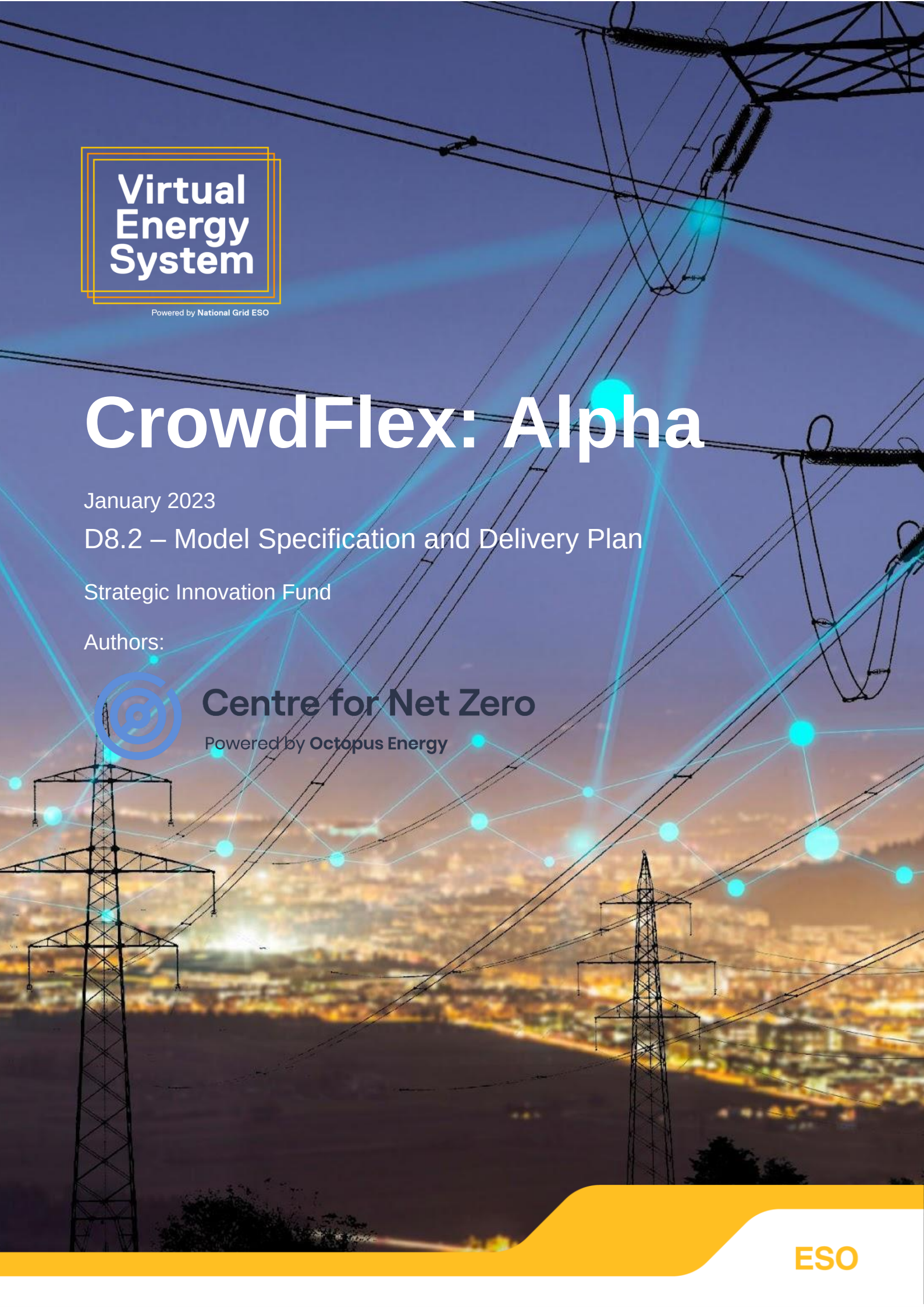


CrowdFlex: Alpha

January 2023

D8.2 – Model Specification and Delivery Plan

Strategic Innovation Fund

Authors:

Centre for Net Zero
Powered by Octopus Energy

ESO

1. Purpose of this document

In 2021, National Grid Electricity System Operator (hereafter, referred to as ESO) launched an ambitious, industry-wide mission to digitise our energy system. This world first, real-time replica of our entire energy landscape will work in parallel with our physical system. A shared industry asset, the Virtual Energy System (VirtualES) will improve our simulation and forecasting abilities to support the long-term vision to operate a zero-carbon electricity system.

To develop this ecosystem of interconnected models and APIs, there also needs to be clarity on the use cases and the users. This understanding will allow the inputs and outputs of a specific model to be defined, which in turn allows for standardised interfaces to be built. This will make it easy for the various users like ESO to interact with many different "replicas" of the same model, as the API interface will be consistent.

There will eventually be many components to the VirtualES. As demonstrated in numerous trials, and previous CrowdFlex projects, domestic flexibility is ready to be operationalised repeatedly and at scale as a core component of our future energy system. A 'firm' grid resource is one that when dispatched, provides a (near-)certain level of response. Domestic flexibility is inherently not a 'firm' resource, but modelling and embracing the probabilistic nature (its *stochasticity*) will ensure we can maximise the value from increased renewable generation, by intelligently matching demand with supply.

These models will give users unprecedented visibility of aggregated domestic assets and their forecasted demand. For example, by linking together different data streams and processes, forecasts of flexibility can be used to operate a net-zero energy system fit for the future.

The CrowdFlex project aims to further the establishment of domestic flexibility as a reliable energy and grid management resource by furthering the development and understanding of the probabilistic nature of domestic demand. A large-scale trial will be used to gather operational evidence, and approaches to modelling domestic flexibility will be developed using statistical techniques, including a consideration of the data exchanges required between flexibility service providers and system operators.

In this paper we outline how the model(s) should be specified to enable CrowdFlex, and provide an outline plan for how to deliver them. This is based on a literature review of common best practice, as well as a range of interviews with VirtualES stakeholders.

2. Contents

3. Terminology

Terminology	Definition
Baseline (also “counterfactual consumption profile” or “forecasted demand”)	The demand profile - electricity consumption in each half hour - of a household or asset if no flexibility request has been made.
Counterfactual	an alternative
CI/CD	Continuous Integration/Continuous Deployment - a software engineering methodology for testing and deploying incremental changes to a piece of software
ESO	(National Grid) Energy System Operator
Flexibility (Service)	The ability to shift power/energy to different times of the day. A flexibility service is an agreed mechanism where a Flexibility Service Provider is paid financially for providing this flexibility, and (a proportion of) this value can then be passed on to consumers.
Flexibility Service Provider (FSP)	An entity that is providing flexibility to the electricity system.
GSP (Group)	Grid Supply Point (Group) - the “spatial units” that make up the GB grid. Adjacent GSPs form a GSP Group, which are regions of the Great Britain (such as London, North West etc).
Low Carbon Technologies (LCTs)	A technology that can be used in a way that is low-carbon and reduces its dependence on fossil fuels. In this context, we mean electrified assets such as Electric Vehicles (EVs), Heat Pumps (HPs), Solar Photovoltaic (Solar PV) and batteries. These technologies are low-carbon since their energy usage can be shifted to times of low-carbon generation. By contrast, gas boilers or heating would be high-carbon.
Minimum Viable Product (MVP)	An early version of a product or tool that meets the minimum necessary requirements as defined by the users, and is integrated in a basic way to other systems and processes. It can, and will, be improved over time through feedback from users, but marks the “first version” that can be used to solve a use case.
One-hot encoding	A data preparation technique for machine learning models, where we encode a categorical variable that takes N distinct values as N separate binary variables.
Point load forecast	A model that produces a single value estimate for a given time step

4. Executive summary and key recommendations

In this section we give a quick summary of key recommendations. It is recommended that the reader refers to the specific sections for a more detailed discussion of why these are our chosen recommendations.

Overall, we are recommending that the first use-case of the VirtualES from a modelling perspective is to develop models for response services. It was identified in CrowdFlex Deliverable D4.1 that Constraint Management is a valid grid need for which domestic flexibility would be suitable, and in CrowdFlex Deliverable D8.1 that this would be among the most high-value use cases. It is for these reasons that we prioritise model development for this use case.

To enable this, we propose developing an updated view of assets (by GSP / GSP Group) that are participating in grid services, as well as a deterministic bottom-up demand forecast for those assets day-ahead. These models will ultimately be owned by the Flexibility Service Providers (FSPs), but follow a common API schema. We are then proposing a probabilistic model at GSP Group level, which is owned by ESO post-Crowdflex, since this model will be targeting the constraint management service, which is an ESO service. A similar model specification could be deployed for other ESO / Distribution System Operators (DSO) routine services. We are proposing a linear regression model for the demand forecast and a quantile regression model for the flexibility model, though we will iterate with different modelling architectures during CrowdFlex when running the trials.

4.1 Model specification (section 6)

- Flexibility Service Providers (FSPs) in CrowdFlex will operate according to a common API Schema, to enable interoperability within the VirtualES.
- Each FSP on CrowdFlex should develop their own model, since this is reflective of how the VirtualES will operate in practice (decentralised) but development will be co-ordinated by a model lead.
- The proposed endpoints, to be updated at least daily (but potentially more frequently in the case of asset operators) will serve three core purposes:
    - **/assets** - An aggregated view of low carbon technologies (LCTs) and assets, by GSP Group. Owned by the FSP; summing over FSPs increases visibility for all VirtualES users, including ESO and DSOs.
    - **/demand** - A deterministic forecast of demand for all assets opted in to CrowdFlex, day-ahead at half-hourly granularity. Owned by the FSP; summing over FSPs increases visibility for all VirtualES users, including ESO and DSOs.
    - **/flexibility** - A probabilistic forecast of (total) flexibility, summed over providers, and split out by quantiles. The model for specifically predicting flexibility on constraint management services will be owned by ESO (post-CrowdFlex). The same spec could be implemented by other end-users such as DSOs.
- The proposed modelling techniques will be (simple / quantile) linear regressions, based on lagged consumption features, as well as household characteristics (as defined in CrowdFlex deliverable D2.1), temporal variables and external variables (e.g., weather).

- We are proposing that suppliers use smart meter data to forecast and settle flexibility, whereas asset operators use data directly from the asset (resampled to 30 minute intervals) as a “pseudo-smart-meter” to forecast and settle flexibility.
    - Different providers will update their forecasts at different intervals, depending on the nature of their data, but this will at least be daily
    - Using the asset as a proxy for a smart meter would enable households without a smart meter to engage in flexibility events prior to a smart meter being installed
    - CrowdFlex will investigate disambiguation of load, and provide recommendations to mitigate against “double counting” of flexibility, from a modelling perspective
- CrowdFlex will test the feasibility of splitting forecasts out by low carbon ownership.
    - This is trivial to do for asset operators, but households with smart meters will be hard to disaggregate entirely.

4.2 Model delivery plan (section 7)

- The project delivery should be agile, and be delivered by a technical product squad.
    - This squad should be “self-contained” and consist of a technical lead, data scientists, data engineers and a product manager.
- The model should be iteratively trained and deployed, with an Minimum Viable Product (MVP) in the hands of ESO end-users as soon as feasibly possible (and ideally before the completion of the first batch of trials).
- The tools (e.g., model dashboards) should be developed in sprints jointly with CrowdFlex stakeholders, and optimised for end-user value creation. These stakeholders will include ESO and wider consortium members, but also external parties. While potentially useful for all stakeholders, the tools that are developed should, in particular, help ESO stakeholders get more comfortable with the integration of probabilistic forecasts and ingesting data from a range of providers – a key aim of CrowdFlex.
- The model should be developed using modern techniques (including cloud technologies, API frameworks and libraries), be well tested and continuously iterated and deployed (CI/CD).
- Model performance should be monitored after each trial, and retrained if it falls outside acceptable thresholds.
    - It will be important to work with grid stakeholders to define the thresholds that represent the “new normal” for working with probabilistic forecasts that balance risk with operationalising a grid system fit for the future.

By building a consistent model specification, FSPs can engage in a range of grid services by “building once, but using many times”. CrowdFlex will also investigate potential routes for bidirectional streams of data transfer between grid operators (ESO / DSOs) and FSPs. In other words, by building systems that enable grid operators to “pull in” data from FSPs, we will investigate which data sets are useful for FSPs to “pull in” from grid operators that would improve the quality of the flexibility service they are providing to the grid.

5. Summary of the literature review

Note: For the full Literature review, the reader is directed to Appendix B.

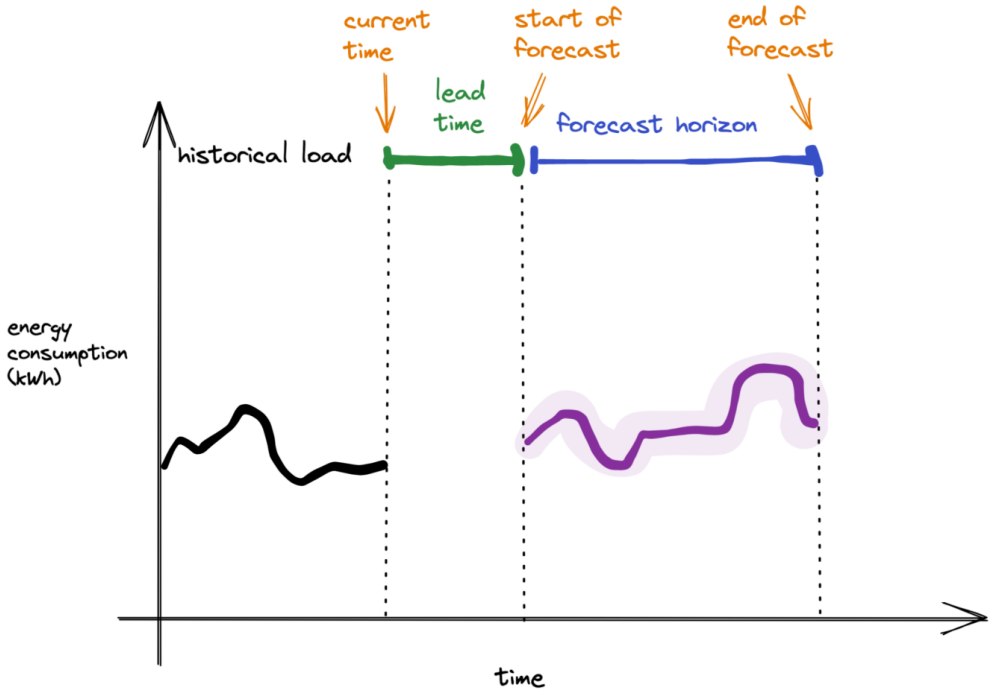
Energy consumption forecasts, or load forecasts, are used in many applications across the energy sector, from use cases in network design and planning such as locating and sizing capacitors[1], to network control applications such as a scheduling system for battery energy storage[2], and to anomaly detection such as theft detection in a smart grid[3] or load behaviour after a loss of supply[4].

The following components are used to define a forecast:

- the forecast horizon (i.e., the time range over which the forecast predicts);
- the lead time (i.e., whether the forecast is for the next hour, day, year, etc.);
- the required data granularity (i.e., half-hourly, daily, etc. - also called the *grain* of the data);
- the spatial aggregation (e.g., over individual units - such as the household - or over whole areas - such as GSP Groups).

The level of aggregation can impact how volatile the load is and therefore impacts the predictability of future load. The other components are defined by the use case.

Figure 1: Visual definition of key terms in a forecast



[1] A. Dupka, B. Venkatesh and L. Chang, "Fuzzy Stochastic programming method: Capacity planning in distribution systems with wind generators," IEEE Transactions on Power Systems, vol 26(4), pp 1971-1979, 2011.

[2] C.J. Bennett, R.A. Stewart and J.W. Lu, "Development of a three-phase battery energy storage scheduling and operation system for low voltage distribution networks," Applied Energy, vol. 146, pp. 122-134, 2015.

[3] V. Loia, G. Fenza and G. Mariacristina, Drift-aware methodology for anomaly detection in smart grid," IEEE Access, vol. 7, pp. 9645 - 9657, 2019.

[4] L.T.M. Mota, A.A. Mota and A. Morelato, "Load behaviour prediction under blackout conditions using a fuzzy expert system," IET Generation, Transmission & Distribution, vol. 1, no. 8, pp. 379 - 387, May 2007.

The techniques below are generic forecasting methods that can be applied for day-ahead or intra-day forecasting (and for manual or automated dispatch). The key changes between the aforementioned use-cases is the deployment of the model; intra-day forecasting is likely to be more costly to deploy a model, since the requirements on the data pipelines and reliability are different from models that are updated less frequently (such as day-ahead).

Since each household may need to be rewarded for their participation proportionally, forecasts will need to be generated at the household level. Unlocking demand side response for domestic customers will require a few components from a technical perspective:

- *baseline* or *counterfactual*, an estimate of the energy consumption had a flexibility request not been made,
- a method to model how the baseline might change as a result of the flexibility request, and
- a measure of the energy consumption of those who participated in a particular flexibility service.

The above assumes that the forecast takes inputs about consumption, but also other factors that influence the *flexibility capital* of the household we are forecasting. A more detailed discussion of this, as well as example inputs, is given in the literature review in CrowdFlex deliverable D2.1. We should ensure that the models perform fairly across a range of customer attributes.

5.1 Load forecasting algorithms

There are different types of load forecasting methodologies that are outlined in the literature, and perform in varying ways depending on the use case.

5.1.1 Rule based algorithms

Rule based algorithms are the simplest form of prediction. Common time-series forecasting rule-based algorithms include:

- Persistence algorithms - e.g., using the day before, or “previous similar day” to predict the next day;
- (Weighted) average algorithms - e.g., using a four-week (weighted) average to predict a given day;
- Rolling average models - e.g., using a combination of daily, weekly and monthly rolling averages to predict a given day.

These simple, explainable algorithms are often called “naive” (forecasting) models. They are frequently used to benchmark the performance of more advanced models.

Table 1: A simple outline of the pros and cons of rule based algorithms

Pros of rule-based models	Cons of rule-based models
• Simple to understand, and explain • Easy to deploy and maintain • Scale well	• Potentially less accurate than more advanced methods • “Static” - i.e., they do not learn from new data and behaviours

Inputs to rule based algorithms can include historical (lagged) consumption as well as other variables such as weather, household characteristics and calendar variables (such as “day of week” or “month of year”).

5.1.2 Statistical models

Unlike rule-based methods above, which output a point estimate (a single prediction for each unit), statistical models can output a distribution of values. These have been used successfully in time-series forecasting problems to help quantify a range of possible outcomes and understand risk. They are increasingly common in energy system research as they help model the volatility of different processes.

The main problem with statistical models is that some models do not scale well over large populations.

5.1.3 Machine Learning (ML) Models

Machine learning models are models that learn their parameters explicitly based on the (training) data. In supervised learning, which is the focus of this literature review and the task for CrowdFlex, we collect *labelled* data of a set of *features* about an event, and the *target* variable we want to predict. The process for collecting data and training these models will be deferred to the Model delivery plan (Section 7) plan, but as an overview:

- Data is collected through trials and experiments.
- Models are specified, and their parameters are learned from this historical data.
- The model(s) are iterated and updated based on collecting new data.
    - Different types of model architectures, and model inputs can be tested to improve accuracy.
    - The model(s) are retrained on a basis determined by the use case (e.g., monthly).
- Once a machine learning model has been trained on historical data, it can be “packaged up” and used to predict new events (known as *inference*).
- This model can be “retrained” on new data to learn the latest behaviours.

Machine learning algorithms have been shown to scale well over large populations and are usually more accurate, but this comes at the expense of potentially being less explainable. Another benefit of ML models is that they are not static, they can continue to *learn* from new data, and update their model parameters to capture a shift in behaviour (provided the model is expressive enough).

5.1.3.1 Computational cost of machine learning models

Simple ML models can be run locally on most laptops. However, when “tuning” these models (a process by which the model is tweaked over several iterations - changing the model architecture or hyperparameters) the model training process needs to be run several times. It is not uncommon to train hundreds of different variations of models to improve performance on a key metric while in this experimental stage.

More complex ML models, such as deep learning methods, may require increased memory (RAM) and storage space. It may need to be run solely in the cloud as the dataset(s) and/or model cannot fit on a standard laptop.

The computational cost is proportional to:

- The training time per model.
- The number of (variations of) models that are trained.
- The storage / memory requirements of the machine used to train the model (locally vs cloud training, and dependent on the size of the data to train the model, as well as the size of the model itself).

There is a trade-off between the computational cost of training complex models and performance improvements. For example, if a 5% reduction in your error metric (such as mean absolute error) means we can increase our accuracy by ~100MWs, then the model training cost is minimal in comparison to the downstream value it creates.

Understanding this trade-off is the job of data scientists in conversation with their end-users, to understand what level of performance is suitable for the end use case, and how much model iteration should be done to balance the training cost with a sufficient level of accuracy.

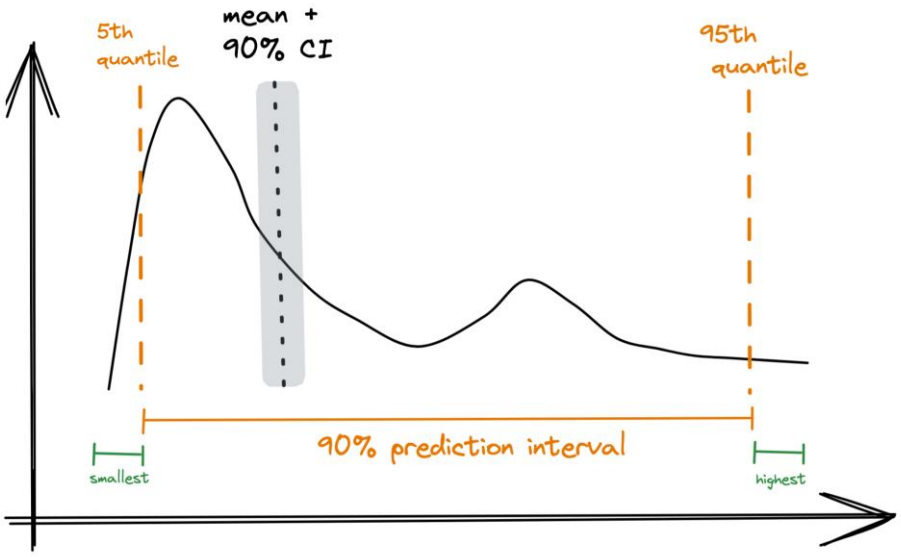
Typically, there is lots of experimentation at the start and once a suitable model architecture is identified, it is retrained on the latest data by “freezing” the desired model architecture and hyperparameters. If the model drifts from its desired performance, further iteration of the model may be required (which increases computational cost, but will result in a more accurate model).

5.2 Probabilistic load forecasts

Load forecasts can broadly be thought of as deterministic or probabilistic. Thus far, this literature review has focussed on point load forecasts. A point load forecast is one that forecasts a single value for each settlement period, often estimating the average as discussed above. Probabilistic load forecasts (PLFs) instead provide multiple estimates for each settlement period and, in doing so, capture some of the uncertainty associated with the estimate.

PLFs can be in the form of *confidence intervals* or *prediction intervals*. Confidence intervals describe the range of values the *average* of a variable can take while prediction intervals describe the range of values the variable itself can take (Figure 2). The type of model we choose will depend on if we care about predicting the average with a certain degree of confidence (and so choose confidence intervals) or if we want to understand the distribution of potential outcomes to quantify risk (and so choose prediction intervals).

Figure 2: depicts confidence interval (shaded grey) and prediction interval. For the energy-specific use case, the x-axis would refer to kWh (at the household level) or MWh (at a higher level of aggregation).



5.3 Hierarchical Forecasting

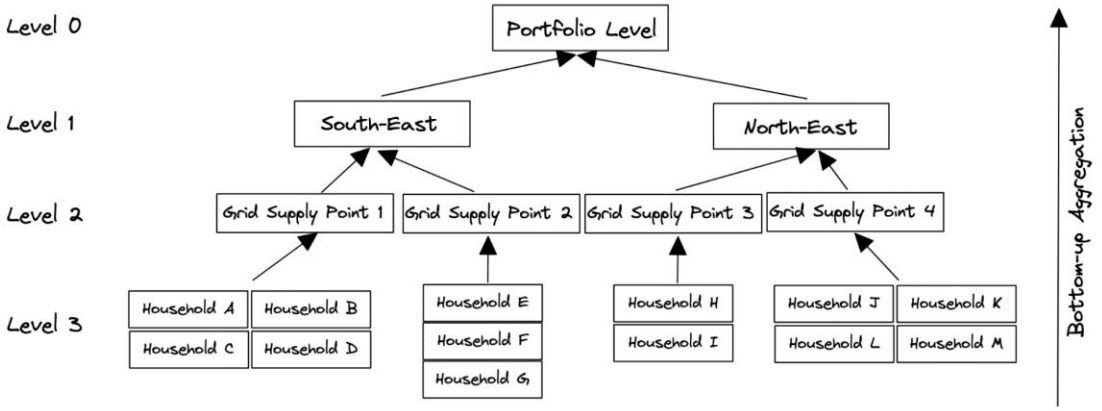
Hierarchical forecasting is the term used for when we need to forecast multiple series that depend on each other according to a hierarchy. An example of such a series is smart meter data, where the lowest level is household level, which can be aggregated up to regional level and finally to portfolio level.

In the context of domestic flexibility, this hierarchical forecasting is helpful as the same set of forecasts can be used in different systems (e.g., at GSP level, vs GSP Group level, vs national level). By forecasting at higher levels in the hierarchy, we can protect household confidentiality and ensure the protection of personal data (since smart meter data is considered personal data under the Data Protection Act and GDPR).[5] The literature indicates that forecasts at a higher aggregation level tend to be more accurate, since the consumption profile is more predictable.

The household level forecast is crucial for Flexibility Service Providers to enact settlement, but might not be required by ESO, for example.

[5] "Regulation (EU) 2016/679 of the European Parliament and of the Council", Legislation GOV.UK. https://www.legislation.gov.uk/eur/2016/679/article/4,(accessed Jan. 30, 2023)

Figure 3: An example of aggregation levels for the energy system, from the household up to the "portfolio" of a Flexibility Service Provider (FSP)



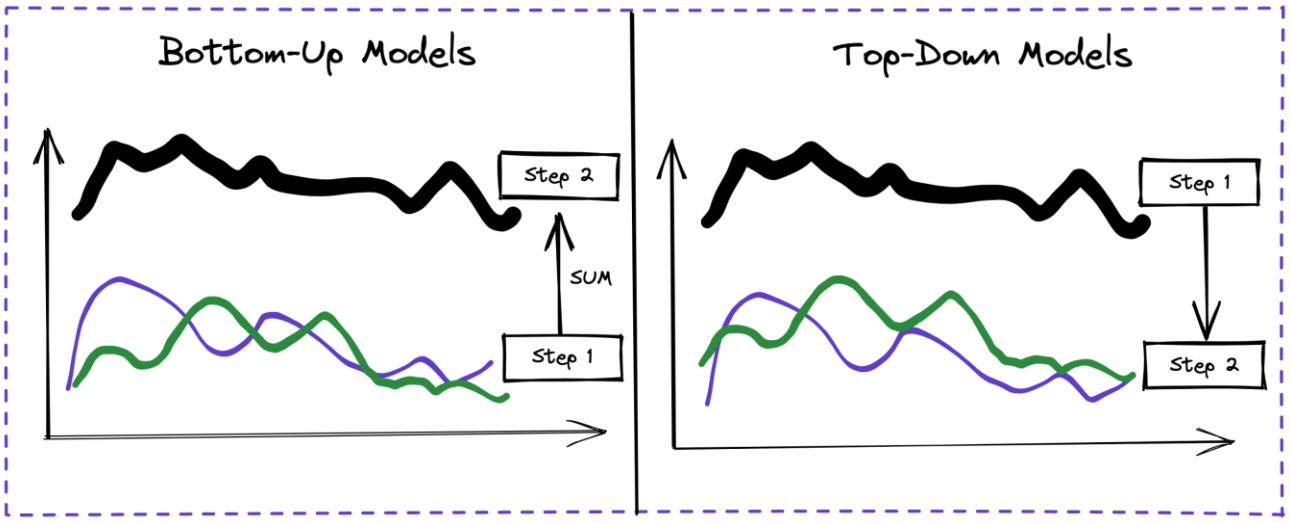
5.4 Bottom-up vs Top-down approach

Traditionally, hierarchical forecasting is achieved via a bottom-up or top-down approach:

- **Bottom-up**: Produce individual forecasts at the "lowest level" of the hierarchy and aggregate up.
- **Top-down**: Produce forecasts at a top-level and apply a 'proportion factor' or 'weightings' to obtain lower-level disaggregation.

Concretely, in the context of CrowdFlex, bottom-up models would forecast each "unit" (e.g., a household, or an asset such as an EV/HP) and then sum up to the portfolio level whereas top-down models would forecast the portfolio and then disaggregate into individual "units" (such as a unique household, or unique asset).

Figure 4: Graphical representation of Bottom-Up vs Top-Down forecasting models



In both of these methods, information only flows one-way. Bottom-up forecasts will be oblivious of top-level trends and vice versa. Literature from the supply chain sector, where these types of forecasting techniques are commonly used in stock forecasting, suggest

bottom-up forecasting produces a more accurate approach compared to top-down methods[6]. But the specific technique will depend on:

- The variability of the lowest level profile.
- Your use case (i.e., do you need the lowest level profile, or just the aggregated level profile?).
- Data and technology requirements (e.g., it might only be computationally practical to forecast higher levels if there are too many subcomponents at the lower level).

5.5 Aggregation of Hierarchical Forecasts

Hierarchical forecasting is straightforward for deterministic forecasts (simply sum the component forecasts) but is a lot more challenging for probabilistic forecasts.

We cannot simply obtain the 95th-percentile forecast at an aggregated level by aggregating the 95th-percentile forecasts of the lower levels. That would assume that all households are consuming at their 95th-percentile simultaneously, which is not realistic. There is literature on how to overcome this problem, but the methods are relatively complex and deferred to the Appendix.

[6] B.J. Dangerfield and J.S. Morris, “Top-down or bottom-up: Aggregate versus disaggregate extrapolations,” International journal of forecasting, vol 8(2), pp 233-241, 1992.

6. Model specification

This model specification was co-created through structured interviews with ESO teams covering:

- System needs definition (current, and future);
- Energy Management and Strategy;
- Data Science and Forecasting;
- Innovation and VirtualES

as well as through informal conversations and structured review processes with the CrowdFlex consortium partners.

The model specification outlines what the model would achieve and how it would be used, model inputs/outputs and recommended modelling techniques.

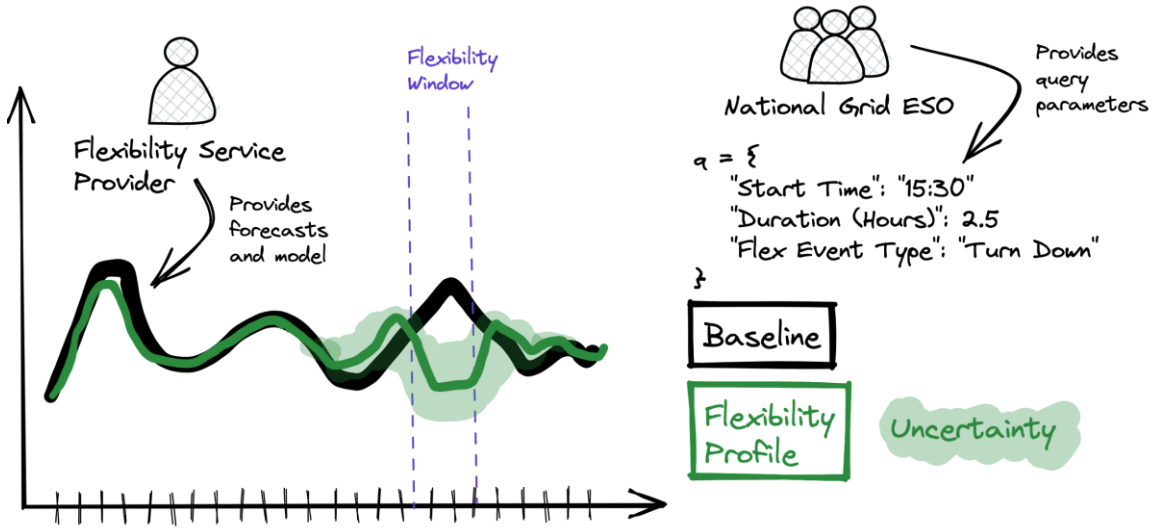
The reader should note that this modelling work differs from the analyses intended to be run post-trial events (after each batch, as outlined in CrowdFlex deliverable D7.1 - Trial Specification). After trials, we will perform data analysis to understand the correlations between, and causal factors for, domestic flexibility. The output will be various graphs that summarise the key learnings. The goal and objective covered by this Model Specification document is the deployment of models that can enable better grid operations through tools such as dashboards, or automated deployment of domestic flexibility post-CrowdFlex.

6.1 A potential vision for integrating with the VirtualES

The following diagram presents a potential vision for the domestic demand and flexibility components of VirtualES; specifically:

- Flexibility Services Providers (FSPs) will own their own data and models.
- ESO will query these providers, which will provide outputs in a consistent format (called a *schema*), via an API.
- ESO will “pull” in this data to use for operational purposes.
- The same “portal” or set of APIs can be used by other end users for other use cases, e.g., by Distribution System Operators (DSOs), for actioning events, delivery and settlement/reconciliation, etc.

Figure 5: A potential "frontend" for the VirtualES model from Flexibility Service Providers. The x axis denotes half-hours, and the y axis denotes MWh. The uncertainty band is exaggerated for visualisation purposes.



In this approach, developing a standard communication protocol (or schema) for the inputs and outputs is crucial. Instead of ESO operating another portal to which each supplier would need to be onboarded, they can instead work on implementing a schema (that may update as grid needs evolve) which each provider's systems conform to, if they wish to participate in the relevant flexibility services.

Each provider can then be self-sufficient in developing their own technical infrastructure to integrate with these services, as well as adding value to their own operations through greater visibility of their own assets (e.g., through dashboards). CrowdFlex will also investigate potential bidirectional data access via the VirtualES. That is, in return for FSPs building the APIs according to a common schema for ESO/DSOs, they can ingest relevant data sources via the VirtualES that can improve their own operations or the value of the service they provide.

The ultimate vision would be for domestic flexibility to be integrated as a resource alongside other services, with ESO (and others) making use of near real-time data streams from various suppliers to make automated cost-optimal decisions for grid operation. This could include using linear programming type algorithms over a range of possible flexibility windows and services to determine which one would meet grid needs at the lowest cost while also benefiting energy consumers. A key enabler of this would be a queryable, provider-led set of models that are:

- Quick and easy to query (e.g., over a simple API, with an API key);
- Standardised (e.g., data inputs and outputs conform to a common standard schema);
- Reliable (i.e., data updates regularly);
- Accurate (i.e., forecasts are within an acceptable tolerance of error).

This view would provide two immediate benefits:

- Greater visibility of demand (the "baseline");
- A queryable way to interrogate different scenarios of response outturn, helpful for a range of purposes, including planning, forecasting and management.

6.2 Different models for different services

Defining a single model to cover the range of trial services in CrowdFlex is not possible. Consider the simple delineation between two example types of services:

1. **Response services**: these would be circumstantial, event-based services with hours- or day-ahead scale notifications. An example of this type of service would be Constraint Management services (ESO services), as defined in CrowdFlex Deliverable D4.1.
2. **Scheduled services**: these would be repeated services (e.g., turning down in a specific area between 4-7pm every weekday in a given season to manage a known / planned issue). An example of this type of service would be Sustain-H / LMA-Secure (proposed DSO services), as defined in CrowdFlex Deliverable D4.1.

Note that participation in the Balancing Mechanism straddles both of these – utilisation of the asset would be a response, but for the purposes of CrowdFlex the consumer's interaction would be scheduled (e.g., plugging in an EV at agreed times or modifying heat pump behaviours on a particular day).

To predict what a customer would have done on a given day (their counterfactual), we want to use the most appropriate data. In case 1, we would want to use historical data about that customer up until the day before the flexibility event is run. In case 2, we can only use historical data about the customer *before they took part in the scheduled service*, which may be from a different year if it is a seasonal service. This is because engaging in the scheduled service every day changes their consumption, and so we would want to reward them based on changing their consumption pattern from a different time period.

In each of these two (simple) cases, the demand prediction part of the model would have to be *architected* differently. While two separate models could be constructed to address each of these use cases separately, we focus in this report on **developing a model to predict the flexibility for response services**. This is due to the recommendation of CrowdFlex Deliverable D8.1, which recommended that the ESO area-based Constraint Management response service was likely to be the most high-value service for CrowdFlex to target.

A near-real time and constantly updated demand forecasting model could also be used for various different DSO and ESO (response) services for a range of services, and thus we focus on this use case. The techniques described could apply to scheduled services, but would need to be adjusted to make sure the models are "trained" on the correct time period to fully reward customers for the flexibility they are providing, on the service that they are providing it on, incurring a significant additional cost if we aimed to deliver these in the next phase of the project as well.

In the future, households within the same geographical area will be able to participate in different flexibility services and "stack" the rewards. This becomes complicated if two different baselines are used, but we will also trial the stacking of two services in CrowdFlex.

6.3 Probabilistic vs Deterministic Hierarchical models

Smart meter data is intrinsically geographically hierarchical with the following different aggregation levels:

- **"Lowest" level**: household-level forecasts where we produce a prediction for each individual household.

- **Intermediate levels**: this would be an aggregation of all households' forecast on to an intermediate level e.g., sub-station, GSP, region, etc.
- **"Highest" level**: this would be an aggregation of all households' forecasts into a highest (portfolio) level forecast for that FSP. Examples may be *all EVs a chargepoint operator owns* or *all households for a given supplier*. In both scenarios, we are assuming we are restricting the aggregation to only those assets where there is an agreement in place to engage in flexibility services. We discuss disambiguation of load in the next section.

6.3.1 Deterministic Hierarchical Models

Deterministic models are models that produce a "single value" output. For time-series models such as smart meter data, this means that each time unit (half-hour, in the case of smart meter data) will contain a single value for the prediction.

Another important aspect of deterministic models is *idempotency*. This means that running the same model twice on the same set of inputs produces the same output.

Examples of deterministic models include rule-based methods (such as averaging historical consumption values to predict consumption at a time unit in the future) and machine-learning based methods such as linear regression, XGBoost and Neural Networks. Deterministic models for time series are also sometimes referred to as "point estimate" forecasters, since they predict a single value for a given time step.

Aggregating deterministic models is trivial - this can be done by simply summing the forecasts for individual households to the required aggregation level.

6.3.2 Probabilistic Hierarchical Models

Probabilistic models are models that produce a distribution of outputs (e.g., quantiles that define a probability distribution).

Aggregating probabilistic models is non-trivial. Obtaining the 95th percentile on an aggregated level by simply summing household-level 95th percentile will be grossly overestimating as that assumes all households are consuming at their 95th percentile simultaneously, which is not realistic. There are some methods which look at this aggregation, described in the literature review in section 9.10 (Aggregation of Hierarchical Forecasts).

6.4 Modelling choices to be made

For this use case, we need a hierarchical model. The previous section, as in the Literature review, notes that we need to make model choices about which parts of the model are deterministic, and which parts are probabilistic.

The model architecture must be a composition of models with the following choices:

- (Ensemble of) Deterministic model(s)
- Probabilistic model(s)

This choice will depend on the "level" at which we forecast, the aggregation we want to do (top-down vs bottom-up) and how the outputs will be used. We must also decide how to construct the prediction intervals or confidence intervals, depending on the use case.

6.5 Disambiguation of load

For domestic flexibility, the market will operate such that asset operators (e.g., chargepoint operators) and suppliers will be able to bid in kW's of flexibility from their assets (the chargepoint, and household, respectively).

In the case where Household A is with a supplier B and has an EV with chargepoint operator C, there is the possibility of "double counting" the kW's of flexibility for a certain event. Assume that the flexibility event is a turn-up event, and the EV shifts its charging to that window, but the household also decides to turn on other white goods (such as the dishwasher and washing machine). In that case:

- Supplier B will receive the smart meter data from the whole household, and relative to the baseline they will measure an increase in demand from the EV, dishwasher and washing machine.
- Chargepoint operator C will receive the asset data from the chargepoint, and relative to the baseline they will measure an increase in demand from the EV

In this case, if the household is participating in the flexibility event with both their supplier and chargepoint operator, the EV demand will be "double counted".

Disambiguation is possible technically; if a record was kept of the MPAN and asset numbers in each household then we could disaggregate the two profiles (simply by subtracting one from the other), and these techniques could scale to GSP or GSP Group level. However, there are complicated questions of governance surrounding who claims the flexibility. The provider claiming the flexibility should be the one that the customer holds the relationship with, and this provider would send email/text/app notifications (if manual) or automated control signals (if automated).

We hope to outline the challenges and blockers to technical deployment in the CrowdFlex deliverables, through real-world learnings that add to the evidence base of domestic flexibility in practice.

6.6 Data availability for modelling

The implicit assumption in most of the literature is that the data is available at the time of the forecast. In practice, there can be errors in data pipelines and ingestions that mean data is not up to date, or errors that need to be corrected to ensure it is of the correct input format to make predictions on.

For CrowdFlex, each Flexibility Service Provider (FSP) will collect data directly from the asset (e.g., chargepoint, or heat pump) or from the smart meter relating to the household. The rate at which this data becomes available depends on the source of data; connecting to "live" assets usually involves a data latency of seconds/minutes, whereas getting smart meter data can take approximately one day. The forecasts should be run on the latest data, but the latest data might involve a "lag".

In addition, the data accuracy and quality will vary by source. Each FSP should work to ensure that the quality of data is as accurate as possible (e.g., outliers are removed and interpolated, and consistently inaccurate data flows are remedied). The recommended way to fill in missing values from time series data is either by "forward filling" the data (i.e., persisting the most recently received data value forward in time) or by linear interpolation. "Forward filling" the data cannot be done over large data gaps, and so it is recommended that for each half hour there is at least one reading from the asset. It is also recommended

that missing values are not filled for more than 2 consecutive readings (i.e., 1 hour). This recommendation might change during CrowdFlex, or be refined by asset type, if it is determined that more automated devices should have data received more accurately than this.

To infill missing values, older data for the same asset (or household) should be preferred in favour of imputing based on the CrowdFlex trial participant average, when used to calculate the baseline.

Assets with persistent errors may have to be temporarily opted out of the trial until the data errors are resolved.

6.7 Baseline demand model

By the "baseline demand model" we are referring to the forecast of half-hourly electricity consumption for a (set of) household(s). As mentioned in the previous sections, we need to choose whether this model will be:

- Deterministic or probabilistic.
- Top down (portfolio → households) or bottom up (households → portfolio).

This will depend on the use case.

6.7.1 Predicting participation in events

CrowdFlex deliverable D7.1 recommends an opt-in to trial, but opt-out of event structure. This means the participation rates are unobservable, so we cannot explicitly forecast the "number of participations in an event" like we might be able to if households had to explicitly respond to a notification indicating that they will take part in the event. For this reason, we cannot predict the "opt-in percentage" per event. Instead, the forecasts will forecast the demand, and flexibility, of all households who are in the CrowdFlex trial pool that have not opted out of the trial.

For the models below, they will predict over the population of assets that have opted into the trial overall. If participants are onboarded in batches, then the model(s) should be updated to reflect a larger pool of participants for each event.

6.7.2 Use cases for the baseline model

As part of CrowdFlex:Alpha, CNZ engaged in Subject Matter Expert (SME) expert interviews with ESO, as well as informal conversations with suppliers and FSPs. A range of grid stakeholders (ESO and DSOs) will be the "end users" of this model and be able to ingest near real-time data about demand to improve their visibility.

In this section we will refer to a "unit" in a "portfolio". Explicitly, we mean a unit is a household for a supplier, or a unit is an asset (such as a chargepoint) for a FSP. In each case, we are restricting the modelling to the households or assets willing to engage in flexibility (opting in).

The baseline model will have two main use cases:

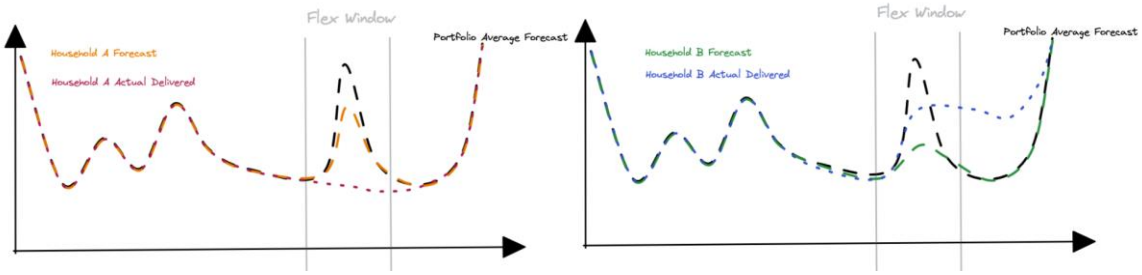
1. Providing a view (to ESO, DSOs and to FSPs) of what the expected forecast demand is across a given time interval (e.g., 1 day).
    a. This can be used to improve visibility, and be used as an input to other systems.

   b. It will also be used as an input in the flexibility model, which will be conditional on the forecasted demand at a given time (as well as other factors about the flexibility event - see section 6.9 (Flexibility model).
2. Settlement.
   a. From ESO / DSO → FSPs, where settlement will be at the portfolio level for total MWh delivered (per region) relative to the baseline.
   b. From FSPs → consumers, where settlement will be at the unit level (e.g., household, or chargepoint) for the total kWh delivered relative to the baseline.

Given these two use cases, we must decide on the model structure (deterministic vs probabilistic, top-down vs bottom-up). We first start with an observation.

A top-down baselining model using a 'portfolio average' to work out compensation could lead to unfair outcomes

Figure 6: Showing two different households (A, left and B, right) where one turns down relative to its forecast but the other turns up relative to its forecast. They "perfectly balance" each other out, so the portfolio average drops. This shows a misalignment of incentives if we were to use the portfolio average.



Every household consumes energy differently. If we used a top-down model to baseline consumption at the household level, we would still need to settle at the household level. One approach might be to take the "portfolio average" turn down/up and measure all households relative to this. Smarter methodologies could weight this based on previous consumption patterns, but it would still be a weighted disaggregation of the total baseline demand.

While portfolio averages would capture the 'average trend' across the portfolio, they may not capture idiosyncrasies at the household level. In Figure 6 above, both households were consuming below the portfolio average during the Flex window and therefore would be eligible to be paid if compensation was calculated using portfolio average.

Household B, however, actually 'turned up' their consumption (compared to their own 'baseline'). This would lead to an unfair situation where some households are compensated while actually turning up at the expense of other households that turned down.

To avoid unfair scenarios like this, we should be calculating baselines on an individual household level, and work out the compensation using the household's individual baseline.

In addition, it is unclear whether being "measured relative to others" would introduce "social competition" effects whereby household A tries to out-compete other households in the same group (and thus having the desired effect) or whether households would not be directionally aligned with the objective (i.e., "other people will participate, so I don't have to") leading to undesirable outcomes. It is simpler to have a household be measured relative to their own (historical) usage, both in terms of fairness and explainability.

6.8 Recommended Baseline model

We propose the baseline demand model be produced deterministically at the household level for the following reasons:

- To use this as a method for calculating unit-level compensation, we would need a unit-level forecast. Using portfolio averages to work out compensation is not ideal as this would disincentive assets from partaking, and in some situations actually be rewarding assets for 'turning up' when the signal was to 'turn down' (and vice versa).
- Because of the need to aggregate to a higher aggregation level for input into the Flexibility model, a deterministic model is preferred due to the challenges of aggregating probabilistic forecasts.

This deterministic model would then be "summed" to get the deterministic baseline forecasted consumption in each GSP Group (and could be for the entire portfolio, if needed), which will be used for portfolio-level settlement and in the Flexibility Model.

This model will output the predicted demand for each half-hour for each unit that has opted into the CrowdFlex trial.

For the baselining methodology, we would propose a simple linear regression model for the following reasons:

1. Whilst rule-based algorithms are simple to implement, it is difficult to take into consideration other variables, such as weather, into account. Linear regression can account for this.
2. Linear regression is well-understood, easy to implement, and its predictions are highly explainable (e.g., to customers). This also meets the requirement that the model could be easily implemented by various flexibility service providers, since the form of the model is simple and explicit.

Other, more accurate, models such as Boosting methods or Neural Networks are less desirable in this case due to explainability for trial participants, but could be experimented with in CrowdFlex.

We recommend that the following features are included in the unit-level linear-regression model:

- Lagged consumption features (from previous days, and weeks, up to 1 month ago).
- Forecasted weather features.

Crucially, **this baseline should exclude previous event days**.

This can then be summed by GSP Group. Our recommendation is that all Flexibility Service Providers (FSPs) engaging in CrowdFlex seek to build a model as described above, which is updated at different temporal intervals depending on the nature of the data:

- Energy suppliers - update daily (in line with when new smart meter data is received).
- Asset operators - update more frequently, perhaps hourly, based on near real-time data from electric vehicles or heat pumps.

We believe this general view of demand will be useful for a range of trial services: DSOs could work at GSP level and ESO can aggregate to GSP Group level, for example. This general view of demand will increase visibility for various grid needs, and be a useful precursor towards engaging in many different (response) grid services.

It is important that this baseline is calculated in a consistent way across FSPs, and output in a consistent format, such that it can be aggregated and compared in the future.

6.8.1 Disaggregating the baseline by asset class

In discussion with ESO data teams, it was mentioned that this baseline would be even more helpful if it were split out by asset class (e.g., household, EV, Solar, Battery, Heat). This is trivially possible based on the provider. However, splitting the "household" load into explicit demand from low carbon technologies (LCTs) within that household would be harder; it would require suppliers to keep up-to-date records of LCT ownership in each household participating in flexibility, and then to disaggregate the meter load into sub-assets. It becomes especially harder when households own multiple LCTs.

While this might be a barrier to general adoption, CrowdFlex will trial splitting this demand by asset class, since CrowdFlex will include a pre-trial survey which collects the information necessary to trial this approach.

6.8.2 Calculating a baseline for scheduled services

As mentioned previously in this report, **we are prioritising modelling response-based services in CrowdFlex**. However, for completeness, this section gives a brief overview of how baselines for scheduled services could be calculated.

CrowdFlex Deliverable D4.1 recommended that the Balancing Mechanism ESO service, and Sustain-H and LMA-Secure DSO services were potential services for domestic flexibility. CrowdFlex Deliverable D8.1 noted the value of automated assets participating in the Balancing Mechanism. The DSO services will have a smaller number of CrowdFlex trial participants due to the nature of them being restricted to a specific area, but could be of value.

In the scheduled services use case, we recommend trialling a different baselining methodology for the reasons discussed in section 6.2 (Different models for different services). For the DSO services, we propose using the modelling method described above, but instead of forecasting tomorrow based on today's data - like for routine services - we propose that the forecasting methodology for that household uses 1-3 weeks of data for a given season when that household is not opted-in to that DSO service. This 1-3 weeks of seasonal data will help to establish a baseline period which can be used to compare the repeated flexibility against. For the Balancing Mechanism, the P376 Methodology with in-day adjustment has been proposed.[7]

We continue to propose that the number of assets in each GSP / GSP Group be made available for grid stakeholders, to improve visibility of assets participating in various grid services.

FSPs may wish to have a rolling-average calculation for the expected returns under participation in the BM, which will be identified in CrowdFlex, to refine the offering of the availability payment proposed to customers. In CrowdFlex trials, availability payments will be randomised to enable methodical testing of customer price sensitivity.

A different approach to baselining for scheduled services that could be trialled in CrowdFlex is to use a group of customers with similar consumption patterns to the group taking part in

[7] "Utilising a Baselining Methodology to set Physical Notifications for Settlement of Applicable Balancing Services", Elexon. https://www.elexon.co.uk/mod-proposal/p376 (accessed Jan. 30, 2023)

the scheduled service but who are not opted in to the scheduled service. Their baselines can then be forecasted day ahead and aggregated, and used as a proxy for those taking part in the scheduled service. This could be used to help correct for weather patterns.

Stacking these two services, with their two different baselines, may be complicated from a modelling perspective. In the later stages of CrowdFlex, small scale trials that mix the stacking of routine and scheduled services simultaneously (rather than routine-only or scheduled-only) could give insights into how to best build models for this use case.

6.9 Flexibility model

In this section, we explain how the output of the baseline model can be fed into a model that calculates the half-hourly flexibility outturn of a given collection of assets for response-based events (those with hours- to day-ahead notice periods). We make a recommendation on the functional form of this model that balances the use of probabilistic information (which is noted by CrowdFlex deliverable D1.2 to provide significant value to system operations) and manageable geospatial aggregation.

6.9.1 Use cases for the flexibility model

The purpose of the flexibility model is for users to evaluate if a flexibility event would be a viable lever to balance the grid on both planning/scheduling and near real-time (control room) timescales.

Take the Constraint Management use case as identified by CrowdFlex Deliverable D4.1: in situations where the capacity between area A and area B is constrained (for example: high wind is forecast in area A, but demand is low, and there is a constraint in moving the energy to area B where demand is high). Currently, ESO would use constraint management services to pay for wind curtailment in area A, and pay for (fossil-fuel-based / dispatchable) generation in area B. Instead, the flexibility model could be used to input the time where the constraint is forecasted and simulate a turn-up event in area A with a simultaneous turn-down event in area B to match the demand with the generation and constraint profiles. This avoids our need to pay for wind curtailment and gas generation, and instead pay for domestic flexibility.

Based on our information gathering, we have assumed in our use cases that:

1. ESO makes decisions, and would therefore like the flexibility model forecast, at a GSP Group level; DSOs would also benefit from visibility of the same dataset and decision making.
2. ESO and DSOs would benefit from improved visibility from the demand and asset side at a GSP and GSP Group level.
3. All parties want to make risk-based assessments of domestic flexibility, by looking at probabilities such as “1 in 365 days” (i.e., 99.7%). A prediction interval (and a model that outputs the quantiles of the distribution of flexibility) is therefore required, rather than a confidence interval (for the average amount of flexibility we would expect).
4. Closer to real-time (e.g., in the control room), ESO and DSOs value accuracy/ precision a lot more (preferring deterministic forecasts, or probabilistic forecasts with tight confidence/prediction intervals).
5. Further from real-time (e.g., many hours ahead planning), ESO could use probabilistic forecasts to make risk-based decisions to balance the grid while also minimising cost.

6.9.2 Recommendations for Flexibility Model

For the flexibility model, we propose a simple quantile linear regression model at the GSP Group level. This balances:

- The data volumes needed to collect to have accurate quantile / probabilistic models;
- Hierarchical aggregation of the model.

6.9.2.1 Proposed input of Flexibility model: Aggregated deterministic demand forecast

1. Baseline demand forecast aggregated up to a desired aggregation level (e.g., GSP, region or nation). We expect to aggregate this to GSP Group level in CrowdFlex.
2. Information about the specification of the flexibility event intended to be run e.g., notice period (hours), incentive level (£/kWh), duration (hours), start time, etc.
    a. For each half hour, there should be a variable which is the relative position of that half-hour in the flexibility event.
    b. This could include one-hot encodings of the flexibility event times into "categories" (Overnight, morning, afternoon, evening) - based on the trial design in CrowdFlex deliverable D7.1. Note this is not a one-hot encoding for each potential flexibility time, just whether the start time falls in a specific one of these windows.
3. While the baseline model will also depend on the weather, we propose that weather inputs are also experimented with for the Flexibility model in CrowdFlex, since understanding the flexibility outturn on "particularly cold" or "particularly hot" days is a worthwhile understanding for the deployment of domestic flexibility. We will perform an analysis after the trial events to understand the relationship of weather on flexibility outturn.

6.9.2.2 Proposed output of Flexibility model: Aggregated (GSP Group) probabilistic forecast

1. The model would return the percentiles of the "updated demand distribution" conditional on the flexibility event. These percentiles could be customised based on grid needs quite easily, but our recommendation is to default to using 20-quantiles (also called ventiles).
2. Selecting just a given percentile of this distribution will collapse the probabilistic forecast into a deterministic one.
3. As ESO would be making decisions at an aggregated level (in addition to the challenge of doing bottom-up aggregation of probabilistic forecast) - we propose the outputs be at an aggregated level (at the GSP Group level) instead of household level as they would be irrelevant.
    a. DSOs may also find a forecasted output at GSP level helpful for their services in addition to the demand data. The same modelling technique can be done at GSP level, but owned by the DSO instead of the ESO.

The quantile regression model will output quantiles for each half-hour of the day, both during and outside the flexibility event window. Since the model will learn from collected data from the trials, we estimate that after each event there will be approximately 700 data points from which the model can learn. This will enable it to better construct quantiles.

When applied to flexibility, the confidence interval captures the average demand flexibility, and how confident we are in that response profile, whereas the prediction interval captures

both the full distribution of demand response following a flexibility request. We propose choosing a quantile regression instead of using confidence intervals since we want to understand the distribution of demand response. Existing probabilistic models used by ESO for other use cases use prediction intervals to quantify risk. In the (domestic) flexibility setting, we will need to output prediction intervals as well to represent the distribution of potential flexibility volume delivery and quantify the risk of delivering at different volumes.

6.9.3 Flexibility modelling assumptions

In the recommendations from the previous section, we have made some assumptions, which can be corrected in the model iteration stage if needed/desired. We assume that the model depends on:

- The baseline electricity consumption (MWh)
- The “event window time category” (overnight, morning, afternoon, evening)
- External variables (e.g., weather)
- Calendar variables (e.g., season, day of week)
- Flexibility event type (Turn Up vs Turn Down)

We implicitly depend on the number of assets opted in to the trial, through the baseline calculation (which is summed over the individual unit level baseline calculations). Therefore, it does not need to be explicitly modelled.

Note that this model does not depend on the half hour, nor on the GSP. That means:

- Holding all other variables constant (e.g., baseline, weather, …) then for a given “event window time category” (e.g., overnight, morning, afternoon, evening) there is “translational invariance” in flexibility outturn:
  - e.g., the model will deviate from the baseline for an event starting at 5:30 in the same way it would for the same baseline at 6:00.
- Holding all other variables constant, the flexibility outturn and deviation from forecasted demand is “the same” in each GSP Group.
  - This could be corrected by 1 hot encoding GSP Groups, or groups of GSP Groups (e.g., Scotland vs England) but would reduce the dimensionality of data-per-trial from which to learn.

6.10 Model inputs and parameter range

The suitable ranges for each parameter will be determined by CrowdFlex deliverable D7.1, the trial protocol. The model can only learn from the data collected during the trial, and so the range is determined by the set of values we are running in each of our trial cells.

Table 2: Example of queryable parameters for the flexibility model. These can be changed to give different forecasts. The forecast model will depend on these features, as well as other features, to forecast its prediction of how the flexibility profile will differ from the baseline under the specified event.

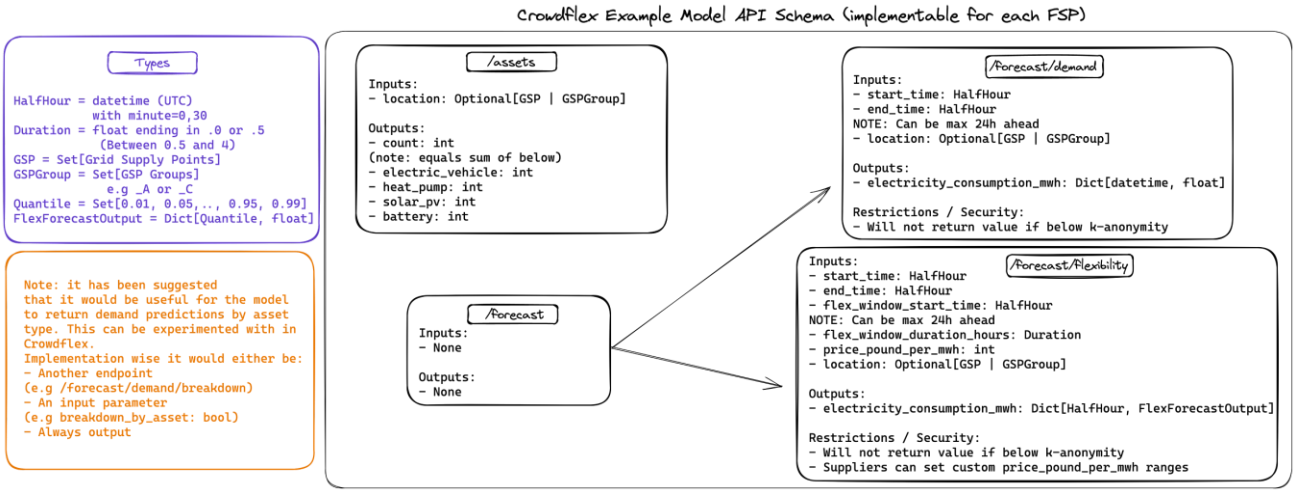
Parameter	Range	Reason
Start time	{00:00, … , 23:30} - a half hour of the day	The GB market operates half-hourly, and flexibility response will be settled half-hourly for domestic households with a smart meter. Other assets may measure sub-half-hourly, but can be resampled to half-hourly.
Duration	[0.5, 4] - Number of hours	Flexibility events will not run for longer than 4 hours during the CrowdFlex trials.
Flexibility event type	{Turn Up, Turn Down}	Both turn up and turn down events will be run, with response being asymmetric.
Location	A Grid Supply Point Area (either GSP or GSP Group)	Some flexibility events will be run in different selected locations only. This input essentially acts as a "filter" to only baseline and model the flexibility potential of assets/households in a specific location opted in to CrowdFlex (and excludes the rest). If location is not set, the entire population will be modelled.
Price	£0/kWh - £1.25/kWh (£0/MWh - £1,250/MWh)	The flexibility response will depend on the price offered, and this will be a known parameter from the end-user who is reviewing the cost stacks of other services. These values are defined as the "utilisation payments" in CrowdFlex Deliverable D7.1. The model can only accept values in the range it was trained on, which for us comes from the prices offered during the trials.
Notice Period	[1, 24] - Number of hours	Should be greater than 1 hour

To check the model offline, it may be useful to predict subgroups of customers such as:

- Opted in to CrowdFlex vs defaulted in.
- Number of events in the past month customers have been invited to.
- Number of events (ever) that they have been invited to.
- Subgroups of customers as identified in WP2.

6.11 Example API Schema

Figure 7: Example API schema showing total assets of a flexibility service provider, and various forecasts (both the baseline demand forecast and the flexibility forecast). The schema shows various inputs and outputs to each "endpoint".



The broad purpose of the schema as defined is to create a common standard of reporting. We propose a REST API, where ESO can GET data from different FSP endpoints and POST different requests to the flexibility endpoint.

- **/assets** - improves ESO/DSO visibility into "dispatchable" assets from a range of FSPs.
- **/demand** - improves ESO/DSO visibility into aggregated forecasted electricity consumption at GSP (Group) level.
- **/flexibility** - enables ESO to calculate how the portfolio of assets would respond to flexibility requests, so that the grid can plan better to dispatch these assets intelligently.

6.12 Model ownership and data flows

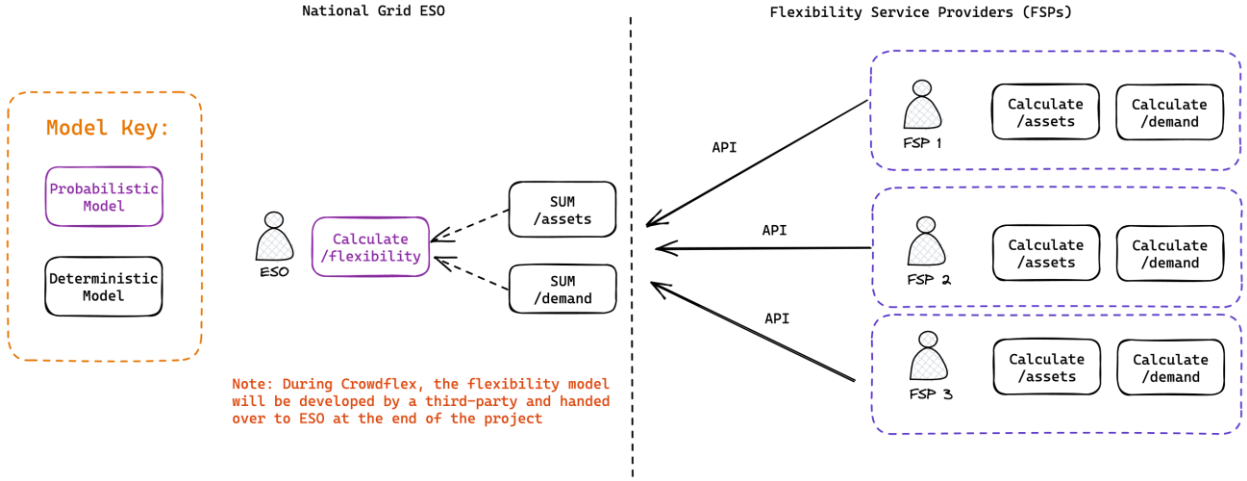
6.12.1 Option 1: ESO owns the flexibility model, FSPs own the asset and demand models

This is our preferred and recommended option. It balances FSPs being able to quickly generate new (deterministic) forecasts and improve visibility for ESO from their own systems without burdening them with the expertise of maintaining a separate flexibility forecast. In addition, as described in Option 2 below, aggregation of probabilistic forecasts is non-trivial and would be a harder flow for ESO to work with.

When the flexibility model has been run to determine the type of event that would intend to be run, there would need to be a methodology to communicate that decision to the FSPs in CrowdFlex. This will also be communicated via APIs, and each provider will be responsible for delivering a turn-up (a proportion of) the demand value they submitted.

CrowdFlex should test providers being able to provide constraints on the turn-up / turn-down they can deliver at each half-hour (which might come from a FSP-owned forecasting model) to prevent grid stakeholders requesting an amount that they do not feel is possible to deliver. These models will be FSP-dependent (e.g., how many EVs do we expect to be plugged in? or heat pumps turned on?)

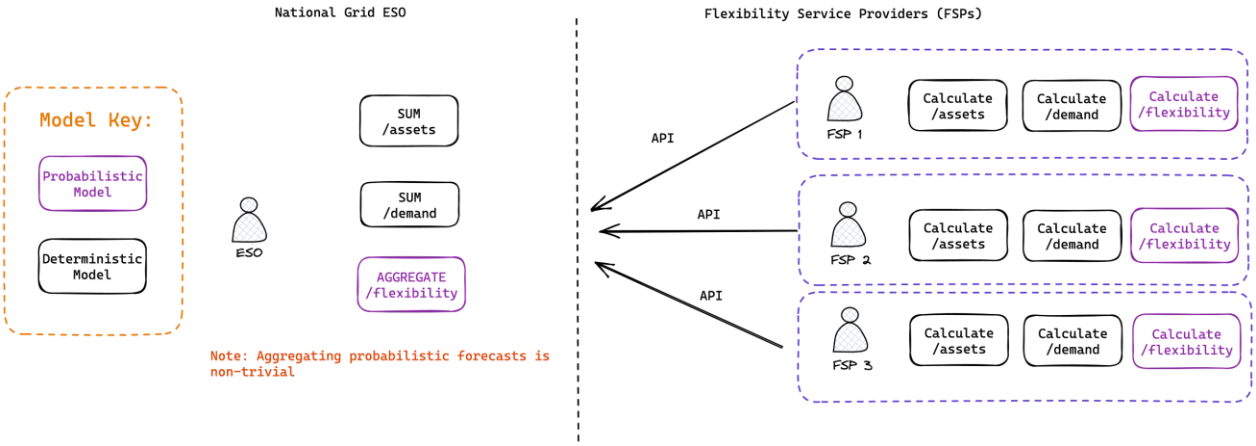
Figure 8: API flow diagram where FSPs own the asset and demand endpoints, as defined in Figure 7, whereas ESO is responsible for the aggregation and calculation of the flexibility endpoint



6.12.2 Option 2: ESO does aggregation only, FSPs own the asset, demand and flexibility models

This method is **not** our recommendation, but is proposed as a valid alternative to Option 1. In this method, each FSP would need to calculate their own probabilistic forecast which would need to be aggregated by ESO across FSPs. This is technically possible, but more complicated. The literature suggests copula-based approaches as one method to do this.

Figure 9: API flow diagram where FSPs own the asset, demand and flexibility endpoints, as defined in Figure 7, whereas ESO is responsible for the aggregation of each of these endpoints



7. Model delivery plan

7.1 Overview of Model Lifecycle

The model lifecycle can be summarised into the following steps:

1. **Data collection**, which will be via CrowdFlex beta trials (see CrowdFlex deliverable D7.1) and co-ordinated via data pipelines.
2. **Model training (building and iteration)**: trying slight variants of different models to optimise for a given objective (e.g., reducing the error of predictions, using metrics such as mean absolute error, or mean squared error).
3. **Model deployment**: Hosting the "trained" model in a place that downstream services can access it, for example on an endpoint in the cloud.
4. **Model monitoring and evaluation**: Building dashboards to track the models' performance over time, and capture any *model drift* (when performance deviates outside user-specified thresholds). This model drift should be measured against a (set of) metric(s), which are used to evaluate the model performance. This is important to get end-user buy-in for the model(s).
5. **Model retraining**: On a certain frequency (e.g., fortnightly, monthly or quarterly) retrain the model on the latest data. This could involve the same functional form of the model and just updating the data, or more general model maintenance and improvements (e.g., adding different features to the model to improve performance).

The definition of a "user-specified" threshold is one that the team of data experts can work with end-users to define. In the CrowdFlex setting, it should be based on a metric that - if improved - would give ESO confidence that the model is predicting what it is intended to do well. They can also define thresholds that represent "good" operation of the model. ESO currently works with a 99.7% - 1 in 365 days - framework. For probabilistic models, CrowdFlex will work to define the "new normal" for working with these models that balance risk with operational value created.

7.2 Team resource required

Various stages of the model life cycle are owned by different data professionals. Usually, a "squad" is assembled that consists of a product manager, a technical lead and various data professionals all working on (different parts of) the same product.[8]

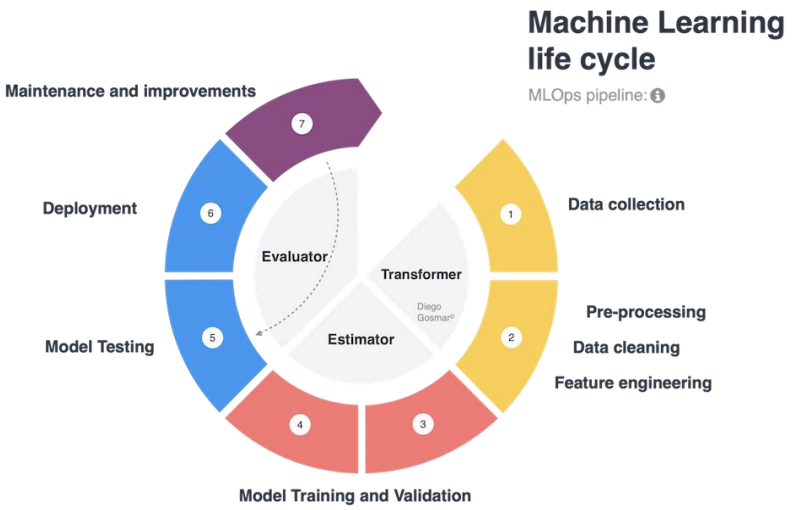
- A **product manager** ensures the right features are built in the right order, and prioritises requirements
- **Data engineers** are responsible for the data pipelines and data collection part of the model lifecycle (step 1 above)
- **Data Scientists** are responsible for the model training, iteration and retraining (steps 2 and 5 above). They are also responsible for offline and online model evaluation.
- **Machine Learning (ML) engineers** are responsible for the model deployment and monitoring (steps 3 and 4 above)
- **Tech leads** co-ordinate all data professionals and ensure the overall product technical architecture and strategy aligns with the needs of the model

[8] There are many different theories on ways to strategically deploy data products in an agile methodology that is outside the scope of this Work Package. The "squad" methodology was popularised by Spotify: "Discover the Spotify model" Atlassian. https://www.atlassian.com/agile/agile-at-scale/spotify (accessed Jan. 30, 2023)

- **(Optional, if applicable) Frontend Software Engineers**, who are responsible for building a (web) graphical user interface (GUI) application for the product

There is not a "clear boundary" between each of the 5 tasks above, but there is certainly a mix of professional skills required to enable the successful deployment of a model. In CrowdFlex, each FSP implementing their own models would need to establish a squad with the skills described above in order to work with their specific technical architecture and establish the common outputs described in 6 (Model specification).

Figure 10: An example workflow for operationalising ML deployments (MLOps) that include the key steps described above. The circular nature of the diagram reflects the iterative nature of model building and deployment.[9]



7.3 Technical Infrastructure needed

There are many different technical architectures that can all achieve the same end state. In line with the section 6.1 (A potential vision for integrating with the VirtualES), each FSP will implement their own models that conform to a common schema. That means that:

- FSPs should ingest the same inputs required to run the models described in Section 6 (Model specification)
  - Their asset data or smart meter data
  - Weather data
  - Shapefiles for geographical regions
- The outputs should conform to the commonly agreed schema (to be developed in CrowdFlex based on the Example API Schema (section 6.11).
- The technical architecture in the middle can vary by FSP, so long as inputs and outputs are the same.

In this section, we outline our recommendation for the technical infrastructure and architecture required which aligns with modern model training principles.[10]

Our recommendation is to deploy the model on the Cloud. Some FSPs may have on-premise clusters that they may wish to do (part of) the pipeline on, but for simplicity we will

[9] "MLOps scalability", Gosmar.EU. https://www.gosmar.eu/machinelearning/2021/01/02/mlops-scalability/ (accessed Jan. 30, 2023)

[10] "Machine learning operations", Microsoft. https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/ai-machine-learning-mlops (accessed Jan. 30, 2023)

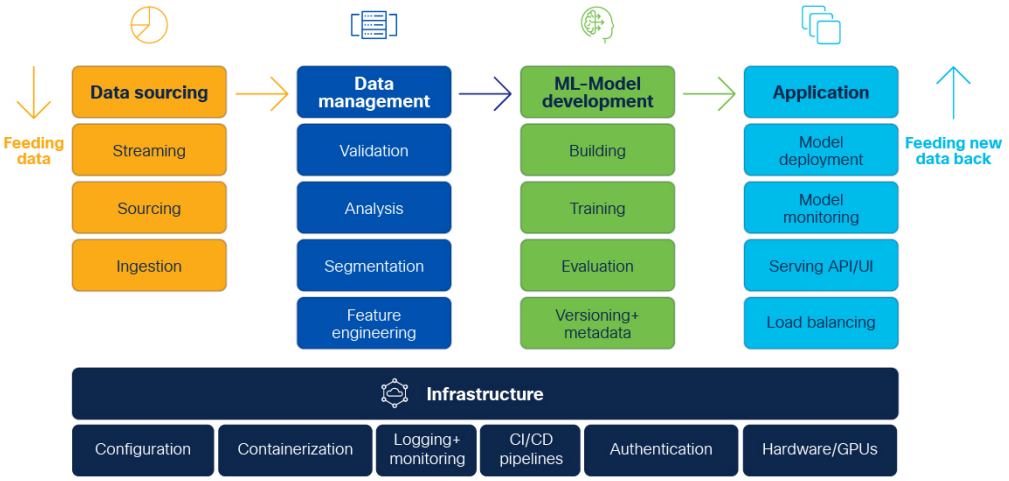
describe the components needed that align with well-established Cloud-based products. Most Cloud providers (Azure, AWS, GCP) provide similar products.

1. **Data collection**, via Extract-Transform-Load (ETL) or Extract-Load-Transform (ELT) pipelining tools. This will move data from various sources into "storage" for use in modelling.
    a. Popular tools include Airflow for orchestrating the data pipelines (which are run as DAGs) and Kubernetes for executing.
    b. We recommend a serverless architecture that spins up resources to run the data pipelines when needed.
    c. We recommend that the data is stored in cloud storage, and then loaded in to cloud SQL-like databases, where it can be transformed for the appropriate downstream uses (e.g., step 2 below).
2. **Model training**
    a. This can be local development for CrowdFlex, taking data from a data-lake or data-warehouse and using pre-established libraries to train Quantile Regression.
    b. We highly recommend Python and Jupyter Notebooks for this task, which enable rapid offline experimentation. Open-source libraries such as Pandas can be used for data manipulation, sklearn can be used to build Quantile / Linear Regression models as well as understand how they perform on the training data, and matplotlib can be used for data plotting.[11,12,13] There are also libraries for more complicated models such as hierarchical forecasting.[14]
3. **Model deployment**
    a. The model has to be deployed over an API. We recommend tools like FastAPI in Python, but many alternatives exist.[15] We recommend deploying to an endpoint that is protected by an authentication layer, so that only specified end users can access it (such as ESO). Furthermore, this endpoint should scale well with the number of requests and be "always on".
4. **Model monitoring and retraining**
    b. Model monitoring will start with ad-hoc offline model checks, and can be tested after each flexibility event.
    c. More advanced methods would include creating a dashboard that shows key model metrics (such as mean absolute error, or other metrics TBD with end users) and monitor how they change over time. It can be automated such that if this metric falls outside a pre-agreed tolerance, an automated model training pipeline is triggered on the latest data. These modelling dashboards can also be useful to understand the real-time performance of the models and increase stakeholder confidence.

[11] McKinney, Wes. "Data structures for statistical computing in python." Proceedings of the 9th Python in Science Conference. Vol. 445. No. 1. 2010
[12] F. Pedregosa et al., 'Scikit-learn: Machine Learning in Python', Journal of Machine Learning Research, vol. 12, no. 85, pp. 2825–2830, 2011.
[13] J. D. Hunter, 'Matplotlib: A 2D Graphics Environment', Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, May 2007, doi: 10.1109/MCSE.2007.55.
[14] "Hierarchical Model Tutorial", ts.gloun.ai. https://ts.gluon.ai/stable/tutorials/forecasting/hierarchical_model_tutorial.html (accessed Jan. 30, 2023)
[15] "FastAPI", FastAPI. https://fastapi.tiangolo.com/ (accessed Jan. 30, 2023)

The specific architecture diagram will depend on the FSPs cloud provider, or whether or not they are using a mixture of cloud and on-prem clusters. But the core components of the architecture are outlined in Figure 11.

Figure 11: The core components of a data pipeline for productionising ML models in an application



7.4 Data collection

The key element of the data collection process is a series of ETL pipelines that takes data from various internal sources (e.g., smart meter data, or asset data) and external sources (e.g., weather, public geospatial shapefiles). It is important that this data is updated regularly.

These pipelines should be rerun at the minimum after each flexibility event, but are ideally run daily. Metadata about the flexibility event should be loaded as well, and all this data is joined together in the next step. We recommend that FSPs follow an Extract, Load, Transform (ELT) pattern for the data, as is common in the modern data stack.

The recommended way to organise this would be to use orchestration tools such as Airflow, and execution tools such as Kubernetes to schedule and run (serverless - so scalable yet low cost) tasks which ingest the data from various sources. This is known as the *extract* step.

This data can then be output to a storage location (e.g., on the Cloud, or on-premises) and then (optionally, yet recommended) loaded into a data warehousing solution. This is known as the *load* step.

This data can then be transformed in the data warehouse, and joined with other data sources, to prepare the datasets used for model training. This can be done with tools such as dbt. This is known as the *transform* step.

All of these steps should be reliable, so that the models can be run on the latest possible data. Our recommendation is that these pipelines are developed and tested in the first few months of CrowdFlex, and “test events” are run with a subset of customers to check reliability of processes.

For real-world productionisation of the ideas in CrowdFlex, the processes should be well tested and refined through the course of CrowdFlex. We recommend a document that

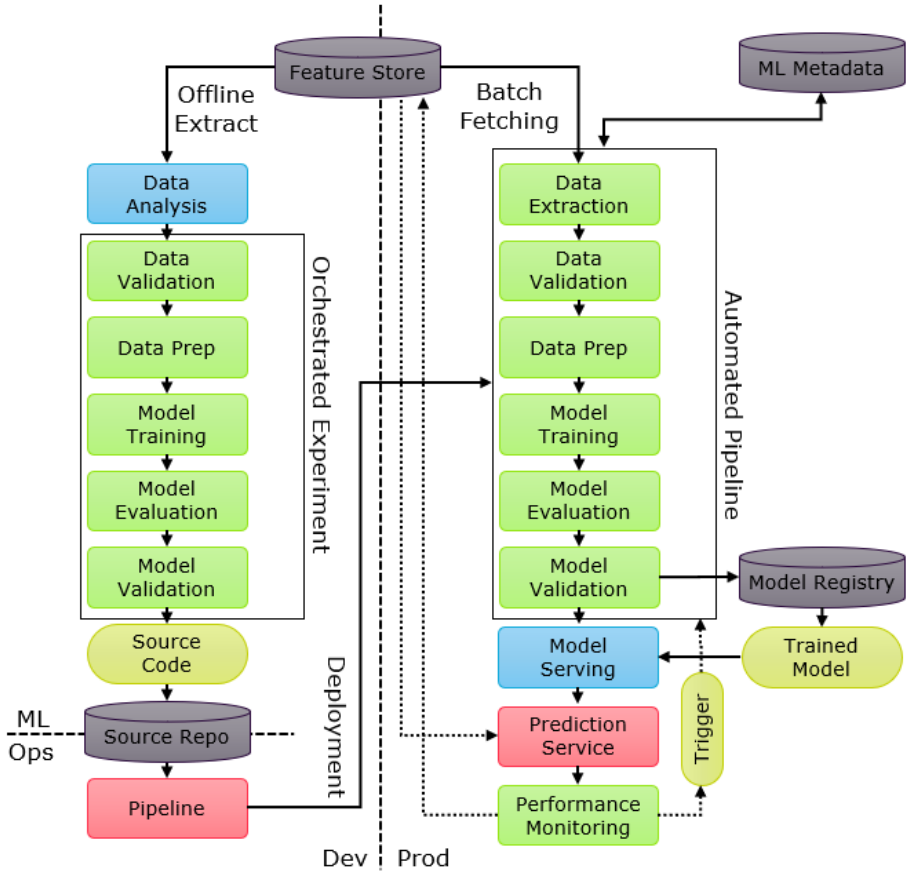
details the early learning and final learning of data flows to be a public deliverable of a model building workstream in CrowdFlex.

7.5 Model (re)training and deployment

Our end-to-end model training, deployment and retraining pipeline should be architected similarly to Figure 12. In the following sections, we describe the three key parts, and our recommendations, in detail:

- Model training
- Model deployment, and security / standards
- Model retraining

Figure 12: An example development and production environment for the deployment of ML models. The models are trained offline in experiments. The pipelines are built, and the production service is entirely automated, with triggers that rerun the training pipeline on a certain schedule or when the model starts to drift according to a specified metric



7.5.1 Model training

The model can be trained in an offline fashion in a “development” environment. This can be on local machines or using cloud VMs (if specific compute is needed to run larger models). Our recommendation is to use tools such as Jupyter Notebooks and Python. We expect that most of the models for CrowdFlex could be trained on local machines, but data scientists may wish to speed up development by using more powerful cloud machines if the time between model iteration cycles becomes prohibitive.

The features are extracted from a “feature store” (typically a SQL database, which has been transformed using a tool like dbt in our previous recommendations to join on all the necessary features together) and various experiments are run to find the best model. To understand how a “better” model is identified in this iteration cycle, see Model evaluation, below.

Once the final model is identified, the pipeline to run this model (as well as retrain it) is then packaged up, and deployed in a production environment which links the final model up to an API endpoint (the *prediction service*).

7.5.2 Model evaluation

Model evaluation is a process where the trained model from the previous step is scored according to an agreed metric of performance. This score provides a single value of the model’s performance over a given dataset. Commonly, the dataset is split into three distinct chunks: the training dataset, the validation dataset and the testing dataset. The high-level purpose of each of these three datasets are as follows:

- **Training dataset**: Used to determine the parameters of the model.
- **Validation dataset**: Used to try different model architectures and evaluate the model to determine which model is “best”.
- **Test dataset**: Used to get an estimate of real-world performance when the model is deployed.

A full discussion of dataset splitting strategies is beyond the scope of this deliverable, but at a high level each dataset should contain the same “distribution” of events (e.g., flexibility events ran at different times of the day, for different lengths) so that the model isn't trained on one distribution of events, but evaluated on another distribution of events. Furthermore, the three datasets should be distinct (i.e., the same event should not exist in both datasets) and “data leakage” should be avoided between the two datasets.

There are different scoring functions to measure the performance of time-series forecasting algorithms, that vary in complexity. Ultimately, the decision of the evaluation metric should be based on the end-use case and be simple to explain to end users what “better” means in the sense of this metric. Common metrics that are used for deterministic models, and that we recommend using for the baseline model in CrowdFlex, are:

- **Mean absolute error**: The average model error (in MWh) for an average half-hourly forecast prediction
- **Mean absolute percentage error**: The average model error (relative, in %) for an average half-hourly forecast prediction.

There are also more advanced metrics that could be designed, such as weighting errors from some settlement periods more than others.

In both of the above cases, lower error is better. During the model training and iteration phase, we should be (offline) evaluating the model on the validation dataset to check the errors that a trained model would have got for predicting those flexibility events, and choose the model with least error.

When deployed in an “online” setting, the model should be monitored live after each event to determine the error in its predictions for each flexibility events. At first, the model is likely to have large errors in specific circumstances, but as we continue to train the model and iterate, the model performance will improve. By the end of CrowdFlex, we should have a

good distribution of flexibility events that can be used to train the final model that can be used.

For the flexibility model, since we are recommending a probabilistic model, we have to quantify how well the quantiles match up to the “true” quantiles of the distribution. We recommend the standard technique of quantile-scoring (based on a pinball loss function) that scores how well the quantiles match with the historical distribution. The flexibility model should aim to be “sharp” (i.e., the distribution of quantiles is relatively bounded) and “precise” (i.e., the quantiles match up to the historical distribution).

7.5.3 Model deployment

Once the model has been trained, this pipeline is then packaged up, and deployed in a production environment (separate from the development environment) which links the final model up to an API endpoint (the *prediction service*). We recommend that this is done in Python, using tools such as FastAPI.

Our recommendation is that the model is deployed on the cloud, to an endpoint which is “always on” and serverless (so it can scale well to a specific number of requests).

We also recommend that the model is deployed to an endpoint that is protected using authentication. Since our model will be deployed to a REST endpoint we recommend:

- HTTPS encryption
- Authentication such as an API key, which expire after a lifetime and is rotated regularly, or OAuth / Bearer tokens.

If using an API key, no two consumer services should have the same API key. In other words, if the model is being used by two different end users, then each end user should have their own API key, which expire after a preset number of days, and are rotated regularly. The exact number of days will depend on ESO guidelines and policy.

When deploying the model, a dashboard of key metrics (such as model performance on previous events) as well as an easy way to interact with the models’ predictions (e.g., using a front-end web app) would increase end-user buy-in for the model, since they can experiment with it under different situations in a simple way.

7.5.4 Model retraining

The model should be retrained on data from new flexibility events regularly. This can be done ad-hoc (e.g., after a few batches) or by triggering an automated retraining job. ESO data and forecasting teams have an ambition in their data strategy to retrain and redeploy other probabilistic models (e.g., weather forecasts) fortnightly, so CrowdFlex could align with this ambition.

Instead of retraining on a schedule, or after a certain number of flexibility events, a more advanced method would be to deploy dashboards that monitor the models performance based on a certain metric (e.g., mean absolute error) and when the model performance drops below a certain pre-agreed threshold, an automated retraining pipeline is triggered. This ensures that the model does not drift and performance becomes unsatisfactory to end users. We propose that CrowdFlex tries this approach in the latter stages of the project, after end users are comfortable with how the model works.

7.6 Milestones and timelines

To build the model, it is recommended that the development team works in “sprints” (e.g., of length 2 weeks) to focus on building out the features and capabilities in a modular and agile manner, optimising for user needs.

The project will be geared to milestones:

- Define and Design - planning of data infrastructure, data sources and data flows.
- Build and Test - building ETL pipelines, databases and model analysis/training platforms.
- Deploy - building public API endpoints to serve the predictions of the model, as well as dashboards for ESO end users to understand model performance.
- Iterate - Work with end users to optimise the deployment and performance of the model ready for wider market trials.

Our ultimate goal is to deliver a Minimum Viable Product (MVP) model (assets, demand forecasts and flexibility) after the first batch of mini trials. We will deploy a dashboard shortly thereafter and work continuously to improve the forecasts and usability of the tools.

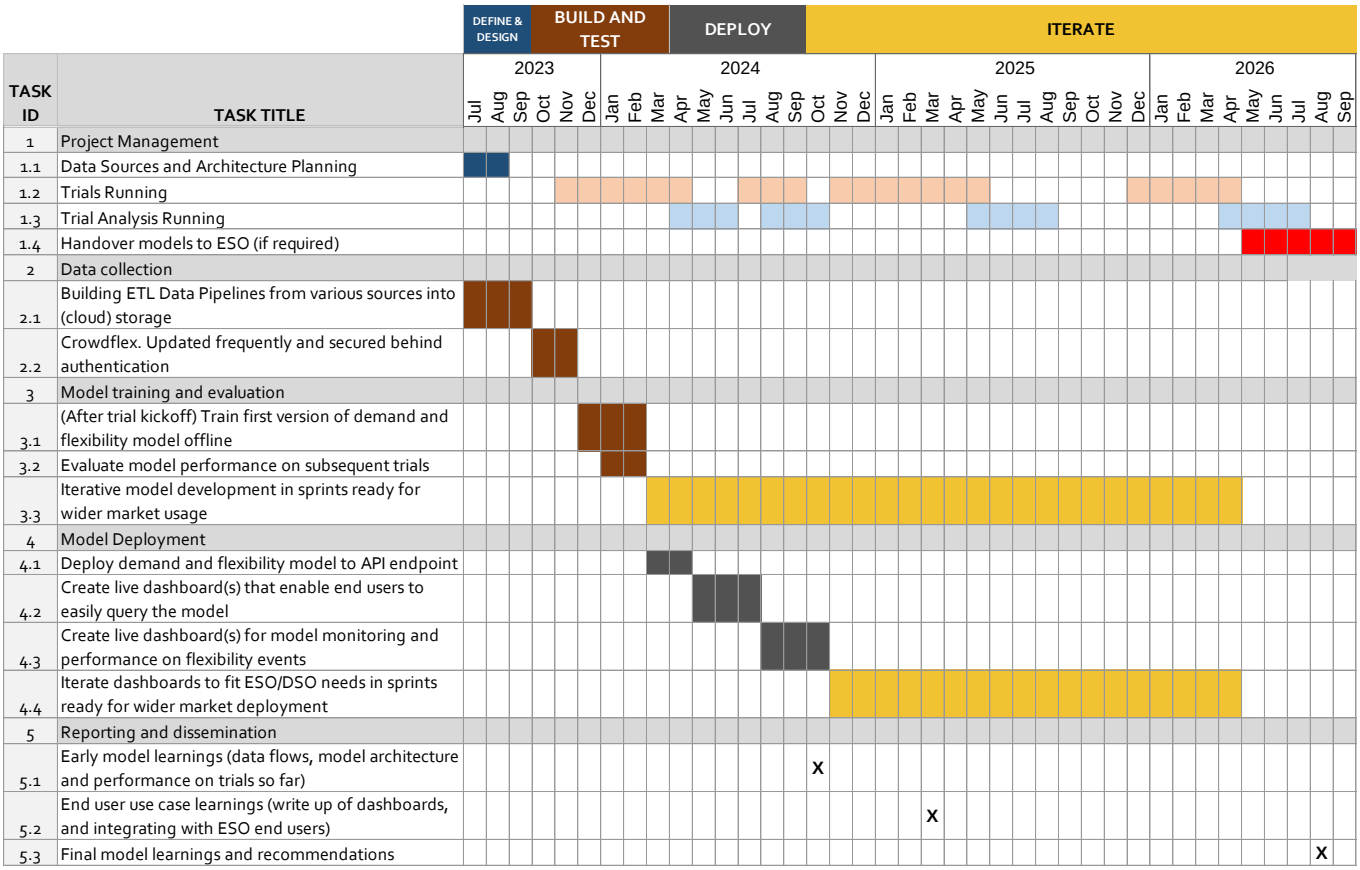
We recommend that there are 3 key modelling deliverables, which may be subsections of other wider consortium outputs:

- Early model learnings (how the model was built, its performance so far) - to be output after the second mini trial. This is the “half-way” point and will include data from two seasons: winter 1 (2023/4) and summer 1 (2024).
- End user use case learnings - a shorter report detailing how end users are interacting with the model, and our intended improvements we are targeting by the end of CrowdFlex.
- Final model learnings and recommendations - the final report that summarises all the modelling learnings from CrowdFlex.

We note that a “draft” of the final model learnings, included the proposed model to be used for wider market trials, should be issued to interested FSPs a few months before the intended start date of the last mini trial batch.

The above is summarised at a high level in the following Gantt chart on the following page, in Figure 13.

Figure 13: A Gantt Chart outlining the delivery of the modelling workstream in CrowdFlex.



8. Appendix A: Methodology for literature review

When performing the literature review, the following search terms were entered into Google, and key journals:

- short term smart meter forecasting
- aggregate (load) forecasting
- probabilistic (load) forecasting
- hierarchical forecasting
- survey of {rule based, machine learning} models (for forecasting)

After which we applied a method of “snowballing” where we checked the reference graph of the initial papers for other relevant papers.

In addition, we found relevant conferences and workshops that specifically focus on the methods described, such as GEFCOM. We reviewed the papers published at these conferences and workshops, as well as related work from the same academics and university research groups which focus on energy system modelling.

9. Appendix B: Full literature review

Note: This section has been summarised in the main paper. There is a repetition of wording in the below to maintain readability.

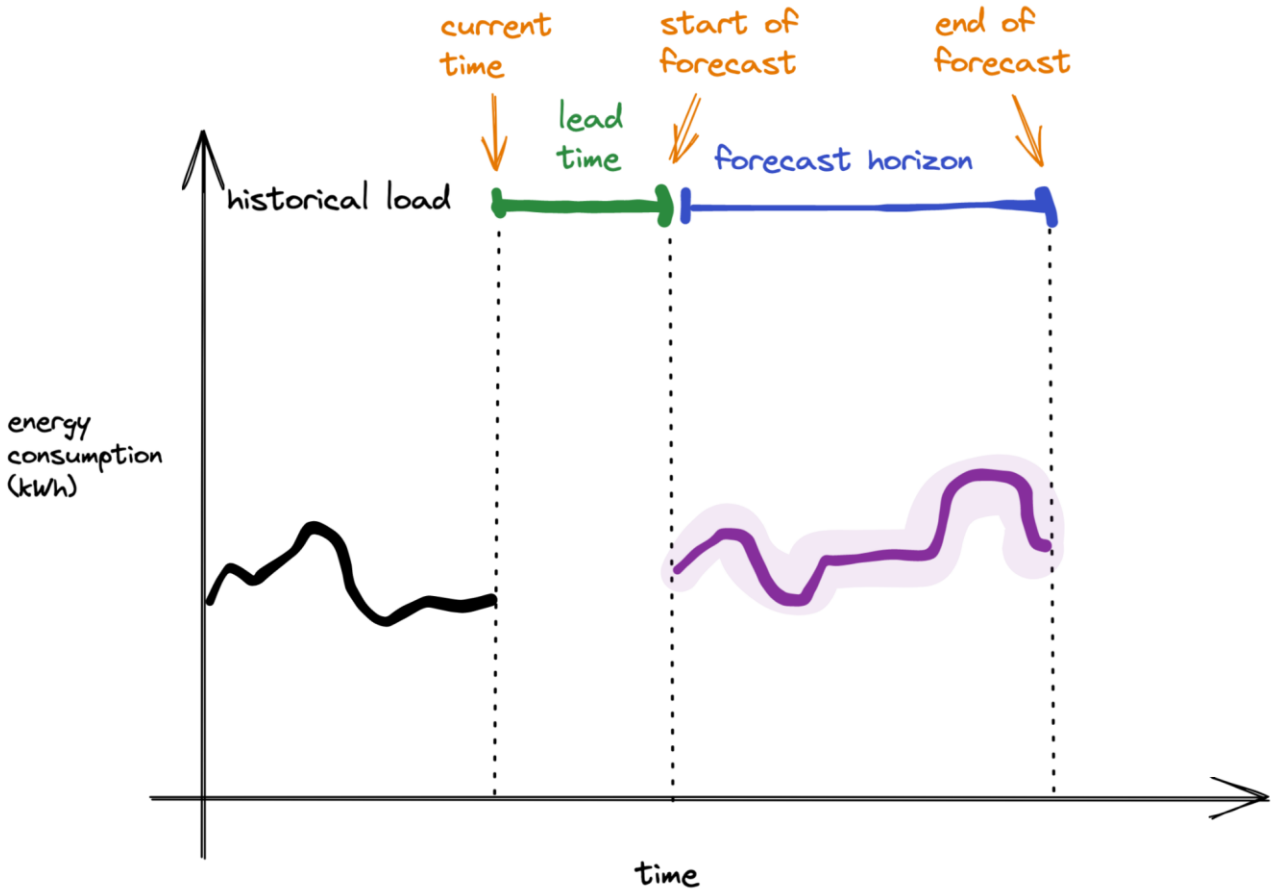
Energy consumption forecasts, or *load forecasts*, are used in many applications across the energy sector, from use cases in network design and planning such as locating and sizing capacitors[16], to network control applications such as a scheduling system for battery energy storage[17], and to anomaly detection such as theft detection in a smart grid[18] or load behaviour after a blackout[19].

Depending on the use case, the forecast horizon (i.e., the time range over which the forecast predicts), the lead time (i.e., whether the forecast is for the next hour, day, year, etc.) and the required data granularity (i.e., half-hourly, daily, etc. - also called the *grain* of the data) might vary. For example, Haben et al (2021)[20] reason that a granularity of 10 minutes to an hour is sufficient for demand control applications but not for voltage control applications.

In addition, the level of aggregation is important. It is possible to forecast load at the lowest aggregation; this may be a forecast of the load profile of individual "units" or appliances, such as electric vehicles (EVs) or of individual households. Forecasting demand at higher levels of aggregation may mean predicting the demand of individual substations or over geographic areas, to which multiple households, buildings and business may be connected. In this case, the load profile is the sum of each of its constituents. At the highest aggregation level, demand is forecasted for the national grid to which *all* households, businesses and other assets are connected. The level of aggregation can impact how volatile the load is and therefore impacts the predictability of future load.

[16] A. Dupka, B. Venkatesh and L. Chang, "Fuzzy Stochastic programming method: Capacity planning in distribution systems with wind generators," IEEE Transactions on Power Systems, vol 26(4), pp 1971-1979, 2011.
[17] C.J. Bennett, R.A. Stewart and J.W. Lu, "Development of a three-phase battery energy storage scheduling and operation system for low voltage distribution networks," Applied Energy, vol. 146, pp. 122-134, 2015.
[18] V. Loia, G. Fenza and G. Mariacristina, Drift-aware methodology for anomaly detection in smart grid," IEEE Access, vol. 7, pp. 9645 - 9657, 2019.
[19] L.T.M. Mota, A.A. Mota and A. Morelato, "Load behaviour prediction under blackout conditions using a fuzzy expert system," IET Generation, Transmission & Distribution, vol. 1, no. 8, pp. 379 - 387, May 2007.
[20] S. Haben, S. Arora, et al., "Review of low voltage load forecasting: Methods, applications, and recommendations," Applied Energy. vol. 304:117798, Dec 2021 Dec.

Figure 14: Visualisation of concepts of forecast horizon and lead time



In CrowdFlex the use case of interest is forecasting domestic electricity consumption to enable households to participate in flexibility services. Thus, most of the research reviewed here will focus on half-hourly meter readings, since this is most representative of the domestic smart meter data that is available. We note that different Flexibility Service providers may instead have sub-half-hourly data, for example, near real-time data explicitly from an asset such as an EV chargepoint or heat pump (HP). In such cases, sub-half-hourly data can be resampled to half-hourly if needed, and so the techniques in this literature review still apply.

Additionally, since households using manual and automated dispatch can participate in trial services for the day-ahead (as was the case in the Domestic Reserve Trials[21]), this literature review mainly reviews forecasting methods that perform well for the day ahead forecast horizon. The research literature on intraday methods and forecasting in an operational setting is less mature, since that is really a question of deployment. Clearly the modelling methods developed in CrowdFlex should be able to generate forecasts intraday as well as day-ahead, to maximise the value of the model to the energy system. In terms of modelling techniques, the same techniques can be applied intraday as well as day-ahead, but might have different constraints from a data availability and deployment perspective

[21] "National Grid ESO and Octopus Energy launch trial to unleash demand flexibility this winter", National Grid ESO. https://www.nationalgrideso.com/news/national-grid-eso-and-octopus-energy-launch-trial-unleash-demand-flexibility-winter (accessed Jan. 30, 2023)

(e.g., it is likely to more costly to deploy a model that operates in near-real-time compared to updated once per day).

Since each household may need to be rewarded for their participation proportionally, forecasts will need to be generated at the household level.

It should be noted that some households, for example those on Intelligent Octopus[22] and Ohme EV customers, may be able to participate in events closer to real-time. In this case, we have asset level data (including programmed charging schedules, voltage, current) at an approximately-minutely level for this specific subset of households. Therefore, different forecasting methodologies may be used to predict the flexibility potential of this subgroup due to the different grain of the data.

Unlocking demand side response for domestic customers will require a few components from a technical perspective:

- *baseline* or *counterfactual*, an estimate of the energy consumption had a flexibility request not been made,
- a method to model how the baseline might change as a result of the flexibility request, and
- a measure of the energy consumption of those who participated in a particular flexibility service.

The above assumes that the forecast takes inputs about consumption, but also other factors that influence the *flexibility capital* of the household we are forecasting. A more detailed discussion of this, as well as example inputs, is given in the literature review in CrowdFlex deliverable D2.1. We should ensure that the models perform fairly across a range of customer attributes.

Identifying the methods to establish the first two of these components is the focus of this report. The last component is easily attainable for customers with a smart meter with the required grain.

9.1 Load forecasting algorithms

The *baseline* can simply be thought of as the estimate of consumption if there was no flexibility event. In other words, it is the answer to “what would have happened if there were no flexibility event?”. The difference between this estimate and the observed metered data over a specified period can then be considered as the flexibility provided. There are many ways to calculate the baseline, the appropriate method being dependent on a few factors related to the use case, such as:

- the lead time and forecast horizon (defined above);
- the (temporal) granularity of data available;
- the (spatial) granularity of the data we are forecasting (e.g., household, or a more aggregated level);
- Deterministic vs Stochastic;
- Point estimate vs Probabilistic.

In deterministic models, the same input produces the same output. In stochastic models, the same input produces slightly different outputs (which form a distribution). Models that produce a point estimate output a single valued prediction for a given input (e.g., the

[22] “Intelligent Octopus”, Octopus Energy. https://octopus.energy/intelligent-octopus/ (accessed Jan. 30, 2023)

expected flexibility at a given half-hour), whereas models that output a probabilistic distribution output either the parameters of a distribution, or the quantiles that characterise the range of values.

The modelling decisions depend on the use case. The literature review below surveys a range of different types of models most common in the literature.

9.2 Rule based algorithms

Rule based algorithms are the simplest form of prediction. Common time-series forecasting rule-based algorithms include:

- Persistence algorithms - e.g., using the day before, or "previous similar day" to predict the next day;
- (Weighted) average algorithms - e.g., using a four-week (weighted) average to predict a given day;
- Rolling average models - e.g., using a combination of daily, weekly and monthly rolling averages to predict a given day.

These simple, explainable algorithms are often called "naive" (forecasting) models.

Regardless of whether the baseline is being calculated for a single household or for an entire region, rule-based algorithms such as a historical average or a persistence forecast perform reasonably well, and even competitively. For example, Groß et al (2021)[23], used data from the previous week as a naive model. At the aggregate level for residential load, this method outperformed a more complicated machine learning model, though the authors noted that the differences were small and an outright winner was not recommended.

In the Domestic Reserve Scarcity (DRS) trials, a four-week average was used to estimate the consumption of each participating household. This baseline was used to communicate demand reduction targets and subsequently to reward participants.

Another rule-based algorithm that is already recommended (and set to be implemented) for use in the balancing market by Ofgem is the P376 baseline,[24] which uses either the most recent 4 non-working days or 10 working days without any DSR events to predict the load for the day ahead. The balancing market does not yet contain any residential domestic customers (with the exception of some innovation trials, such as Powerloop[25]), but is intended to apply to those assets when they enter this market. The P376 baseline is also the chosen baselining method for the Winter 2022 Demand Flexibility Service[26]. An addition of an in-day adjustment which uplifts or downturns baseline values using metered data from the settlement periods immediately preceding gate closure on a day where there is a flexibility event helps account for the impacts of weather and temperature, and mitigates against any baseline "drift" (where the model systematically over or under predicts values).

[23] A. Groß, A. Lenders, et al., "Comparison of short-term electrical load forecasting methods for different building types," Energy Inform, vol. 4, no. 13, pp. 1 - 16, 2021.
[24] "Utilising a Baselining Methodology to set Physical Notifications for Settlement of Applicable Balancing Services", Elexon. https://www.elexon.co.uk/mod-proposal/p376 (accessed Jan. 30, 2023)
[25] "Powerloop: Vehicle-to-Grid Technology", Octopus Energy. https://octopusev.com/powerloop (accessed Jan. 30, 2023)
[26] "Demand Flexibility Service", National Grid ESO. https://www.nationalgrideso.com/industry-information/balancing-services/demand-flexibility (accessed Jan. 30, 2023)

Algorithms such as these have multiple benefits: they are simple to understand, they are relatively easy to deploy and maintain and they also scale well when there are a large number of participants. More complex algorithms may be difficult to implement, or require specific expertise in order to deploy the model (for example, machine learning methods). If a rule-based algorithm performs comparably to a more complex algorithm (e.g., in terms of a desired metric, such as mean absolute error), then modellers usually default to a rule-based methodology. For these reasons, rule-based algorithms can be a good choice for a standardised baselining methodology for a new service.

9.3 Features and model inputs

Only historical load is used to create forecasts using rule based algorithms. However, other factors such as weather, household characteristics and calendar variables (such as “day of week” or “month of year”) may improve the forecast accuracy. Humeau et al (2013)[27] found that a simple linear regression (LR) model (eq. 1) worked best out of all the models they implemented at the household level. They utilised input features of consumption and calendar variables for their day-ahead forecast whereas their hour-ahead forecast also included derivatives of load, to capture recent changes in demand. The same study also found benefit in adding temperature variables to forecast aggregated load but noted that this feature added no predictive power at the household level.

Equation 1

$$y = X\beta + \epsilon$$

In the above equation, y represents the load, X represents the input features which may be historical load or temperature, as discussed, β is a vector of coefficients that are learned during the training and ϵ is the random error term, assumed to be independent and identically distributed with mean zero and some known variance. Values of β close to zero suggests the corresponding input feature has little predictive power.

Calendar variables, such as day of the week, hour of the day, can also be used as inputs to the model. Groß et al (2021) used Fourier transforms to capture the cyclic dependence of calendar variables which they argue allows models to be trained with fewer inputs.

Some studies have highlighted the value of appliance data along with smart meter data for forecasting at the household level.[28,29] In one of these studies, simply adding the appliance data improved the mean absolute percentage error (MAPE) from 34% to 28%. Considering the best model had a MAPE of 22% and was substantially more complex, it is possible that the addition of appliance data is more valuable than a more complex machine learning algorithm.

Domestic energy consumption is highly variable, even on similar types of days. Smart meter data can also report incorrect values, and so requires data cleaning and processing.

---

[27] S. Humeau, T.K. Wijaya, et al., “Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households,” 2013 Sustainable internet and ICT for sustainability, pp. 1-6, October 2013.

[28] W. Kong, Z. Y. Dong, et al.,, "Short-Term Residential Load Forecasting Based on Resident Behaviour Learning," IEEE Transactions on Power Systems, vol. 33, no. 1, pp. 1087-1088, Jan. 2018.

[29] M. A. Shah, I. A. Sajjad, M. F. N. Khan, M. M. Iqbal, R. Liaqat and M. Z. Shah, "Short-term Meter Level Load Forecasting of Residential Customers Based on Smart Meter's Data," 2020 International Conference on Engineering and Emerging Technologies, pp. 1-6, 2020.

Doing so can improve forecast accuracy even without accounting for model complexity30. In fact, three out of the top five performing algorithms in GEFCOM 2012 explicitly used data cleansing methods. Groß et al (2021) outlined a selection of data processing methods such as replacing missing values with linear interpolation, outlier detection and feature scaling. The feature scaling used in Groß et al (2021) was min-max scaling, which is popular in the literature, though others such as standard normalisation could also be used.

9.4 Statistical models

In the last decade, time series models have proved successful in predicting load forecasts as they are able to exploit the temporal dependencies. For example, Marinescu et al., (2013)[31] tested several algorithms that forecasted large-scale load to a high accuracy and found an autoregressive integrated moving average (ARIMA, eq. 2) forecast performed the best overall for a day-ahead forecast.

An ARIMA model is made up of three parts: i) an autoregressive part where the output depends linearly on its own previous values, ii) a moving average part which indicates that the forecast error is a linear combination of past errors, and iii) an integrated part which indicates that the data values may have been replaced with the difference between the values and previous values, to remove trends or seasonality.

Equation 2

$$X_t - \alpha_1 X_{t-1} - \cdots - \alpha_p X_{t-p} = \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$$

Mubashar et al (2022)[32] implemented a complex machine learning algorithm, which was benchmarked against an ARIMA model as well as an exponential smoothing model, another popular time series method. While the model in Equation 2 cannot directly take temperature or other exogenous variables, modifications can be made to add this functionality.

Exponential smoothing algorithms also offer plenty of flexibility. It is possible to add non-load variables as well as parameters that are cyclic in nature (for example day of the week or time of the day) to improve forecast accuracy. The simplest exponential smoothing algorithm, which does not take into account seasonality, returns a weighted average of the previous forecast and the previous actual value (eq. 3).

Equation 3

$$\hat{x}_{t+1} = \alpha x_t + (1 - \alpha)\hat{x}_t$$

However, the time series algorithms are not without drawbacks. It can be complex to include temperature variables or other features in such models. Further, when there are multiple time series, e.g. one for each household, estimating the parameters such as α and θ in the above equations can be very time consuming so they are best used when dealing with aggregated load.

[30] T. Hong, P. Pinson, and S. Fan, “Global energy forecasting competition 2012,” International Journal of Forecasting, vol. 30, no. 2, pp. 357-63, Apr 2014.

[31] A. Marinescu, C. Harris, et al., “Residential electrical demand forecasting in very small scale: An evaluation of forecasting methods”, 2013 2nd International Workshop on Software Engineering Challenges for the Smart Grid, pp. 25-32, IEEE., May 2013.

[32] R. Mubashar, M.J. Awan, et al., “Efficient residential load forecasting using deep learning approach,” International Journal of Computer Applications in Technology, vol. 68, no. 3, pp. 2015-214, 2022.

Also, as was seen in the work by Mubashar et a (2022), the training was done using data from 12 households in total, but the forecast accuracy varied drastically differently depending on the household; in some cases, the mean absolute error (MAE) was as high as over 100 kW and in other cases as low as 2 kW. This variability was particularly driven by the household and not choice of forecasting method.

Support Vector Regression (SVR) or support vector machines have been a historically popular algorithm. Unlike other regression models, which try to minimise the error between actual and forecasted values, SVR tries to fit the best line within a threshold. Shi et al., (2017) improved their ARIMA by roughly 5-7% with a SVR algorithm. Humeau et al., (2013) used SVR as one of their more complex algorithms and found it to be quite competitive with the best algorithms, with a 2% improvement for day-ahead forecasting.

The K-nearest neighbour (KNN) model has also become popular in individual load forecasting. KNN, when used for forecasting, predicts an output based on k of the closest data points, where k is a parameter that needs to be identified (e.g., by averaging the k nearest neighbours' historical values). This method can be approximated to still yield good results but is scalable over larger quantities of data.[33] In Groß et al., (2021), the KNN algorithm performed the best at the individual household level, with an improvement of 10% over the naive model, but was somewhat similar to the LR and SVR model. Kong et al., (2017) implemented several KNN models, some with household load only, and others with appliance data as well as household load. While the KNN was not their best algorithm, the MAPE of the best KNN model was 26%, compared to the 22% of the best algorithm, which was substantially more complex.

9.5 Machine Learning Models

Machine learning models are models that learn their parameters explicitly based on the (training) data. In supervised learning, which is the focus of this literature review and the task for CrowdFlex, we collect *labelled* data of a set of *features* about an event, and the *target* variable we want to predict. The process for collecting data and training these models will be deferred to the Model delivery plan (section 7, below) but as an overview:

- Data is collected through trials and experiments.
- Models are specified, and their parameters are learned from this historical data.
- The model(s) are iterated and updated based on collecting new data.
    - Different types of model architectures, and model inputs can be tested to improve accuracy.
    - The model(s) are retrained on a basis determined by the use case (e.g., monthly).
- Once a machine learning model has been trained on historical data, it can be "packaged up" and used to predict new events (known as *inference*).

While time series models are statistically robust, they do not scale well when multiple households require an individual baseline. To address this, more recent research has looked at machine learning algorithms. For example, Shi et al., (2017)[34] implemented three different types of recurrent neural networks (RNN) with an ARIMA model as the benchmark

[33] E. **Bernhardsson, "Annoy: Approximate Nearest Neighbors in C++/Python", GitHub Manual, https://github.com/spotify/annoy, 2018.**

[34] H. Shi, M. Xu, and R. Li, R, "Deep learning for household load forecasting—A novel pooling deep RNN" IEEE Transactions on Smart Grid, vol. 9, no. 5, pp. 5271-5280, 2017.

to beat. The best of their RNNs had a mean absolute error (MAE) of 0.25 kWh, compared to 0.30 kWh for the ARIMA model (an improvement of 16%) and 0.29 kWh for the SVR model (an improvement of 12%).

However, the performance of machine learning models can be mixed, especially depending on the architecture. A very simple and shallow machine learning model may not offer much increase in accuracy. For example, the multiple layer perceptron (MLP, Equation 4), with one hidden layer implemented in Humeau et al., (2013) was no better than the linear regression model and the SVR model.

Equation 4

$$y = W_2 \times \sigma(W_1 X + B)$$

where X are the input features, y is the load, σs is a non-linear transformation and W_1, W_2 and B, often referred to as weights and biases, are coefficients to be determined through training. These types of models are quite flexible as the inputs can be varied and nonlinear dependencies can be modelled. Further depth and complexity can also be built in by adding more sequences of nonlinear transformations.

In recent years, many load forecasting algorithms have been suggested using long short term memory (LSTM) models. LSTMs can process entire sequences of data and thus lend themselves to time series quite well. These types of models can thus learn how much of the past data to retain and can thus learn underlying patterns and trends. In load forecasting, Kong et al., (2017) implemented several algorithms as already discussed, including some LSTM models. Using an empirical mean model (a rule-based algorithm) as a benchmark - which scored a MAPE metric of 50% - the LSTM model with the same inputs had a MAPE score of 27%. One benefit of the LSTM approach is the ability to add additional features, such as appliance data. The LSTM with extra appliance data had a MAPE of 22%, improving on both previous models. This suggests that improvement comes both from the addition of the extra data and from the choice of architecture. However, this improvement comes at a significantly higher computational cost, particularly in the training of the LSTM model. Once trained, even though the inference time of the machine learning model can be higher than a rule-based algorithm, it is not prohibitive for energy system forecasting applications.

Semmelmann et al., (2022)[35] applied the LSTM model to the aggregated load of 130 households in Germany to forecast the load pattern and XGBoost component to predict the peak. XGBoost is an algorithm shown to perform well on time series applications, particularly in finance.[36] This model had an overall MAPE of 10-25% depending on the components and inputs to the model.

Finally, Hou et al., (2021)[37] also implemented an LSTM model to predict both the load of individual households and the aggregated load for a cluster of households. The inputs to the model included maximum and mean daily load, as well as indices related to the volatility of the load. When forecasting the load of 200 households individually, 70 households had a

[35] L. Semmelmann, S. Henni, and C. Weinhardt C, “Load forecasting for energy communities: a novel LSTM-XGBoost hybrid model based on smart meter data,” Energy Informatics, vol. 5, no. 1, pp. 1-21, Sep 2022.
[36] Vadim Borisov, Tobias Leemann et al., “Deep Neural Networks and Tabular Data: A Survey”, ArXiv, Jun 2022
[37] T. Hou T, R. Fang R, et al., “A novel short-term residential electric load forecasting method based on adaptive load aggregation and deep learning algorithms,” Energies, vol. 14, no. 22, pp. 7820, Nov 2021.

MAPE between 20-40% while 40 had a MAPE of over 100%. The average MAPE was around 74% even with such an advanced method. This highlights how difficult forecasting individual households is, with some households being more difficult to predict than others. Clustering households and predicting the aggregated load reduced the average MAPE to between 8-20%.

9.5.1 Computational cost of machine learning models

Simple Machine Learning models can be run locally on most laptops. However, when "tuning" these models (a process by which the model is tweaked over several iterations - changing the model architecture or hyperparameters) the model training process needs to be run several times. It is not uncommon to train hundreds of different variations of models to improve performance on a key metric while in this experimental stage.

More complex machine learning models, such as deep learning methods, may require increased memory (RAM) and storage space. It may need to be run solely in the cloud as the dataset(s) and/or model cannot fit on a standard laptop.

The computational cost is proportional to:

- The training time per model
- The number of (variations of) models that are trained
- The storage / memory requirements of the machine used to train the model (locally vs cloud training, and dependent on the size of the data to train the model as well as the size of the model itself)

There is a trade-off between the computational cost of training complex models and performance improvements. For example, if a 5% reduction in your error metric (such as mean absolute error) means we can increase our accuracy by ~100MWs, then the model training cost is minimal in comparison to the downstream value it creates.

Understanding this trade-off is the job of data scientists in conversation with their end-users, to understand what level of performance is suitable for the end use case, and how much model iteration should be done to balance the training cost with sufficient level of accuracy.

Typically, there is lots of experimentation at the start and once a suitable model architecture is identified, it is retrained on the latest data by "freezing" the desired model architecture and hyperparameters. If the model drifts from its desired performance, further iteration of the model may be required (which increases computational cost, but will result in a more accurate model).

9.5.2 Limitations of Machine Learning Models

While machine learning methods have demonstrated performance improvements in some scenarios, and enable us to easily add additional features that would be impossible in other approaches such as ARIMA, there are also some limitations.

One common limitation of machine learning models is that they are "black-box models". Since the formulation of the predicted variables is not explicitly given by an easy-to-understand formula, it can be hard to interpret why a model makes a given prediction. This is important for the domestic flexibility use case since the electricity consumption forecast may be applied at an individual asset level, and so explainability becomes crucial in communicating to households how that baseline was calculated.

More complicated models have shown to be more performant (e.g., less error / more accurate) and so there is a trade-off between model performance and explainability that needs to be balanced. Some techniques have attempted to identify how machine learning models make predictions, but these techniques work by analysing predictions on different outputs, rather than the model itself being explainable by construction (so-called “white box” models, which include linear regression).[38]

Development, iteration and deployment of machine learning models also require a specialist skill set. Overly complex models could limit industry adoption of techniques, so simpler methods should be preferred. In addition, more complex models can mean that long-term support is more costly. It is cheaper and simpler to maintain a simple model compared to a complex one. For two models that have similar performance, the principle of Occam's Razor should be followed: choose the simplest.

9.5.3 Probabilistic load forecasts

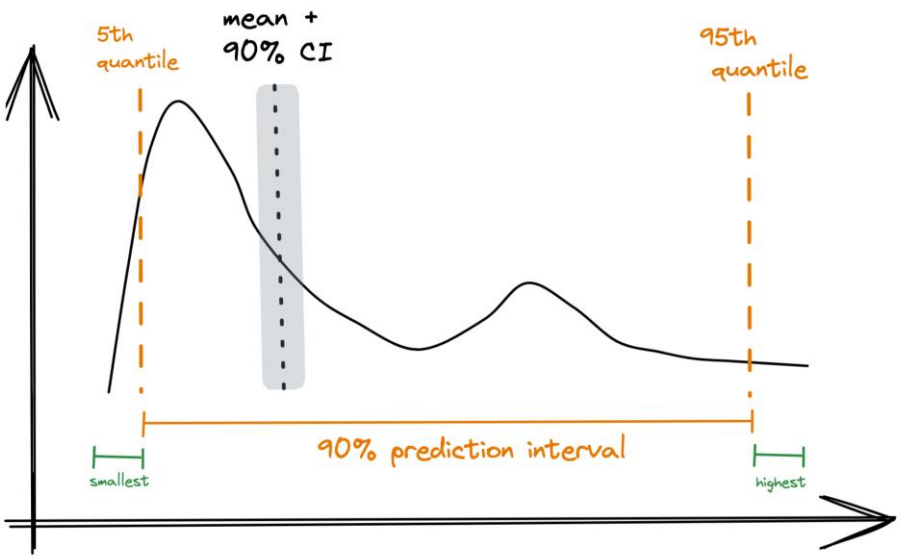
Load forecasts can broadly be thought of as deterministic or probabilistic. Thus far, this literature review has focussed on point load forecasts. A point load forecast is one that forecasts a single value for each settlement period, often estimating the average as discussed above. Probabilistic load forecasts (PLFs) instead provide multiple estimates for each settlement period and in doing so capture some of the uncertainty associated with the estimate. The literature on probabilistic load forecasts (PLF) is less mature even though their value to enabling domestic flexibility services has been made already in CrowdFlex deliverable D1.2.

PLFs can be in the form of *confidence intervals* or *prediction intervals*. Confidence intervals describe the range of values the *average* of a variable can take while prediction intervals describe the range of values the variable itself can take (Figure 15). Consider the example where we are predicting house prices: the 90% confidence interval refers to the fact that we are 90% confident that the average house price (in a certain region) is £X. We would be able to make statements such as “in region R, we are 90% confident that the average house price is in the range [£395,999, 405,000]”. By contrast, a 90% prediction interval describes the distribution of house prices excluding the cheapest 5% and the most expensive 5%. For a prediction interval, we would be able to make statements such as “in region R, 90% of house prices fall between [£295,000, £680,000]”. Thus the prediction interval is wider than the confidence interval.

The type of model we choose will depend on if we care about predicting the average with a certain degree of confidence (and so choose confidence intervals) or if we want to understand the distribution of potential outcomes to quantify risk (and so choose prediction intervals).

[38] S. M. Lundberg and S.-I. Lee, ‘A Unified Approach to Interpreting Model Predictions’, in Advances in Neural Information Processing Systems, 2017, vol. 30. Accessed: Jan. 30, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

Figure 15: Depicts confidence interval (shaded grey) and prediction interval. For the energy-specific use case, the x-axis would refer to kWh (at the household level) or MWh (at a higher level of aggregation).



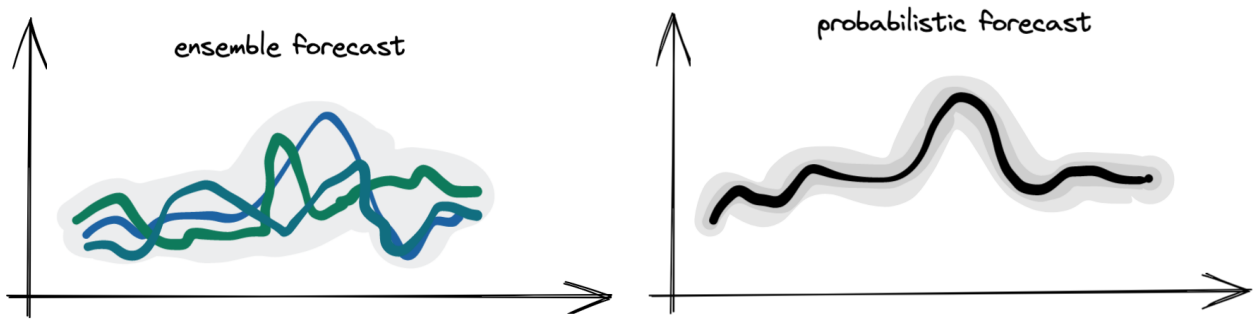
GEFCom2014[39] explored this concept for a US utility on a rolling basis. Participants were expected to create a forecast for each percentile for a month ahead. The best algorithms included quantile regression and generalised additive models while others also included extensions of exponential smoothing and ARIMA. It must be noted that the competition did not focus on creating PLF at the household level.

With a probabilistic type of forecast, traditional forecast evaluation metrics like MAPE and MAE may not be sufficient. This is because we are now dealing with a distribution of values, not a "point estimate" like the mean. Haben et al., (2021) suggest that appropriate evaluation metric must reward the forecast if i) it is sharp, i.e., gives a higher score for a narrower distribution, and ii) well calibrated meaning that, for example, if the prediction interval is 90% then 90% of the predictions must lie in that interval. In GEFCom2014, the pinball loss was used but the continuous ranked probability score may also be an appropriate choice.

Multiple deterministic forecasts could be combined to produce a single forecast. This is known as ensemble forecasting. Ensemble forecasts can either be deterministic (averaging multiple deterministic forecasts to create another deterministic one) or probabilistic (using multiple deterministic forecasts to construct quantiles).

[39] T. Hong T, P. Pinson, et al., "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," International Journal of forecasting, vol. 32, no. 3, pp. 896-913, Jul 2016.

Figure 16: Types of probabilistic forecasts. Note that the left-hand side could be used to construct quantiles (rather than showing each model separately) if needed.



9.6 Suitable load forecasting models for domestic flexibility

The literature does not provide a consensus on which algorithms to use but it gives a lot of candidates. It has proved difficult to compare them in the academic literature as the benchmark is not always a standard algorithm and the evaluation metric is not common. For this reason, competitions such as GEFCOM have been designed with a common error metric and a benchmark to beat. Nonetheless, questions still remain such as what is a good trade-off between accuracy of forecast and ease of implementation, and if the forecasts work well for different kinds of households. Both the literature and GEFCom have not addressed if there are specific time periods where some forecasts perform better than others, though the work of Marinscu et al., (2013) seems to indicate that this may be true. Further, there are very few smart meter datasets that are publicly available that tell you characteristics of the household such as LCT ownership and thus there is no discussion on which algorithms might perform better if these characteristics are known. The only way is to generate data on your specific problem and trial a range of algorithms to find out which method works best.

Lastly, it is assumed in many of these studies that the data to create the forecast is available at the time of forecasting but this is not necessarily the case in practice, for example, if the goal is to predict the day ahead and we use the previous day as a baseline, is the smart meter from the previous day available when they forecast is being created? We give a specific recommendation in relation to the domestic flexibility problem in Section 6.6 (Data availability for modelling), but broadly the literature proposes that infilling missing values using interpolation or “forward-filling” the most recent data is standard practice in forecasting. The “freshness” of data is important for forecasting, as we want to have reliable data pipelines that give us the latest data to use for our model predictions.

In conclusion, forecasting load at the household level is a complex task, not just because information on the appliances and their performance is not easily available but because human behaviour is quite spontaneous and unpredictable. Moreover, as has been noted, forecast errors differ dramatically depending on the household even when the same algorithm is used.

One way to get around this is to forecast the aggregated load of multiple households as this is a much simpler task; certainly, research suggests the errors are much lower at an

aggregate level (with some studies quoting around 5%[40]). Another way around is to cluster customers who are similar and forecast on each cluster's aggregated load. This has been done in many studies. For example, Humeau et al (2013) compared the forecast errors of each algorithm on individual households, on clusters of similar households and aggregate of all households. For the SVR algorithm forecast errors were lower if households were clustered into four households but for the others, forecast errors were lower when all households were aggregated. Clustering improved forecast errors on all algorithms compared to forecasting individual households. In contrast, Shi et al (2017) used clustering to build models on different groups of households. Each household was randomly assigned to one of 10 clusters and a separate model was trained for each cluster. Using the best model for the group, forecasts were then created for individual households. This approach could be further strengthened by clustering households with similar appliances or characteristics. In general, however, clustering approaches may not be appropriate as household forecasts are necessary to calculate and reward individual participation in flexibility service. Due to the competitive and scalable nature of rule-based algorithms, these might suffice at the household level. The historical household load may be aggregated at the supplier or substation level and more complex algorithms such as those discussed in this review may be used to create more accurate forecasts at the supplier level.

9.7 Demand side response and intelligent demand

While it is crucial, calculating an appropriate baseline is only one part of unlocking the flexibility value in domestic demand side response (DSR). It is equally important to be able to model the response consumers will provide for a given flexibility request. Priolkar et al., (2020)[41] demonstrated the financial value of demand response to both the DSO and the consumer from participating in varying DSR programs. In this study, a baseline of one substation is adjusted to provide the desired change in demand. The financial value of the change is then evaluated based on various DSR programs, i.e. the financial benefit to a consumer on a flat tariff, versus Time-of-Use (ToU) tariff, etc. and the value to the DSO. The latter remained unchanged because it was assumed that consumers made the curtailments as instructed. While this is somewhat artificial, the study underpins the need for accurate baselines so that the correct instruction can be given to customers and thereby maximise the value to both the consumer and to the DSO.

In contrast, Garulli et al (2015)[42] propose an approach to load forecasting in the presence of active demand, i.e., to forecast directly the response provided by domestic consumers in a demand response scenario. In this approach, the assumption is that some consumers are managed by an aggregator who has agreed to provide a certain response, in the form of a load profile, based on the needs of the market and this load profile has been validated by the DSO. The aggregator then sends an appropriate price/volume signal to its consumers to elicit the desired response. Thus, the problem is to predict what the customers will do

[40] Y. Wang, Q. Chen, et al., "An ensemble forecasting method for the aggregated load with subprofiles," IEEE Transactions on Smart Grid, vol. 9, no. 4, pp. 3906-3908, Feb 2018.

[41] J. Priolkar, E.S. Sreeraj and A. Thakur, "Analysis of Consumer Baseline for Demand Response Implementation: A Case Study," 7th International Conference on Signal Processing and Integrated Networks, pp. 89-94, 2020.

[42] A. Garulli, S. Paoletti and A. Vicino, "Models and Techniques for Electric Load Forecasting in the Presence of Demand Response," IEEE Transactions on Control Systems Technology, vol. 23, no. 3, pp. 1087-1097, May 2015.

when given the price signals and assuming the aggregator has additional information such as historical load, temperature and calendar information. This problem formulation is different from the approach proposed in CrowdFlex: specifying specific windows and prices for successfully turning down demand, rather than giving customers a new price signal to follow.

Larsen et al (2017)[43] evaluated DSR by forecasting the consumption using weather variables, real-time and day-ahead electricity prices. Various data processing techniques were applied such as scaling and fourier time series to capture the dominant interactions. The consumption forecasts were created in aggregate for groups of households where some households had no pricing information (control), while others had pricing information but were required to manually provide DSR, and yet others who had pricing and automation software including an aggregator who manage load, similar to the vision in Garulli et al (2015)9. The trials used 1900 households, focussing on heating, where the DSR was expected to be bid into a real-time balancing market. The results of the modelling showed that households with aggregator control had the fastest response to pricing as well as the largest. While there was some DSR from the control group and the manual automation, it was not enough to distinguish from model uncertainty. The model was trained on very high grain (5 minutes) data but most smart meters in households do not provide this level of granularity. Thus for a full scale roll-out, the authors suggested that other data sources such as system level frequency, rather than household load, could be used to supplement or, alternatively, select households which are reliable and representative of the population to be used to replicate the methodology.

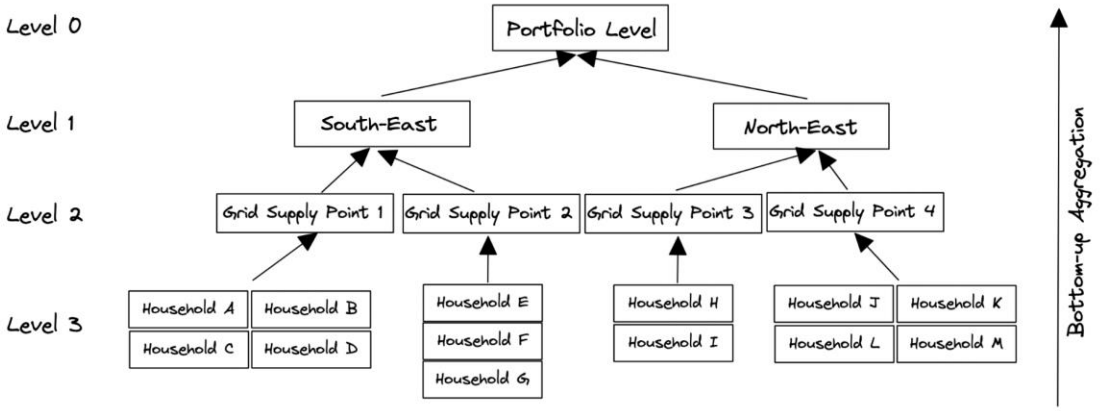
9.8 Hierarchical Forecasting

Hierarchical forecasting is the term used for when we need to forecast multiple series that depend on each other according to a hierarchy. Example of such a series is smart meter data where the lowest level is household level, which can be aggregated up to regional level and finally to portfolio level.

In the context of domestic flexibility, this hierarchical forecasting is helpful as the same set of forecasts can be used in different systems (e.g., at GSP level, vs GSP Group level, vs national level). By forecasting at higher levels in the hierarchy, we can protect household confidentiality and ensure the protection of personal data (since smart meter data is considered personal data under the Data Protection Act and GDPR).[44] The household level forecast is crucial for Flexibility Service Providers to enact settlement, but may not be required by ESO, for example.

[43] E.M. Larsen, P. Pinson, et al., "Demand response evaluation and forecasting—Methods and results from the EcoGrid EU experiment." Sustainable Energy, Grids and Networks, vol. 10, pp. 75-83, 2017.
[44] “Regulation (EU) 2016/679 of the European Parliament and of the Council”, Legislation GOV.UK. https://www.legislation.gov.uk/eur/2016/679/article/4,(accessed Jan. 30, 2023)

Figure 17: An example of aggregation levels for the energy system, from the household up to the "portfolio" of a Flexibility Service Provider (FSP)



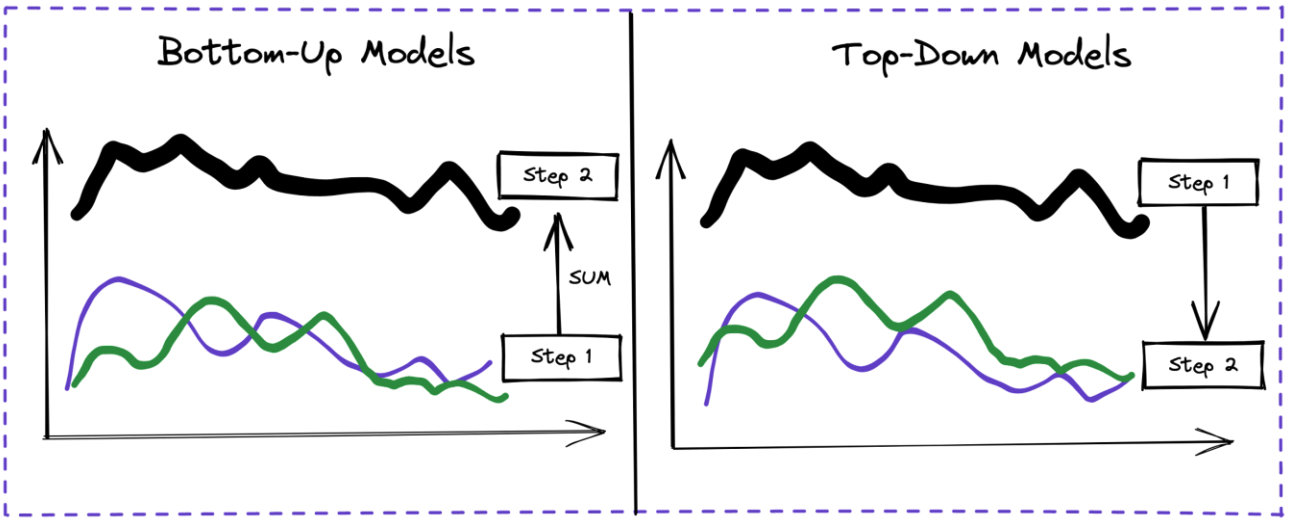
9.9 Bottom-up vs Top-down approach

Traditionally, hierarchical forecasting is achieved via a bottom-up or top-down approach:

- **Bottom-up**: Produce individual forecasts at the "lowest level" of the hierarchy and aggregate up.
- **Top-down**: Produce forecasts at a top-level and apply a 'proportion factor' or 'weightings' to obtain lower-level disaggregation.

Concretely, in the context of CrowdFlex, bottom-up models would forecast each "unit" (e.g., a household, or an asset such as an EV/HP) and then sum up to the portfolio level whereas top-down models would forecast the portfolio and then disaggregate into individual "units" (such as a unique household, or unique asset).

Figure 18: Graphical representation of Bottom-Up vs Top-Down forecasting models.



In both of these methods, information only flows one-way. Bottom up forecasts will be oblivious of top-level trends and vice versa. Literature from the supply chain sector, where these types of forecasting techniques are commonly used in stock forecasting, suggest

bottom-up forecasting produces a more accurate approach compared to top-down methods[45]. But the specific technique will depend on a number of factors:

- The variability of the lowest level profile
- Your use case (i.e., do you need the lowest level profile, or just the aggregated level profile)
- Data and technology requirements (e.g., it might only be computationally practical to forecast higher levels, if there are too many subcomponents at the lower level)

Other research presents an alternative called the 'middle-out' approach where each forecast is produced for all the hierarchies and a way to reconcile the forecasts[46]. Reconciliation is needed here because when producing forecasts for all hierarchies (e.g., household-level, grid-supply-point level, national level), there is no guarantee that aggregating household-level forecasts to national level will give the same numbers as the 'national-level forecast'.

9.10 Aggregation of Hierarchical Forecasts

Hierarchical forecasting however is still relatively straightforward for deterministic forecasts. It is a lot more challenging for probabilistic forecasts however. We cannot simply obtain the 95th-percentile forecast on aggregated level by aggregating the 95th-percentile forecasts of the lower levels. That would assume that all households are consuming at their 95th-percentile simultaneously which is not realistic.

There is literature which provides methods of aggregating probabilistic forecasts by computing the joint-distributions of all the hierarchies to obtain a coherent probabilistic forecast[47,48].

[45] **B. J. Dangerfield and J. S. Morris, 'Top-down or bottom-up: Aggregate versus disaggregate extrapolations', International Journal of Forecasting, vol. 8, no. 2, pp. 233–241, Oct. 1992, doi: 10.1016/0169-2070(92)90121-O.**

[46] R. J. Hyndman, R. A. Ahmed, G. Athanasopoulos, and H. L. Shang, 'Optimal combination forecasts for hierarchical time series', Computational Statistics & Data Analysis, vol. 55, no. 9, pp. 2579–2589, Sep. 2011, doi: 10.1016/j.csda.2011.03.006.

[47] S. B. Taieb, J. W. Taylor, and R. J. Hyndman, 'Coherent Probabilistic Forecasts for Hierarchical Time Series', in Proceedings of the 34th International Conference on Machine Learning, Jul. 2017, pp. 3348–3357. Accessed: Jan. 30, 2023. [Online]. Available: https://proceedings.mlr.press/v70/taieb17a.html

[48] S. B. Taieb, J. W. Taylor, and R. J. Hyndman, 'Hierarchical Probabilistic Forecasting of Electricity Demand With Smart Meter Data', Journal of the American Statistical Association, vol. 116, no. 533, pp. 27–43, Jan. 2021, doi: 10.1080/01621459.2020.1736081.