**7 System Evaluation and Experimental Results**

**7.1 Process Evaluation**

The AGILE approach to developing the system successfully maintained the cadence of development in completing most of the functional requirements identified in **chapter 3.4. FTR05** was not implemented which would have provided value to the user flow in exporting data for use in academic research papers. The software without this functionality still provides value to the academic user by automating the complex mathematical functionality used in analysis to evaluate the performance of different load forecasting models. Therefore, the AGILE approach provided incremental valuable features in the system to be used by the user, rather than incomplete features that did fulfil any of the use cases of the academic user.

Meetings with the stakeholder provided continual evaluation of developed features to ensure they met their expectations. Features were changed from initial phase of deriving requirements from the user specification. For example, the error distribution graph detailed in **US08** in **FTR03** was added during development. It was determined through meetings to be a good visualisation to compare the performance of different load forecasting models. A meeting with SONI did not take place because of the limited availability during the duration of development. This would have been useful to gain a commercial system operator’s perspective on whether the proposed system could be commercially beneficial in being a tool to assist their load forecasts.

**7.2 Implementation Evaluation**

The implementation of Dash framework (**see** **chapter 5.2)** in the systemsuccessfully adapted the proposed system design in **chapter 3**. The data centre architecture and the data transformation process were realised in the system by utilising the components used in the Dash lifecycle. The user interface components provided by the framework facilitated the components in the user interface mock-ups. The framework’s underlying Flask architecture and use of REST requests and responses for component interaction accommodated the requirement of the solution to be web ready which was identified as a potential usage by a user of the system.

Deployment of the system to the internet by Heroku provided a method to host the system on the internet. This was used to demonstrate the system to a wider audience who do not have to install system specific dependencies, and as a method for web environment testing. A continuous integration process to automatically deploy the system onto Heroku upon a push to the master branch was not used during implementation of the system. This would have included automated testing to verify the system is functioning correctly. Deployment to Heroku was manual. This was because the Git environment hosting the system did not support a build runner to run unit tests, and the acceptance tests were written as a test script to be followed manually. Hence, they could not be ran using an automated runner.

The system was designed to be data agnostic, not restricted to the Northern Ireland dataset used in analysis. The ‘holidays’ Python library used in preprocessing and analysis to map data entries to holidays specific to the user specified locale formed the basis of providing the same highlighting functionality for different datasets. The use of configuration enabled the list of linear regression models to be specific to the dataset, including models found in prior analysis using the system or other analysis tools to produce accurate load forecasts for the dataset. For example, load forecasting models evaluated in **chapter 1.7** were included in the configuration file for the SONI dataset. The datasets tested using the preprocessing functionality were New York and SONI datasets, which both had been cleaned and had the same density of load data. A dataset that was raw and had a much higher density was not available to test whether the system can still provide the same performant functionality as the other tested datasets. It cannot be guaranteed the system will work with all load datasets. Whether a dataset being compatible with a provided dataset can only be made once it is preprocessed successfully, and the system functionality is exploratory tested to produce visual and statistical results.

**7.3 Academic Benefit**

Automation of Analysis Tasks Assessment

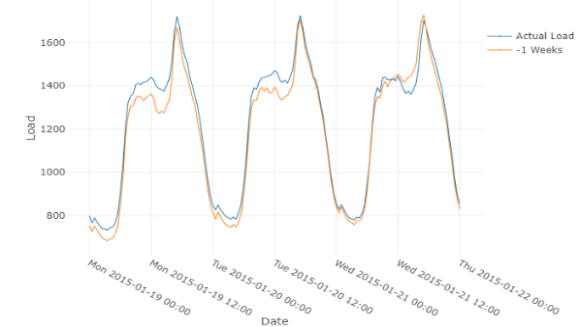
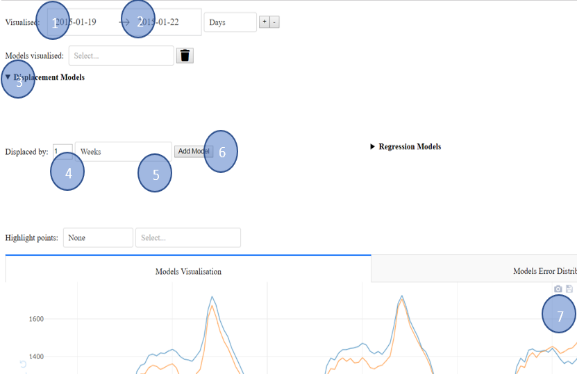
The solution successfully enables the academic user identified in **chapter 2** to automate the complex mathematical functionality required for load forecasting that they would have otherwise had to create manually in a specialist data analysis environment. They can successfully construct and evaluate the performance of different load forecasting models without having to know specific language syntax, as they are required to produce the visualisation of model performance in **Figure 7.1.** The user interface experience processes an academic can use in the system to produce the same visualisation is described in **Figure 7.2**

***Figure 7.2*** *The process of visualising of the performance of a SDLW displacement graph using the system.*

*(a) The user’s interaction with the system as follows:*

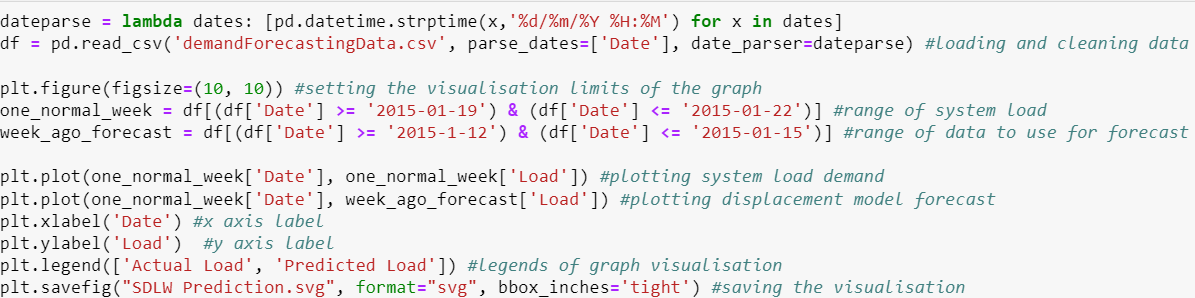
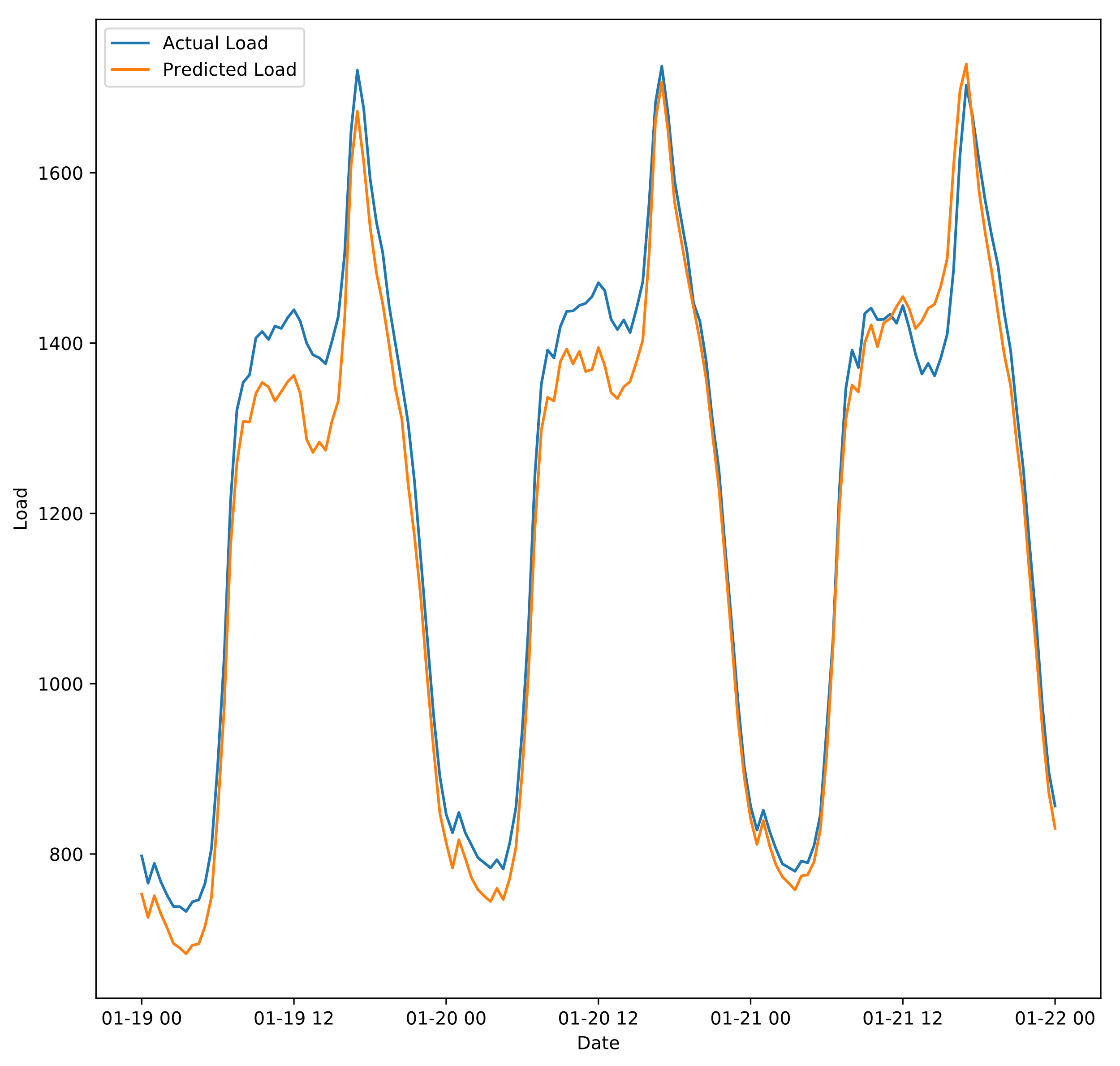
1. *Selecting load visualisation start date*
2. *Selecting load visualisation end date*
3. *Open displacement model carat*
4. *Click input field of displaced by and type in displacement value*
5. *Select displacement unit*
6. *Click add model*

*(b) The model performance visualisation.*



(a)

(b)



**Figure 7.1** The process of visualising of the performance of a SDLW displacement graph using Jupyter Notebook. (a) Python code to generate the visualisation. (b) The model performance visualisation.

(a)

(b)

General understanding of load forecasting

The system functionality does not replace the functionality provided using other data analysis environments. The construction of displacement models in the system is flexible in changing the displacement value and unit, but the construction of linear regression models is limited to the configuration structure defined in the README that maps to their implementation in the system. Analysis tools enable an academic user to be experimental in constructing linear regression models, for example including experimental variables not initially in the dataset. Additionally, the system does not enable full customisation of the presentation and layout of visualisations produced like analysis libraries provide through functions calls, passing in arguments that change the parameters.

The real value of the system to the academic user is that it does not require programming language knowledge to change the visualised date range of the dataset to evaluate the performance of load forecasting models on different subsets of the date range. They can also test the performance against a subset data set. Therefore, the system is not a replacement for the analysis tools an academic user can use. It is companion tool that enables them to compare and evaluate the performance of models they have evaluated in their own or other researcher’s analysis to perform optimally with the dataset.

Validation of Use Cases Fulfilled

The system has met most use cases identified for the academic user in **chapter 3.** Each use case is listed below with evidence of it being fulfilled by the solution and a short description:

|  |
| --- |
| **1 Arrange and clean a provided dataset**  The user can use a dataset that is organised in accordance with the defined the data contract in **chapter 3** and load it into the preprocessing program. There is no error handling for the user input, with the expectation the user will choose from options listed in the ‘holidays’ library documentation referenced in the console. |
| **Figure 7.3** An example of running the preprocessing program to augment the dataset with required variables for highlighting and model construction functionality. |
| **2 Visualise Dataset**  The user can visualise a subset of the dataset’s load values and compare the load with other characteristics of the dataset by selecting a column in the characteristic’s dropdown. The characteristics dropdown includes all the columns in the dataset, including those augmented to the dataset for use with date highlighting functionality and linear regression model construction. The value to an academic user is that it allows them to identify variables that correlate with the load visually and variables used in linear regression models. The highlighting functionality enables the user to identify trends specific to type of days. |
| **Figure 7.4** An example of visualising a subset of the load dataset and comparing it with a characteristic of the dataset (Temperature). The highlighting functionality is also visualised, with data points of day Monday being highlighted in red. |
| **3 Add Models**  The system enables the academic user to add displacement and linear regression load forecasting models to evaluate their performance. There is no limit to the number of models the user can add, it is up to the academic user to decide whether the visualisation has the appropriate presentation of data. |
| **Figure 7.5** The two flows to construct a forecasting model in the solution. |
| **4 Compare Model Performance**  The system enables the user to compare the load forecasting performance of models they have added visually by assessing the accuracy of the load forecast with the actual load |
| **Figure 7.7** An example of comparing the forecasting performance of test and visualised data through an error graph visualising the distribution of errors within the dataset. The metric ‘90% Threshold Absolute Percentage Error’ accompanies the graph as a statistical method to compare the load forecasting performance.  **Figure 7.6** An example of adding a SDLW displacement and a linear regression model to visualise the model load forecasts. The visualisation and accompanying error statistics below are used to compare the performance of the model on the specified visualised and test data subset. |
| **5 Export Data**  The proposed process of exporting both visual data as vector graphs and statistical data results in excel format from the system was not implemented. The academic user can however export PNG visualisations which can be used in their academic papers. |
|  |
| **Figure 7.8** Examples of static PNG visualisations produced by in-built Plotly functionality.  *(a) Load and model forecast visualisation*  *(b) Model forecast error visualisation* |

**7.5 Further Work**

Analysis

The scope of the analysis focused on displacement and linear regression load forecasting models. LASSO variable selection to rank explanatory variables has not been previously explored in other SONI dataset Load Forecasting papers. The analysis could be continued with other methods for variable selection, for example Elastic Net which includes LASSO and Ridge regression to gain the benefits from using both. Non-linear models have not been explored by this analysis which can provide more accurate load forecasts than linear regression models [2].

Use of Implementation Libraries

The dissertation provides an appreciation of the challenges encountered in implementation to guide developers in extending the code base. For assistance in implementing new functionality to the system the Dash framework and the Plotly visualisation library both have detailed documentation online [1][2]. They are both being updated with regular updates, therefore their change notes in their repositories should be viewed on a regular basis to keep up to date with new features that could be used to refactor or add new functionality [3]. This has been recognised previously in **chapter 5.4** with new functionality enabling multiple outputs that can be utilised to refactor the existing implementation.

Commercial Benefit of the System

The system is of commercial benefit to a system operator as they can use the system to have a greater visibility of the performance of a load forecasting model on historical data. They can then use the produced results in their decisions making regarding their day ahead load forecasting to make their operations more efficient. A way to make the system more commercially valuable to system operators is to adapt the visualisations and interaction to forecast system load for a given day using an automated model selection process that uses the best model or a product of models to generate an accurate load forecast. The proposed wire frames of this functionality are in **Appendix.**

New Forecasting Models

The implementation of new types of models e.g. non-linear and neural network models, into the system would require changes to the system. A new interface in the layout to construct the model analogous to the current interface for models in **Figure 7.5**, a new file with a predict method and additions to the configuration structure would also be required. They would still however still be included in the same dropdown for the user to modify the list of currently added models. The functionality to adding model traces to a graph is reusable through unified utility files with shared functionality. The utility files also make the implementation of new graphs and statistical error metrics more efficient by eliminating the need to write the same ‘boiler-plate’ code again to deliver core functionality.

Internationalisation

The strings used in the user interface to label functionality is contained within the system layout. The layout should have instead referenced a configuration file with a list of strings that contain language translations. This would make the solution capable of being made accessible to an international non-English speaking audience by adding new foreign language translations to the configuration file.

Additional Functionality

Other functionality was proposed for the system, but was not chosen for development because of time constraints:

* A tool with a graphical user interface to create linear regression models for use in the solution could be more intuitive to the user than having to manually modify the JSON configuration.
* The load dataset decoupled from the solution and stored remotely. Subsets of the dataset can then be accessed through an endpoint instead of loading all the data into the host system’s memory at once.
* Minor bugs and tasks identified during implementation in the ‘Backlog’ and ‘Parking Lot’ section of the system’s Trello board [4].

**Appendix**

All the industrial user considerations

**Citations**

[1] <https://dash.plot.ly/>

[2] <https://plot.ly/python/user-guide/>

[3] <https://github.com/plotly>

[4]<https://trello.com/b/RrMXYpy1/short-term-electricity-load-forecasting-visualisation-and-interaction-tool>