the industry version approach — "sum\_dpc", "cal\_aa", and "previous\_eac" — along with the synthesis target "current\_eac" from the training dataset. This will train the ML model to yield optimal results.

**3.3 The ML Model – XGBoost**

XGBoost is an extensive library comprising a collection of gradient-boosted tree models, suitable for efficient supervised regression. The XGBoost algorithm series is scalable and versatile, compatible with various environments, including Amazon Web Service (AWS) and Google Cloud Platform (GCP). It often delivers highly accurate results due to its boosting policy. An XGBoost model offers numerous tunable hyperparameters, such as tree depth and learning rate. With optimized hyperparameter settings, it employs a collection of decision trees (DT) to execute regression on different data subsets for each tree, boosting through them in parallel. XGBoost has been successfully applied and tested across various real-world applications, yielding satisfactory results. Notably, it secured first place in a Kaggle big data competition shortly after its initial release.

**3.4 Training and Deployment Considerations**

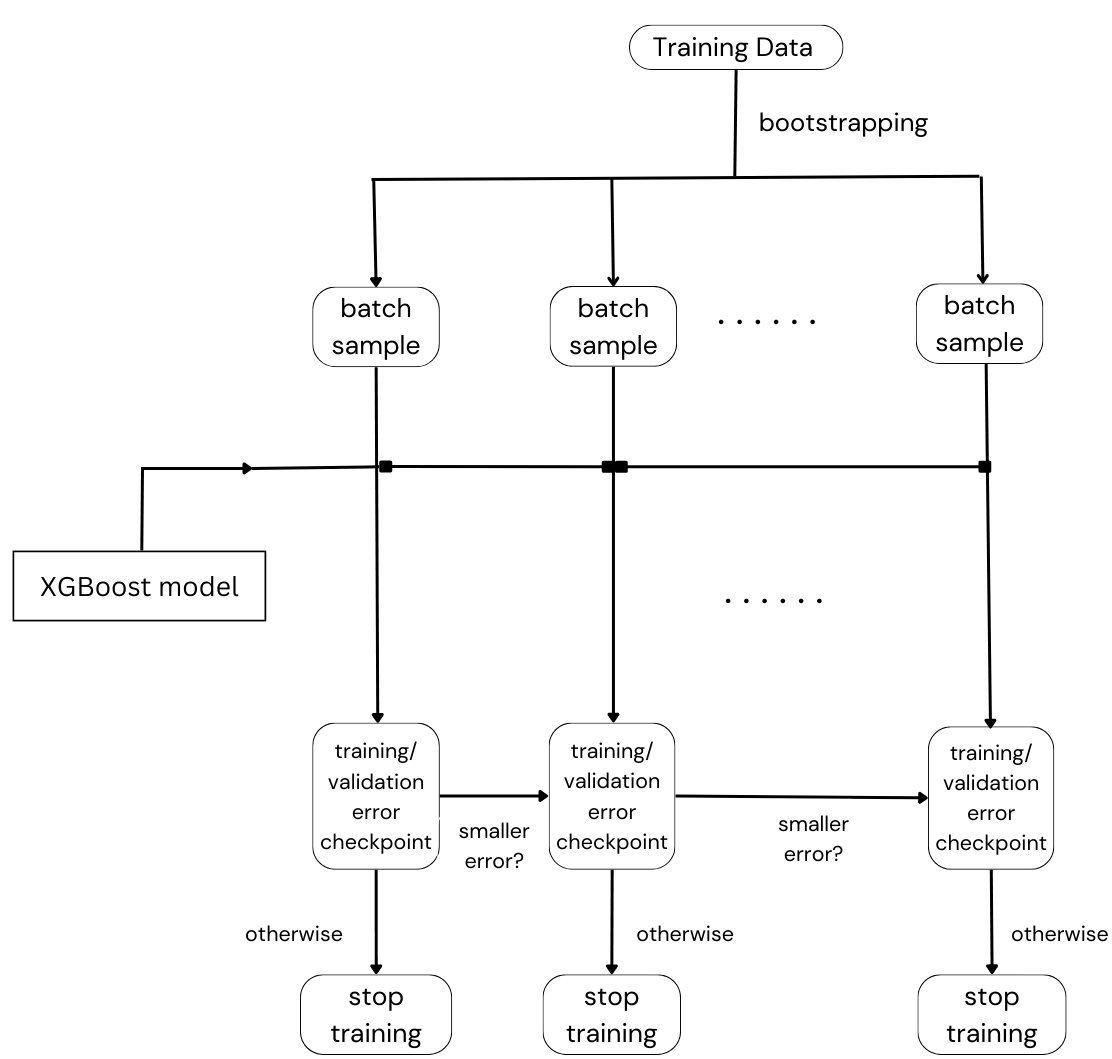
For the ML model, a standard ML pipeline is envisioned, sourcing data from EL’s S3 and Athena AWS account. This pipeline is an integral component of the synthetic EAC project workflow, which combines the data pipeline and the model pipeline, as depicted in Fig. 1. The data pipeline encompasses the ETL process, sourcing data from the EL data lake, storing it in AWS S3, and subsequently loading it into secure and scalable storage systems like Vertica or Athena, which also support query execution. The model pipeline consists of two models. For the ML model, it details how the model is chosen for the data, trained, hyperparameter-tuned during validation, tested on test data, and finally deployed for API serving. The industry version model, being based on direct calculations, doesn't necessitate training. The EAC API can also fetch data directly from storage when only retrieving a record of previous EACs. The workflow illustrates that these pipelines are ongoing processes, especially when energy customer behaviors evolve at scale with data updates. At certain junctures, the model may require retraining to maintain its accuracy and relevance. Adopting such practices aligns well with continuous integration and continuous delivery (CI/CD) principles, automating the overarching workflow within a broader context.



**Fig. 1. Synthetic EAC workflow**

**Training**

There are two options to train a XGBoost model: option (a) is to call a typical regression model from official XGBoost API and train it on AWS; option (b) is to directly call the built-in XGBoost model library from AWS SageMaker, a cloud-based ML platform to train and deploy ML models for a diverse range of business needs. The main difference between the two cases are model versions and tunable hyperparameters. In our case, the latest XGBoost version available in SageMaker is “xgboost-1.7.1” and is available to use with clearly defined hyperparameters readily accessible for tuning.



**Fig. 2. Model training process**

The machine learning model training process is demonstrated in Fig. 2. Since we are using XGBoost for tabular regression on a massive tabular dataset (i.e., the training table mentioned in the last chapter), loading and training on the entire training data at once are practically impossible, so we leverage a sampling technique named bootstrapping random sampling, to formulate training batches in smaller sizes in a simple Monte-Carlo style. In this way of sampling, there are two benefits: the SageMaker instance could have enough memory space to hold the data at once for a single training session; the number of samples are far enough provided there are only three inputs as features to train the model thus it is very likely the case of oversampling, which leaves a large room for validation and testing.

Although the scales of the input features differ significantly, XGBoost is a highly scalable and tolerant ensemble model which is capable of handling unnormalized raw feature data without much need for feature engineering. Initial experiments also prove feature normalization even negatively affects the prediction accuracy.

Practically, we have around 380 million samples, which are divided into twenty smaller batches, where each batch comprises about 10 - 30 million samples. Moreover, there is also a 5% of data (~ 19 million samples) preserved for model validation to ensure the model is trained well after each batch training. Next, the model is trained over the batches one by one, with an exit criterion on the training and validation error check, which is determined by if the training or validation error is reduced after each batch training. If the errors already stop in the training process on the first batch, then the entire training process stops before moving the model to the second batch; in contrast, if training or validation error is still decreasing after training finishes on one batch, we will move to the next batch to keep training the model. The model hyperparameters are continuously tuned through training, regardless of which batch the training process is at. Moreover, manual hyperparameters tunning is used to get a better control and insights of the training process. As a matter of fact, experiments do not show any significant performance differences among different hyperparameter sets, the optimal hyperparameter set is chosen and listed in Table 2.

As envisioned, the model is always able to achieve the minimum training error before finishing training on the first one-two batches (equivalent to a maximum of 10% of the training samples), with different hyperparameter sets. This verifies that 380 million samples are practically an abundance amount of data for training for an ensemble model like XGBoost even though it often comprises numerous trainable parameters and is subject to overfitting if there is a training data deficiency issue. This is certainly not the case and we can confidently say that the trained model is unbiased and can perform consistently well on any testing EAC data when provided. This leaves sufficient data solely for testing purpose and will be verified in the test result chapter.

**Table 2. Training specification**

|  |  |  |
| --- | --- | --- |
| Model | XGBoost-1.7.1 | |
| Hyperparameters | eta | 0.2 |
| gamma | 4 |
| Min\_child\_weight | 6 |
| subsample | 0.8 |
| Max\_depth | 5 |
| Num\_round | 100 |
| Training/Validation/Test | training | 10% |
| validation | 5% |
| test | from 10% up to 100% |

**Deployment / Production Configuration**

For deliberate consideration, the two models are deployed at different places with different AWS services: the industry model is deployed at AWS Lambda service, a serverless platform runs code with customizable layers responding to events; whilst the ML model is conventionally deployed on SageMaker inference endpoint that only runs when the instance is active. These deployments are advised despite a SageMaker version deployment for the industry model has also been built on hardcoded docker-container build. However, the industry model is rather very simple and not even close to the complexity of any ML model whilst SageMaker is a ML oriented service. Additionally, SageMaker inference endpoint instance incurs cost as long as it is running or deployed for production in real time so it could incur double cost for running endpoints. Note there are cases when multiple models can be deployed at the same endpoint, but it requires the models to be in the same farmwork when calling the API even with underlying traffic loads balancing control challenges. Considering above, the best solution is a combination of Lambda and SageMaker for deploying the models.

**IV. Testing Result and Quality Assurance**

**4.1 Numerical Test Results**

Now we will be looking at the testing results of how both models perform on the terms of synthetic error. Two error metrics Mean Absolute Percentage Error (MAPE) and Weighted Mean Absolute Percentage Error (WMAPE) will be tested for both models on the complete dataset. MAPE and WMAPE are defined as following:

,

.

In the above equations, and are the predicted and actual EAC values. We define a standard EAC is accurate if it is close to its actual EAC value, which is given in the EAC-AAS dataset provided by the EletraLink. The EAC-AAS dataset provides the industry EAC/AA values from actual D0019 flows from DTS.

Tests are carried out at two different environments – AWS Athena and Python 3.9. The testing result are shown in Table 3. From Table 3 it is clear the ML XGBoost model is the winner by reducing both the MAPE and WMAPE by a substantial amount compared to the industry approach for EAC synthesis. Note that when calculating these errors, only EAC with values greater than 100 is used, to avoid divide-by-zero that destabilizes the MAPE accuracy. For inference time required for real time deployments, both solutions show similar behaviors – they both require less than one microsecond (ms) to make a single inference for the inputs and require approximately 0.1 ms for a tested 10,000 inputs at their endpoints. It is important that during deployments, the inference payload sizes need to be optimized to balance the inference cost and throughput demands.

Particularly, when looking at prediction accuracy on the fraction of data with gap of days of any two consecutive EACs within 45 days, the industry model is only at 34.6% MAPE and 37.7% WMAPE whilst the ML model is at 3.3% MAPE and 4.6% WMAPE. For the fraction of data with gap of days greater than 45 days, for prediction on the data with 45 days gap or more, industry model accuracy is at 61.7% MAPE and 56.7% WMAPE whilst the ML model is at 9.1% MAPE and 10.2% WMAPE. However, if looking at testing on the fraction of data with gap of days more than a year, the industry model accuracy dropped to 97.7% MAPE and 81.8% WMAPE; but the ML model only decreases to 12.6% MAPE and 13.7% WMAPE. It tells both algorithms lose accuracy when predicting next EAC one year out, but the ML algorithm is able to maintain much better consistency at much less accuracy loss and does not fall out 15% at worst; the industry algorithm however drops accuracy fast and can exceed 100.0% MAPE/WMAPE errors. In average, factoring in all EAC data, the industry model accuracy is at 38.2% MAPE and 40.1% WMAPE whilst the ML model is at 5.6% MAPE and 6.0% WMAPE. These testing are done in the AWS Athena environment where the very small fraction of data EAC volumes close to zero (< 100) have been removed before computing the errors.

**Table 3. Performance test for both models**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | EAC gap of days range | MAPE | WMAPE | environment | single inference | batch (10,000) inference |
| Industry approach | any | 16.6% | 31.3% | Python 3.9 | < 0.1 ms | ~ 0.1 ms |
| any | 38.2% | 40.1% | AWS Athena | N/A | N/A |
|  | <= 45 | 34.6% | 37.7% | AWS Athena | N/A | N/A |
|  | > 45 | 61.7% | 56.7% | AWS Athena | N/A | N/A |
|  | > 360 | 97.7% | 81.8% | AWS Athena | N/A | N/A |
| ML (XGBoost) | any | 5.2% | 6.0% | Python 3.9 | < 0.1 ms | ~ 0.1 ms |
| any | 5.6% | 6.0% | AWS Athena | N/A | N/A |
|  | <= 45 | 3.3% | 4.6% | AWS Athena | N/A | N/A |
|  | > 45 | 9.1% | 10.2% | AWS Athena | N/A | N/A |
|  | >360 | 12.6% | 13.7% | AWS Athena | N/A | N/A |

**4.2 Quality Assurance**

Only after testing the trained ML model on the vast majority of available data can we draw up the quality assurance of the model for its production process.

The industry approach uses an out-of-date oversimplified two-case linear equation to synthesize customer energy EAC without paying attention to the customer energy usage behavior, which changes over the year. It is economically unfriendly and unsustainable. It has been tested on the 380 million real customer EAC data and proved to be very inaccurate with 16.6 - 38.2% MAPE and 31.3 - 40.4% WMAPE errors. The errors can be particularly high for customers who use little energy during a certain range of time.

The ML model can easily overcome the industry approach’s weakness and can be readily adopted in the modern digital products environment. It is compatible with new business models and propagates values through the value chains in any complex business form. The ML model provides a ground breaking innovative solution to the challenging problem of EAC synthesis or estimation in the energy industry. It initiates global energy industries to refine their product strategies and steps out in the competition, particularly when dealing with the current global energy crisis and net-zero challenging. As an open-source ML model, XGBoost has earned the favor of the business world for years and proven its success in performance and scalability in solving challenging use case problems. It can be added into existing energy data pipeline without much barrier and efficiently trained on different datasets of any schema. Once the model is trained, the deployment and production process are straightforward and fast at speed. The model can also be pipelined and adjusted to customer behaviors if a retraining is required though it is unlikely this will be required. The model can be easily automated and monitored in the production process and serve customers at scale. Most importantly, the ML model has been tested on all the 380 million real customer EAC data on two different platforms and proven to be very accurate with 5.2 - 5.6 % MAPE and 6 % WMAPE errors. These are a 5-fold substantial error decrease and huge improvement on the EAC synthesis accuracy. It has substantially improved the data quality of the current dataset as a completely sellable and profitable data product. Also, the ML model counts on the exact same inputs as the industry method, thus the very same information data from the users to make a prediction, therefore no change of user data protocol or requirement is needed. Once the ML model is operational in production, it is expected to deliver energy bill quotations that are at least five times more accurate. This enhancement will lead to improved customer satisfaction, resulting in fewer negative reviews, complaints, or unnecessary upfront overpayments. Customers will gain better control over their financial commitments and will benefit from the ability to track their energy consumption with precise real-time estimates. This feature is particularly beneficial for individuals grappling with cost-of-living challenges, enabling them to plan their energy usage more effectively.

**V. Summary**

In this work, the synthetic EAC project has been conducted to develop a sustainable and scalable workflow from database ETL to EAC API updates in real time. Initial big data query work has greatly improved the data quality of the original EAC datasets by dropping the irrelevant columns through a series of efficient subqueries on AWS Athena, as well as rows with null values in the key fields, as part of data governance effort. The query result table is then obtained with around 380 million rows for 22.5 million MPANs across the UK. The query result table has all the key columns so thus it is used as a training and testing data that provides the key inputs and outputs to train the ML model in a supervised learning manner for it to generate accurate synthetic EAC values that have also been included in the final result table as a reference benchmark dataset for loading into the EAC database. Test results show the ML model is able to achieve much better accuracy on both MAPE and WMAPE error comparing to the existing industry approach.

The next phase would be to activate the model endpoints once the EAC API is uprunning and set a weekly or daily update scheme for the data pipeline to fully operationalize the system. Once the developed data pipeline and efficient EAC estimation model are productionized and made customer facing, millions of users will have improved user experiences and get accurate energy quotations any time they submit a new meter reading.

1. <https://www.electralink.co.uk/> [↑](#footnote-ref-1)
2. <https://www.electralink.co.uk/eac-api-solution/> [↑](#footnote-ref-2)
3. [https://xgboost.readthedocs.io/en/latest/index.html#](https://xgboost.readthedocs.io/en/latest/index.html) [↑](#footnote-ref-3)
4. Tianqi C and Carlos. G. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785 [↑](#footnote-ref-4)
5. Ian S, ElectraLink EAC/AAs, ian.Scougal@electralink.co.uk. [↑](#footnote-ref-5)