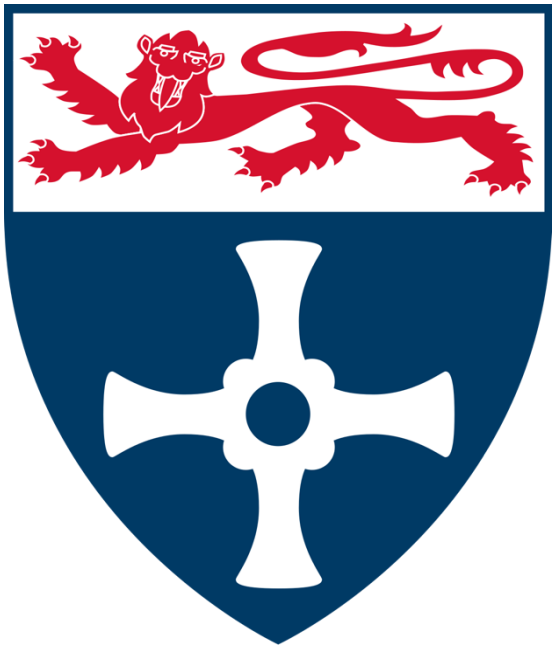


Forecasting household electricity consumption with machine learning techniques using smart meter data

BSc Computer Science

Kenneth Jeremy Cajigas – 170450661
Supervisor: Dr Yu Guan
May 2022

Word Count: 11,980

Declaration

I declare that this dissertation that has been submitted to Newcastle University for the award of BSc Computer Science contains my own work except where otherwise stated and is clearly referenced. In addition, the work within this dissertation has not been submitted as part of any other degree.

Signed: Kenneth Cajigas

Student number: 170450661

Date: 27th May 2022

Abstract

Managing energy today has become an increasingly relevant topic, especially within the UK where many of the population are now facing increased energy prices during the cost of living crisis. With the continued rollout of smart meters by the UK government, electricity consumption data is now more readily available than ever. A key tool that is not widely available to consumers to be able to manage their future consumption is using predictive analytics tools such as machine learning. The aim of this dissertation is to apply the UK Power Network London Smart Meter dataset to various machine learning techniques, developing them to working models and evaluating their effectiveness to be able to generalise well for forecasting electricity consumption to any household. The dataset contained household data from London between November 2011 and February 2014 and had a vast amount of information associated with each household and their corresponding electricity consumption which was pre-processed according to the suitability of the features included. Models developed include Multiple Linear Regression, K-Nearest Neighbours, Support Vector Regression and Long Short Term Memory. They were evaluated with several metrics: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and finally the R2 score.

Acknowledgements

This has been an extremely rewarding but also challenging dissertation for me. I would like to thank my supervisor Dr Yu Guan for his guidance, feedback, and support to relevant resources to this project during this dissertation process.

I would also like to express my gratitude for my friends and my family, especially my parents Maria Cajigas and Warlito Cajigas for providing the support whenever I needed to fall onto them.

Finally, I would like to thank my partner Viorel Grigorita, who has been a continuous pillar and motivation for me throughout this project from start to end. You helped me to realise my potential and kept me going when I felt I couldn't. Without your love and support I would not have been able to get to where I am today.

Table of Contents

Table of figures

Table of tables

Chapter 1: Introduction

This chapter provides an insight into the context, motivation, and current problem that this dissertation looks to address and provide an understanding by the end of it.

1.1 Motivation

Context:

Energy is an essential need to everyone today, it powers our lives and the critical technology around us. Electricity consumption continues to increase globally and in 2021 a 5% increase in consumption was seen from previous years [1]. This trend is only set to continue to grow. According to EE, the average UK smart home will have at least 50 connected devices by 2023 [2]. However, at the time of writing the UK is experiencing an energy crisis resulting in prices for electricity steeply increasing, causing concerns on affordability for many consumers. [3] With an aim to monitor consumption and reduce emissions through climate change policies, the UK Government is targeting all 26 million homes to be fitted with smart meters by 2025 [4]. As a result of this, energy usage data is now more accessible than ever and with breakthroughs being made with predictive analytic technologies, there has never been a better time to invest in technologies that can help to forecast electricity demand on both the consumer and supplier side.

Current Problem:

Corresponding to the cost of living crisis happening within the UK at the time of writing [5], consumers don't have many ways to make use of the latest data from their own smart meters to be able to reduce their consumption and in the long term reduce overall costs. In most installations, consumers are provided with an In-Home Display (IHD) which shows their current usage and how much it costs. In a recent study [6] it was found that only 45% of the consumers surveyed looked at it most days and that around 47% of the consumers felt that having a smart meter improved their understanding of their energy consumption over the year. This is a good starting point for raising awareness but more needs to be done in this area to help consumers on a day-to-day basis to be aware of their usage on a day, what they're currently predicted to use and what steps they can take to reduce their consumption. Currently, most energy providers only provide a yearly price prediction of what a consumer is expected to have based on their current usage, but that is all and more could be done within this area to provide additional support for consumers.

1.2 Rationale

This dissertation looks to provide a starting point as to how smart meter data can be used in machine learning models to provide a forecast on consumer household usage by making use of crowdsourced energy consumption data of households in London. In a field study [7], it was noted that consumers of smart meters struggled to see what impact their behavioural changes had and that there was a lack of benchmarking for how their consumption was on a certain day. In addition to this, there has been little research available to investigate this area of consumer household electricity prediction on an individual level and this is the purpose of this dissertation. This dissertation attempts to provide one solution to the problem through investigating the role of machine learning models for generalising in forecasting for household consumption to be able to help consumers to identify their usage trends based on previous data and identify what their 'natural baseline' is over a certain period.

1.3 Aim and objectives

The overall aim of this dissertation is to investigate the use of machine learning techniques to forecast future short term energy consumption using smart meter data. I intend to use a dataset which contains crowdsourced household smart meter electricity consumption data in London and train different types of algorithms to see their performance in generalisation for providing an accurate prediction for households. To accomplish this my objectives are as follows:

1) **Review and summarise published findings in the types of machine learning algorithms that have been applied to forecasting short term electricity consumption**
   Reviewing existing work that has been done within the area of energy forecasting would allow me to develop an understanding on previous approaches taken around the topic area. This would provide insight into the types of machine learning algorithms that have been investigated, how accurate they were at providing a prediction of energy consumption and opportunities available for improvement based on issues that were found.

2) **Evaluate the types of machine learning algorithms referenced and identify those that will be implemented and applied to the crowdsourced London household smart meter dataset**
   In this objective based on my findings in my previous objective, I will evaluate each of the algorithms and select the types of machine learning algorithms that are relevant to this dissertation and have been seen to be effective with accurately forecasting energy consumption. I will also look to understand how each of the algorithms selected work to generate an output from their inputs.

3) **Pre-process and analyse the dataset and investigate correlations between features to be provided as input for training each of the machine learning algorithms**
   Pre-processing the dataset is important for each of the algorithms to be able to accurately provide predictions as output. Analysing the dataset would provide a deeper understanding of the relationship between various factors that can affect electricity consumption on a certain day and how trends of consumption can vary over a year. It would also provide more clarity as to which features will be passed in as input to each model to improve efficiency.

4) **Train each of the chosen machine learning algorithms and develop working models**
   The processed dataset will then be passed into each of the selected machine learning algorithms for training and a model will be developed for each type. This objective will have been completed once a model has been developed with each of the chosen machine learning algorithms.

5) **Evaluate the effectiveness of generalisation in the dataset for predicting short term electricity consumption for households in each type of algorithm chosen**
   This final objective looks to provide the results of how each of the models performed against test data on certain households including my own using statistical performance metrics such as RMSE Root Mean Square Error (RMSE), Mean Squared Error (MSE) and Mean Absolute Error (MAE). It will also reveal the best performing model out of the algorithms chosen and provide the foundations into how this work could be extended within the energy industry.

Chapter 2: Current research

This chapter provides an insight into the existing and similar works within the area of energy forecasting, the machine learning algorithms used, and the results achieved from applying them to relevant datasets.

2.1 Existing work with machine learning algorithms applied to energy forecasting

A Kell et al [8] made use of a public data set of around 709 Individual Irish homes with their electricity consumption for short term load forecasting. They investigated the use of clustering in grouping households with similar consumption patterns with an aim to improve accuracy in the models they produced for forecasting energy consumption for a household. The machine learning algorithms models developed consisted of Support Vector Regression, Random Forest, Artificial Neural Network and Long-Short Term Memory Neural Network. They found that with K-Means clustering grouping of households prior to training each of the models, accuracy improved. It was also noted with calendar attributes such as hour, day of the month, day of the week, month and public holidays, the accuracy increased by up to 6% with the artificial neural network.

S. Sulaiman et al [9] discusses their methodology on developing an Artificial Neural Network for forecasting short term energy consumption. The dataset consisted of a single home consisting of minute by minute electricity consumption data for 3 months. Their paper discussed the methodology and steps involved in developing a multi-layer feed forward artificial neural network. It was also particularly useful to see the structure of how they laid out their ANN, which could potentially be applied to this dissertation if it was decided that one of the algorithms to use would be an ANN. With the model that they had developed, a 70.54% accuracy was achieved.

K. Gajowniczek et al [10] considered two different types of machine learning algorithms for forecasting short term load forecasting 24 hours ahead on an individual household level at an hourly basis. The data used for forecasting included a single household consisting of two adults and a child during a period of 60 days between 29th August 2012 to 27th October 2012. Like the previous paper reviewed, their feature vector was constructed of the previous 24 hour load: the average, maximum, minimum and range of the previous 3, 6, 12, 24 hours, the day of the week and finally the temperature observed in each hour. They noted that from their own literature findings, it was concluded that time series analysis techniques weren't possible to apply to the dataset effectively because of the high volatility seen. As a result, artificial neural network and support vector machine algorithms were chosen for model development in short term electricity forecasting. They were able to achieve reasonable prediction accuracy, where the artificial neural network had 65% accuracy and support vector machine had 64%. This paper provided an insight from their concept of what forecasting on an individual basis can be expected to be like and how more factors being included in the used dataset such as weather data could provide improvements.

Z. Camurdan et al [11] developed three machine learning algorithms models for forecasting electricity demand related to the market in Turkey. They looked at forecasting daily and showed results on a yearly, monthly, weekly and daily basis. The dataset used was publicly available and had 56 features related to the electricity market at the time. To reduce the size of the dataset and improve model performance, they used a technique called mutual information to select the most relevant features to forecasting electricity demand. This could be another method I would consider during the pre-processing and analysis of the dataset I plan to use when undergoing feature selection. For evaluation of each of the three models developed, four metrics were considered: Mean Absolute Percent Error (MAPE), R Square, Mean Absolute Error (MAE) and Mean Squared Error

(MSE). Overall, all three models performed well – based on the R squared achieved, Decision Tree had 97% accuracy whereas Linear Regression and Random Forest both had 98% accuracy.
In summary, this paper was very insightful as to what approach can be used for feature selection as well as which metrics are relevant for evaluating model performance.

S. Chadoulos et al [12] proposed a novel architecture algorithm with the aim to create a single deep learning model which has the ability to generalise well with short term load forecasting consisting of a dataset of around 1000 US households with hourly energy measurements between 2012 to 2019. They highlighted key challenges and issues around developing a model for generalising in forecasting, most notably how a model being trained on data from a single household would be unable to generalise well and cannot make good enough predictions for new consumers. Further to this, the cold start problem was highlighted as another problem with this approach as modelling would not be possible if a single household did not have enough historical consumption data. They found that one of the key benefits of using a deep learning approach was because of the way a model would be able to effectively discover common training patterns between consumers and provide accurate forecasts. Their training dataset consisted of multiple households and clustering techniques to group household energy profiles together and their model was a combination of a Recurrent Neural Network encoder and a Multilayer Perceptron. Overall, they were able to achieve a Mean Absolute Percentage Error of 10.1% on known data and 12.5% on new unseen data. This is most notably the most similar work of what I aim to do within this work on my dissertation.

Chapter 3: Machine learning algorithms

This chapter provides an understanding to the selection process of which machine learning algorithms will be used within this dissertation as well as an understand to how they work behind the scenes.

3.1 Overview of referenced machine learning algorithms in research

The below table is a summary of what which machine learning algorithms were discussed in the existing research that was investigated in the previous chapter as well as other key points such as the performance metrics used.

Dataset	Research focus	Machine Learning algorithms used	Performance metrics	Best model
709 Irish households between July 2019 and December 2010 [8]	Short term load forecasting – 30 minutes ahead using clustering profile techniques	- Random Forests - Neural Networks - Long Short-Term Memory - Support Vector Regression	- RMSE - MSE - MASE - MAPE	Random Forests – Achieved a MAPE between 5-6% between various clusters
Single consumer household between May 2012 and July 2012 [9]	Short term load forecasting - Forecasting each hour 24 hours ahead	- Artificial Neural Network	- Accuracy	Artificial Neural Network – Achieved 70.5% accuracy
Single household between August 2012 and October 2012 [10]	Short term load forecasting - Forecasting each hour 24 hours ahead	- Artificial Neural Network - Support Vector Regression	- MSE - Accuracy	Artificial Neural Network – Achieved 65% Accuracy, 0.09 MSE
Electricity demand in Turkey between 2011 and 2016 [11]	Forecasting for energy demand daily for one year	- Linear Regression - Decision Tree - Random Forest	- R squared - MAPE -MAE - MSE	Random Forest – 97% accuracy, 1.4% MAPE
Pecan Street dataset in the UK between 2012 to 2019[12]	Short term forecasting – Forecasting hourly consumption forecast	- Novel Architecture involving RNN and MLP	- MAPE - R Squared - MSE	- Novel model developed had achieved a 12.5% MAPE in forecasting for unseen households

Table 1: Machine Learning Algorithms referenced in background research

3.2 Evaluation of referenced algorithms

The below Machine Learning algorithms were mentioned and applied on various datasets for the purpose of energy forecasting. Their feasibility within this dissertation is considered and evaluated below based my understanding level, the results that they had produced in referenced datasets above as well as the speed of training and testing of the algorithms with consideration to the time constraints that this dissertation has.

Machine Learning algorithm	**Understanding level** Simple, Medium, Complex	**Speed of training and testing** Slow, Medium, Quick
Multiple Linear Regression [13]	Simple: Representation is a linear equation that combines a set of features to the target feature	Quick for training and testing. All points are put onto a line of best fit which is then used to predict the target feature. Training complexity is: $O(p^2n + p^3)$ Testing complexity is: $O(p)$ Where n is the number of samples and p is the number of features.
Random Forest [13]	Medium: Representation is composed of different decision trees with different paths that eventually leads to the target feature to predict.	Fast to train, however slow to make predictions on tests. The larger the number of trees, the slower it becomes. Training complexity is: $O(n^2pn_{trees})$ Testing complexity is: $O(pn_{trees})$ Where n is the number of samples, p is the number of features and ntrees is the number of trees.
Neural Network [14][15]	Complex: Representation is composed with various formulas for each node within the model being composed of input data, weights, bias, and an output.	Slow on training and tests for predictions. A neural network involves using an optimisation method to be able to map the inputs and weights to the target feature. Training and testing complexity is: $O(pn_{l_1} + n_{l_1}n_{l_2} + \dots)$ Where p is the number of features and n is the number of samples.
Decision Tree [13][16]	Medium: Like the Random Forest algorithm, its representation is composed of several nodes and paths which eventually lead to the target feature to predict.	Slow to train, however fast when on tests when making predictions. Once a model has been trained, each sample just needs to follow the tree to an end leaf which would contain the value for the target feature. Training complexity is: $O(n^2p)$ Testing complexity is: $O(p)$

Machine Learning algorithm	**Understanding level** Simple, Medium, Complex	**Speed of training and testing** Slow, Medium, Quick
		Where n is the number of samples and p is the number of features.
k-Nearest Neighbours [13][17][18]	Simple: kNN is simple to represent as it is just the input data provided into the model laid out by the input features against the target feature. K is a set value which is then used to predict a target feature based on the mean of the nearest K points.	Slow to train as well as on tests when making predictions. As the number of samples and features increase, the longer it takes for the algorithm to be able to perform training and testing. For every prediction that is needed from the algorithm, it needs to go through the whole dataset again to be able to identify the nearest points and then provide an output. Training and testing complexity is: $O(np)$ Where n is the number of samples and p is the number of features.
Support Vector Regression [13][19] [20]	Simple/Medium: SVR is an extension to Multiple Linear Regression and works by converting non-linear inputs to a linear problem and looks to optimise a hyperplane that is created to separate input data into classes.	The speed of training depends on the dataset itself and this is something that needs to be considered when evaluating model performance. If the data is noisy and has overlapping points, then it will take longer to be able to identify the right hyperplane and decision boundaries for separating points. Training complexity is: $O(n^2p + n^3)$ Testing complexity is: $O(n_{sv}p)$ Where n is the number of samples and p is the number of features.
Long Short Term Memory [13][21][22]	Medium/Complex: LSTM is a type of Recurrent Neural Network which can learn from previous historic values for sequence prediction problems. LSTM algorithms are a complex area within deep learning and their units contain various structures that allow it to perform well with many types of regression problem.	The speed of training depends on the size of the samples that are provided as sequences to the model as well as how big the timesteps set are. From background research, it has been said that LSTMs are recommended not to have timesteps larger than 400 prior days with large LSTM model. Upon looking for the algorithm complexity, I was unable to find any reference to it.

Table 2: Evaluation of referenced Machine Learning Algorithms

3.3 Algorithm selection

Following from the evaluation in the previous section, the following machine learning algorithms have been selected to be investigated and applied to the dataset to be used within this dissertation based on their ease of understanding as well as the speed that they are able to perform at considering time restraints for this dissertation.

3.3.1 Multiple Linear Regression (MLR)

Multiple Linear Regression is a type of machine learning algorithm which builds upon simple linear regression, however instead of just one single independent feature which is used to make predictions on a target feature, it is used to provide an insight into the relationship between two or more independent features and one target feature [23].

$$y = w_0 + w_1x_1 + w_2x_2 + ...+ w_nx_n$$

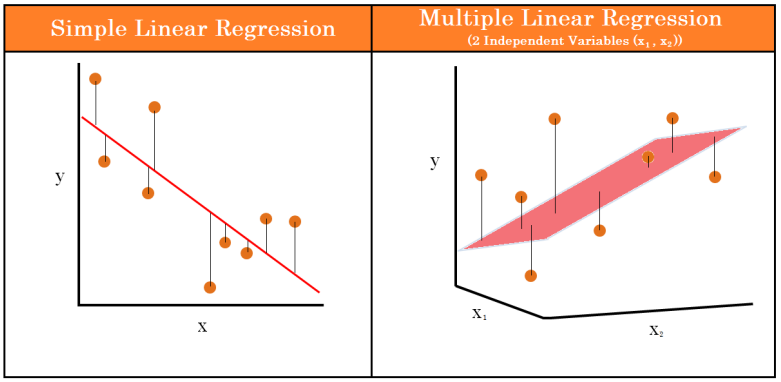


Figure 1: Multiple Linear Regression Representation with formula [24]

3.3.2 k-Nearest Neighbours (kNN)

kNN is another type of machine learning algorithm, however it does not rely on a linear based relationship and is instead distanced based. kNN works based on 'feature similarity' which allocates a new point with a predicted value to be based around what the nearest points or the new point's neighbours have. K is a value that can be set when building the model. Depending on the set value of k, the average of the nearby k data points are calculated to determine the final value of the new predicted point. kNN calculates distance is between points as the square root of the sum of squared differences between a new point and an existing point. This is also known as the Euclidean distance [25]. Below is the graphical representation of KNN for when K is set to 3.

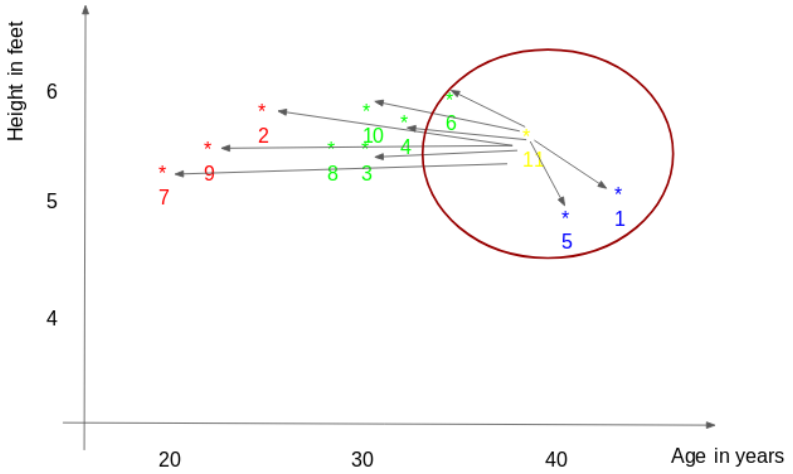


Figure 2: kNN Representation with 3 nearest neighbours [25]

3.3.3 Support Vector Regression (SVR)

Support Vector Regression is essentially the same concept as the Support Vector Machine algorithm however are used specifically for regression problems [26]. It makes use of a combination of a Kernel, Hyperplane, and a Decision Boundary. A hyperplane which is a line that helps the algorithm to be able to provide a prediction for a new point that has not been seen before. It makes use of a Kernel which finds the hyperplane in the first place, without increasing the computational cost needed for this data increase. Finally, the decision boundaries are composed of two lines which are of distance *+e* and *-e* from the Hyperplane which the algorithm has decided that at *e* distance data points which are closest to the hyperplane are within that boundary line. The distance between the two lines is known as a margin. The Support Vectors are the data points which are closest to the decision boundaries.

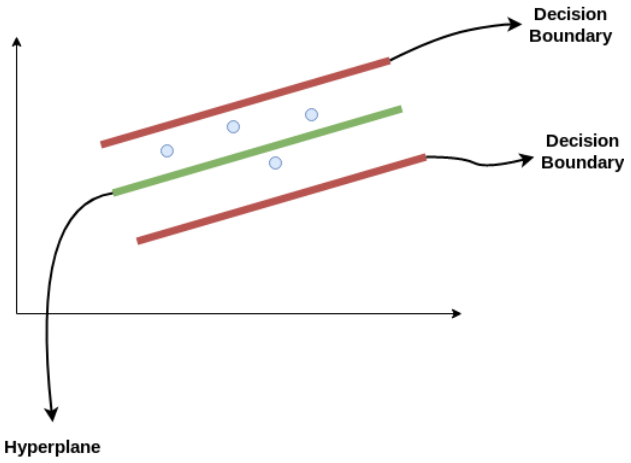


Figure 3: Support Vector Regression representation [26]

3.3.4 Long Short Term Memory Recurrent Neural Network (LSTM)

A Long Short Term Memory (LSTM) network is a type of deep learning which is also a type of Recurrent Neural Network (RNN) which are capable of learning long term dependencies [27]. With a variety of different problems, LSTM models have been found to provide high performance results. With the same principle of how RNN's work, a LSTM machine learning algorithm works within networks with loops which allow them to have the ability to remember certain pieces of information.

However, a common issue with using a standard RNN is that they can perform quite poorly on tasks which involve long-term dependencies; These usually are where datasets that contain a target feature which depend on input features with historic data. The structure of a LSTM is seen below. They have four different neural network layers, where each carry a whole set of data from the output of one previous LSTM node to be passed as input to another.

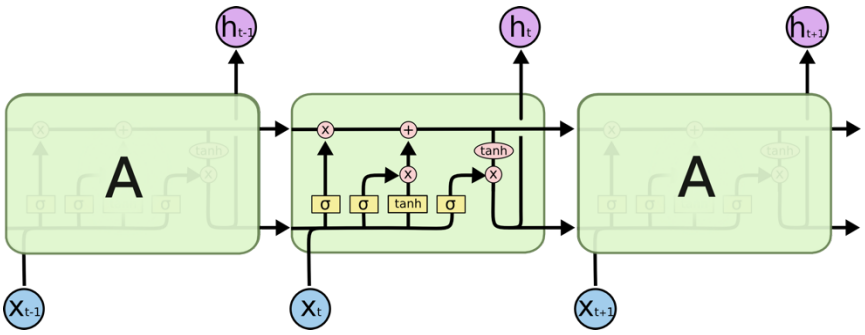


Figure 4: Structure of a LSTM unit [28]

Chapter 4: Dataset understanding

This chapter provides the overview into the dataset that was used to be applied to the selected machine learning models. It provides insight to the structure of each dataset; how null values were dealt with as well as the trends and correlations seen between features for feature selection.

4.1 Introduction to the dataset

The selected algorithms will be trained using the UK Power Networks Smart Meter Energy Consumption dataset which contains up to 5567 household's consumption data in London. These households were part of the UK Power Networks Low Carbon London project. The dataset contains data recorded between November 2011 and February 2014 and is part of a bigger dataset found on the public dataset website Kaggle [29], which also contains daily weather and public holidays during the same period. Also, it is noted that in the dataset, there are two tariff groups that households were on: the first contains around 1100 households who were on Dynamic Time of Use (ToU) tariffs, leaving around 4500 households on standard (Std) tariffs. For the dataset within this dissertation, only those on standard tariffs will be considered. This dissertation will be completed with the use of Google Colab, which provides a platform to run executable cute in a Jupyter notebook style. [30]

4.2 Dataset structure

Initially, the dataset is composed of 19 different files. These files were filtered out based on the focus of this dissertation which was to forecast energy consumption daily. 4 of the 19 files were selected and are discussed within the next section on their initial structure and then for pre-pre-processing which eventually leads to various datasets used to train the machine learning algorithms selected.

4.2.1 Consumption data

	LCLid	day	energy_median	energy_mean	energy_max	energy_count	energy_std	energy_sum	energy_min
0	MAC000131	2011-12-15	0.4850	0.432045	0.868	22	0.239146	9.505	0.072
1	MAC000131	2011-12-16	0.1415	0.296167	1.116	48	0.281471	14.216	0.031
2	MAC000131	2011-12-17	0.1015	0.189812	0.685	48	0.188405	9.111	0.064
3	MAC000131	2011-12-18	0.1140	0.218979	0.676	48	0.202919	10.511	0.065
4	MAC000131	2011-12-19	0.1910	0.325979	0.788	48	0.259205	15.647	0.066

In the *consumption_data.csv* file, it is structured by the unique meter identifier, the date and time and the electricity usage in kWh in various types of statistical format for each day. Initially in its current form, its shape is at 3510433 rows with 9 features.

It was mentioned in the introduction of the dataset that it had contained up to 5567 households at one point. The nationwide rollout of smart meters began in 2011 [31], which meant that because of this the wealth of data was not immediately accessible. It can be seen in the graph below that the number of households included within the trial increased as time went on, which is also in correlation to the continued rollout by energy companies installing smart meter in more consumer homes in line with the government target to have the ability to offer a smart meter to all 26 million UK homes by 2025.

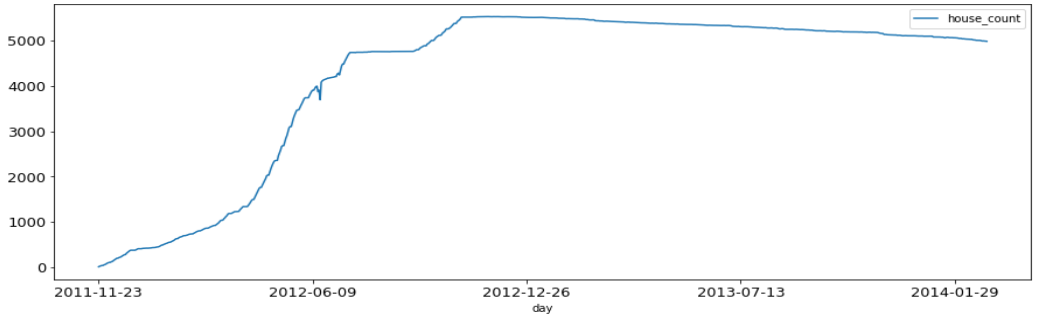


Figure 6: Household consumption recording count over the dataset period

4.2.2 Tariff and ACORN data

Tariff and ACORN data were stored in the *informations_households.csv* file, which contains each of the unique household LCLids, the type of tariff they were on, which Acorn grouping they were identified as and the specific ACORN group they were on. The file indicated which block they were in - however this was not relevant here as each individual block file was merged into one big .csv file. Overall, the file shape is 5566 rows with 5 features.

```
household_information = pd.read_csv(full_path+'/daily/informations_households.csv')

household_information.head()
```

	LCLid	stdorToU	Acorn	Acorn_grouped	file
0	MAC005492	ToU	ACORN-	ACORN-	block_0
1	MAC001074	ToU	ACORN-	ACORN-	block_0
2	MAC000002	Std	ACORN-A	Affluent	block_0
3	MAC003613	Std	ACORN-A	Affluent	block_0
4	MAC003597	Std	ACORN-A	Affluent	block_0

Figure 7: Household information DataFrame structure

The Acorn_grouped feature is of significant interest here for further analysis on when looking into the trends of how energy consumption differs between each of the groups, as well as the impact on the performance of the models developed later to see if models developed can be improved. Within the dataset, the households are categorised and split into 3 distinct groups that will be analysed further down in this dissertation.

```
household_information.Acorn_grouped.unique()
```

```
array(['ACORN-', 'Affluent', 'Comfortable', 'Adversity', 'ACORN-U'],
      dtype=object)
```

Figure 8: Distinct Acorn groups in the dataset

4.2.2.1 Acorn Grouping clarification

Acorn grouping is the segmentation of households within the UK [32]. It classifies each postcode in the country into one of the 62 possible types, which are then part of 6 overall Acorn groups: Affluent achievers, rising prosperity, comfortable communities, financially stretched, urban adversity and not private households.

Category	*Group*	*Type*
1. Affluent Achievers	A	Lavish Lifestyles	1-3
	B	Executive Wealth	4-9
	C	Mature Money	10-13
2. Rising Prosperity	D	City Sophisticates	14-17
	E	Career Climbers	18-20
3. Comfortable Communities	F	Countryside Communities	21-23
	G	Successful Suburbs	24-26
	H	Steady Neighbourhoods	27-29
	I	Comfortable Seniors	30-31
	J	Starting Out	32-33
4. Financially Stretched	K	Student Life	34-36
	L	Modest Means	37-40
	M	Striving Families	41-44
	N	Poorer Pensioners	45-48
5. Urban Adversity	O	Young Hardship	49-51
	P	Struggling Estates	52-56
	Q	Difficult Circumstances	57-59

4.2.3 Weather data

Within the *weather_daily_darksky.csv* file, it contained a wide variety of features not limited to but including: the temperature, humidity, visibility, and wind speed. This data was sourced from the darksky api during the same period as the consumption data from UK Power Networks. Overall, in its current format, it had a shape of 882 rows with 32 features.

```
daily_weather = pd.read_csv(full_path + '/daily/weather_daily_darksky.csv')

daily_weather.columns
```

```
Index(['temperatureMax', 'temperatureMaxTime', 'windBearing', 'icon',
       'dewPoint', 'temperatureMinTime', 'cloudCover', 'windSpeed', 'pressure',
       'apparentTemperatureMinTime', 'apparentTemperatureHigh', 'precipType',
       'visibility', 'humidity', 'apparentTemperatureHighTime',
       'apparentTemperatureLow', 'apparentTemperatureMax', 'uvIndex', 'time',
       'sunsetTime', 'temperatureLow', 'temperatureMin', 'temperatureHigh',
       'sunriseTime', 'temperatureHighTime', 'uvIndexTime', 'summary',
       'temperatureLowTime', 'apparentTemperatureMin',
       'apparentTemperatureMaxTime', 'apparentTemperatureLowTime',
       'moonPhase'],
      dtype='object')
```

Figure 10: Daily Weather DataFrame structure

4.2.4 Public holiday data

This dataset of public holidays from the *uk_bank_holidays.csv* file contained of the dates and the type of bank holiday during the period in the dataset between November 2011 and February 2014. At its current form, it has a shape of 25 rows and 2 features.

	Bank holidays	Type
0	2012-12-26	Boxing Day
1	2012-12-25	Christmas Day
2	2012-08-27	Summer bank holiday
3	2012-05-06	Queen?s Diamond Jubilee (extra bank holiday)
4	2012-04-06	Spring bank holiday (substitute day)
5	2012-07-05	Early May bank holiday
6	2012-09-04	Easter Monday
7	2012-06-04	Good Friday
8	2012-02-01	New Year?s Day (substitute day)
9	2013-12-26	Boxing Day

4.3 Data preparation and cleaning

To be able to get a single combined dataset that would be passed into each of the models for training, data cleaning is performed on the data, involving the following process:

1. Checking that data is valid and correct
2. Dealing with null values and which values to replacement with
3. Creating new columns for better understanding and efficiency in the models
4. Scaling / Normalising the data for machine learning algorithms which require it

4.3.1 Preparing daily consumption data

For daily consumption data, the day column is put into a pandas readable datetime format and then the data is sorted by day. Only the LCLid (household identifier) and *energy_sum* columns are kept as relevant columns for the final dataset.

```
#Change day into readable format
daily_consumption['day']=  pd.to_datetime(daily_consumption['day'],format='%Y-%m-%d').dt.date

#Reset index so that it is ordered by day rather than by LCLid
daily_consumption = daily_consumption.sort_values(by=['day']).reset_index(drop=True)

#Keep only relevant columns which is day and energy_sum
daily_consumption = daily_consumption[['LCLid', 'day', 'energy_sum']]

daily_consumption.head(10)
```

	LCLid	day	energy_sum
0	MAC000149	2011-11-23	2.287
1	MAC000154	2011-11-23	5.798
2	MAC000156	2011-11-23	6.523
3	MAC000150	2011-11-23	9.254
4	MAC000147	2011-11-23	3.036
5	MAC000146	2011-11-23	5.619
6	MAC000157	2011-11-23	7.408
7	MAC000152	2011-11-23	5.969
8	MAC000151	2011-11-23	3.273
9	MAC000145	2011-11-23	8.952

4.3.2 Preparing household information and ACORN data

With household information data, only the 3 relevant columns for further analysis later were kept and the rest dropped. Making use of the sklearn inbuilt *make_column_transformer* function, I had passed through the *Acorn_grouped* categorical column to be transformed to one hot encoding. Once done, the columns are re-ordered for better understanding of the overall structure.

```
#Get the relevant columns only
household_information= household_information[['LCLid', 'stdorToU', 'Acorn_grouped']]

#Use make_column_transformer to make the categorical column into one hot encoding
onehot_transformer = make_column_transformer((OneHotEncoder(), ['Acorn_grouped']), remainder='passthrough', verbose_feature_names_out=False)

#Apply one hot encoding to the columns
transformed = onehot_transformer.fit_transform(household_information)

#Replace the existing household_information dataframe with the transformed one
household_information = pd.DataFrame(transformed, columns=onehot_transformer.get_feature_names_out())

#Re-order the dataframe to so that LCLid is first
household_information = household_information[['LCLid', 'stdorToU', 'Acorn_grouped_ACORN-', 'Acorn_grouped_ACORN-U',
                                               'Acorn_grouped_Adversity', 'Acorn_grouped_Affluent',
                                               'Acorn_grouped_Comfortable']]

household_information.head()
```

	LCLid	stdorToU	Acorn_grouped_ACORN-	Acorn_grouped_ACORN-U	Acorn_grouped_Adversity	Acorn_grouped_Affluent	Acorn_grouped_Comfortable
0	MAC005492	ToU	1.0	0.0	0.0	0.0	0.0
1	MAC001074	ToU	1.0	0.0	0.0	0.0	0.0
2	MAC000002	Std	0.0	0.0	0.0	1.0	0.0
3	MAC003613	Std	0.0	0.0	0.0	1.0	0.0
4	MAC003597	Std	0.0	0.0	0.0	1.0	0.0

4.3.3 Preparing daily weather data

For daily weather, a similar process was done with converting the day into a python readable format and sorting the data in order by day. Only relevant columns related to daily weather on each day were selected - there were 8 different measurements of temperature which made it redundant to keep all of them and too many features would significantly increase the time for training. As a result, only the low and high temperatures were kept from this dataset. Most of the features that were selected were numeric apart from the icon column which is renamed to weather_type and is also later converted to numerous columns through one hot encoding.

```
# convert time into date time format that python can read
daily_weather['day']=  pd.to_datetime(daily_weather['time'])
daily_weather['day']=  pd.to_datetime(daily_weather['day'],format='%Y%m%d').dt.date

#Keep only relevant columns
daily_weather = daily_weather[['day', 'icon', 'temperatureLow', 'temperatureHigh', 'dewPoint', 'cloudCover', 'windSpeed', 'pressure',
                               'visibility', 'humidity', 'uvIndex', 'moonPhase']]

#Reset index so that dataframe is in order by day
daily_weather = daily_weather.sort_values(by=['day']).reset_index(drop=True)

#Rename icon to weather type
daily_weather.rename(columns={'icon':'weather_type'}, inplace = True)

daily_weather.head(10)
```

Figure 14: Daily weather feature selection code snippet

```
#Use make_column_transformer to make the categorical column into one hot encoding
onehot_transformer = make_column_transformer((OneHotEncoder(), ['weather_type']), remainder='passthrough', verbose_feature_names_out=False)

#Apply one hot encoding to the columns
transformed = onehot_transformer.fit_transform(daily_weather)

#Replace the existing household_information dataframe with the transformed one
daily_weather = pd.DataFrame(transformed, columns=onehot_transformer.get_feature_names_out())

#Re-order for better structure and clarity
daily_weather = daily_weather[['day', 'temperatureLow', 'temperatureHigh',
       'dewPoint', 'cloudCover', 'windSpeed', 'pressure', 'visibility',
       'humidity', 'uvIndex', 'moonPhase', 'weather_type_clear-day',
       'weather_type_cloudy', 'weather_type_fog', 'weather_type_partly-cloudy-day',
       'weather_type_partly-cloudy-night', 'weather_type_wind']]
```

Figure 15: Daily weather one hot encoding code snippet

weather_type_clear-day	weather_type_cloudy	weather_type_fog	weather_type_partly-cloudy-day	weather_type_partly-cloudy-night	weather_type_wind
0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0

Figure 16: Daily weather type one hot encoded DataFrame structure

4.3.4 Preparing public holiday data

With the public holiday data upon brief inspection of the data and referencing against a trusted source on historic UK bank holidays [33], I noticed that they were imported incorrectly, and this caused some of the bank holiday dates to have the wrong day and month. When combined with the final dataset, this could lead to each of the machine learning models to have the wrong impression on how public holidays affect consumption on these days as well as result on the models to train on incorrect historical data. I manually added additional rows for missing public holidays and corrected the dates for holidays which had the wrong dates. Once it had been corrected with the right dates and bank holidays during the dataset period, this was saved into a new *uk_holidays_fixed.csv* file for future use.

```
#Dataset ranges from 2011-11-23 to 2014-02-28
#Holidays in 2011 were not included for some reason
uk_holidays.loc[uk_holidays.shape[0]] = ['2011-12-25', 'Christmas Day']
uk_holidays.loc[uk_holidays.shape[0]] = ['2011-12-26', 'Boxing Day']
uk_holidays.loc[uk_holidays.shape[0]] = ['2011-12-27', 'Christmas Day (substitute day)']


#Also some holidays were incorrectly formatted from the import

#2012
#Drop [2012-06-04 Good Friday] as it will cause problems later when correcting dates and add back in manually
uk_holidays.drop([7], inplace=True)
uk_holidays.reset_index(inplace=True, drop=True)

uk_holidays['Bank holidays'] = uk_holidays['Bank holidays'].replace({'2012-02-01': '2012-01-02'})
uk_holidays['Bank holidays'] = uk_holidays['Bank holidays'].replace({'2012-09-04': '2012-04-09'})
uk_holidays['Bank holidays'] = uk_holidays['Bank holidays'].replace({'2012-07-05': '2012-05-07'})
uk_holidays['Bank holidays'] = uk_holidays['Bank holidays'].replace({'2012-04-06': '2012-06-04'})
uk_holidays['Bank holidays'] = uk_holidays['Bank holidays'].replace({'2012-05-06': '2012-06-05'})

#Then add in good Friday
uk_holidays.loc[uk_holidays.shape[0]] = ['2012-04-06', 'Good Friday']

#2013
uk_holidays['Bank holidays'] = uk_holidays['Bank holidays'].replace({'2013-01-04': '2013-04-01'})
uk_holidays['Bank holidays'] = uk_holidays['Bank holidays'].replace({'2013-06-05': '2013-05-06'})

#Make sure that they are date readable after fixing string dates
uk_holidays['Bank holidays'] = pd.to_datetime(uk_holidays['Bank holidays'],format='%Y-%m-%d').dt.date

#Sort index
uk_holidays = uk_holidays.sort_values(by=['Bank holidays']).reset_index(drop=True)

#Save file to be used later
uk_holidays.to_csv(full_path + '/uk_holidays_fixed.csv', index=False)
```

Figure 17: UK Public holiday code correction

4.3.5 Combining them together

At this stage, the dataframes are currently structured as below. The next step is to combine them as before finalising pre-processing of the data.

Dataframe name	Description	Shape
daily_consumption	Daily individual household energy	3510433, 9
household_information	Associated ACORN grouping and tariff type for each household	5566, 5
daily_weather	Weather conditions for each day during the dataset period	882, 32

uk_holidays	Public holidays during the energy dataset period	28, 2

Table 3: Overview of the 4 files and their shapes

Combining the 4 dataframes above into one single .csv file, at this point its shape is 3517032 rows.

4.3.6 Dealing with null data

After combining the daily consumption data, daily weather and public holidays checking for any null values with *isna().any()* on the pandas dataframe revealed that some features had null values for some households in *energy_sum* as well as all of the weather features that were merged.

```
LCLid                              False
day                                False
energy_sum                          True
stdorToU                           False
Acorn_grouped_ACORN-               False
Acorn_grouped_ACORN-U              False
Acorn_grouped_Adversity            False
Acorn_grouped_Affluent             False
Acorn_grouped_Comfortable          False
temperatureLow                      True
temperatureHigh                     True
dewPoint                            True
cloudCover                          True
windSpeed                           True
pressure                            True
visibility                          True
humidity                            True
uvIndex                             True
moonPhase                           True
weather_type_clear-day              True
weather_type_cloudy                 True
weather_type_fog                    True
weather_type_partly-cloudy-day      True
weather_type_partly-cloudy-night    True
weather_type_wind                   True
public_holiday                     False
```

Figure 18: Null data in combined dataset

With the null values recorded within the *energy_sum* feature, I decided to just drop them as they would not provide any value to the dataset. Electricity consumption can be so varied between households and difficult to predict without any certainty of accuracy, and for this reason filling in consumption with averages may not be representative of the actual energy used on those days. As a result, this reduced the number of rows from 3517032 to 3517002.

```
#drop null values
daily_individual_combined_consumption = daily_individual_combined_consumption.dropna(subset=['energy_sum'])

# 3517032 -> 3517002
print(len(daily_individual_combined_consumption))
```

Figure 19: Dropping null energy_sum rows

After doing this, I looked to deal with the null weather variables. First, I checked on which dates had null values; as only the weather features had null values, it would be safe to assume that if one weather feature had a null value, then the weather features sourced from darksky would also follow. Checking against the *temperatureLow* feature, revealed that only two dates in the dataset had null data.

```
#Check which unique dates have missing values in the weather data - appears to only be ['2012-10-28' '2013-10-27']
print(daily_individual_combined_consumption[daily_individual_combined_consumption['temperatureLow'].isnull()].day.unique())
```

```
['2012-10-28' '2013-10-27']
```

Figure 20: Code snippet checking on which dates weather features were null

As a result of this, I decided to use the pandas interpolate method [34] with padding, which uses existing previous values to fill in the values which had missing data.

```
#Interpolate with existing value
daily_individual_combined_consumption = daily_individual_combined_consumption.interpolate(method = 'pad', limit_direction='forward')
```

Figure 21: Code snippet filling in null weather features through padding

Feature status before and after

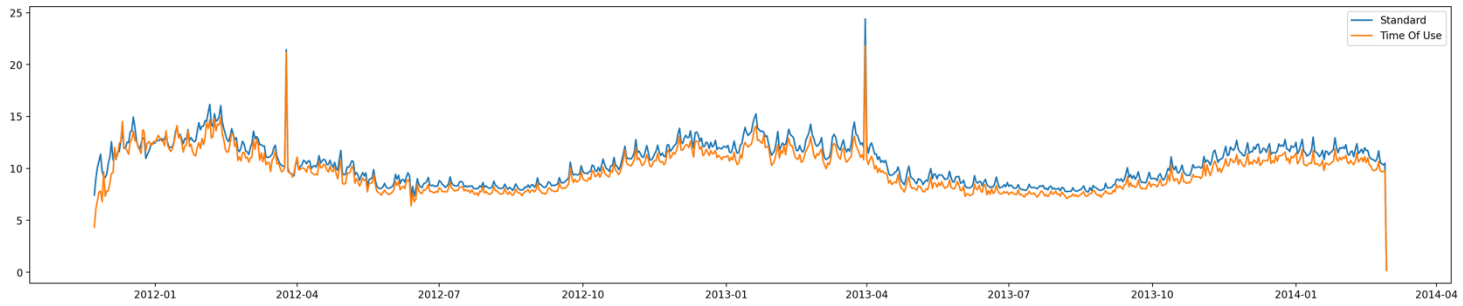
LCLid	False
day	False
energy_sum	True
stdorToU	False
Acorn_grouped_ACORN-	False
Acorn_grouped_ACORN-U	False
Acorn_grouped_Adversity	False
Acorn_grouped_Affluent	False
Acorn_grouped_Comfortable	False
temperatureLow	True
temperatureHigh	True
dewPoint	True
cloudCover	True
windSpeed	True
pressure	True
visibility	True
humidity	True
uvIndex	True
moonPhase	True
weather_type_clear-day	True
weather_type_cloudy	True
weather_type_fog	True
weather_type_partly-cloudy-day	True
weather_type_partly-cloudy-night	True
weather_type_wind	True
public_holiday	False

LCLid	False
day	False
energy_sum	False
stdorToU	False
Acorn_grouped_ACORN-	False
Acorn_grouped_ACORN-U	False
Acorn_grouped_Adversity	False
Acorn_grouped_Affluent	False
Acorn_grouped_Comfortable	False
temperatureLow	False
temperatureHigh	False
dewPoint	False
cloudCover	False
windSpeed	False
pressure	False
visibility	False
humidity	False
uvIndex	False
moonPhase	False
weather_type_clear-day	False
weather_type_cloudy	False
weather_type_fog	False
weather_type_partly-cloudy-day	False
weather_type_partly-cloudy-night	False
weather_type_wind	False
public_holiday	False

4.3.7 Tariff and date period specification

At this point in pre-processing, the dataset is still quite large in terms of rows, which would likely cause issues with lengthy training times. To cut it down further, I decided that I would only be looking at households on a standard tariff rather than a Time of Use (ToU) tariff. According to UK Power Networks the 1100 customers who were on these tariffs were given prices of electricity a day ahead during specific times which were either: High (67.20p/kWh), Low (3.99p/kWh) or normal (11.76p/kWh) [35]. Similar to how Agile Octopus (a tariff that resembles Time of Use tariffs today [36]) works, the main intention of this is to encourage consumers to be mindful of their usage, especially during peak times to be able to reap on the financial benefits of doing this.

It can be seen below how consumption varied between the two groups. Initially, there was quite a large variation between the two groups, but as time went on, it could be seen that households ToU tariffs consumed less electricity.



In addition to only using standard tariff households, from looking at the number of unique households in the consumption dataset, the most optimal period with the most households appeared to be between 2012 and 2013. The dataset is then filtered to only contain data between 2012 and 2013, lowering the dataset size from 3517002 to 3210380 rows.

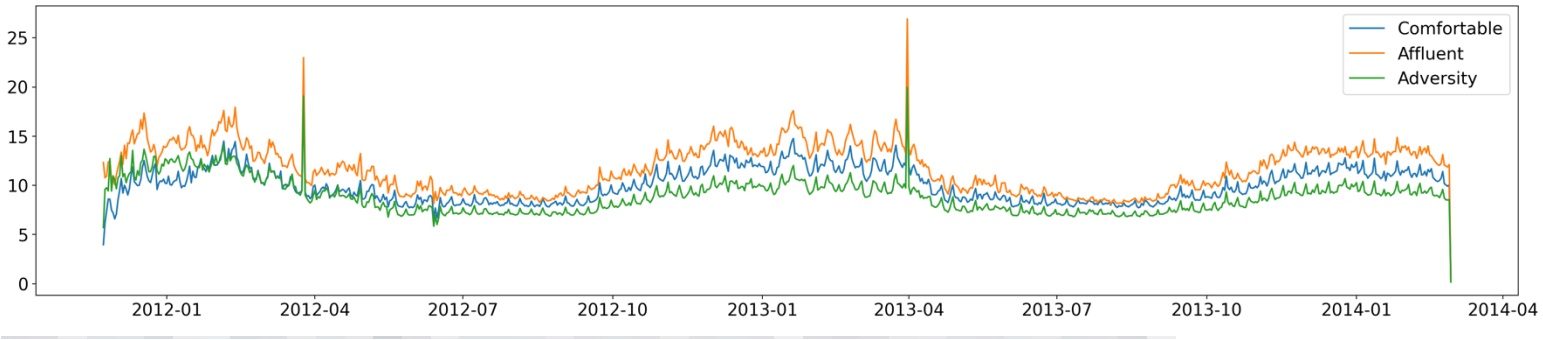
4.3.8 Acorn grouping

As mentioned previously in the overview of the *information_households.csv* file, the Acorn groups were of interest due to the potential improvements they could have on the performance of models if their separate groups were investigated. First, it was important to completely remove households which were marked as either Acorn grouping *ACORN-* or *ACORN-U* as these two groups were not relevant to what could be useful in the final dataset and cannot be estimated as there was not enough information to properly reclassify those households. Once completed, this had further reduced the size of the dataset from 3210380 rows with 29 features to 2544154 rows with 26 features.

```
#Get only households on standard tariffs
daily_individual_combined_consumption = daily_individual_combined_consumption[daily_individual_combined_consumption.stdorToU == 'Std']

#Then filter irrelevant rows with the below ACORN groups
daily_individual_combined_consumption = filter_rows(daily_individual_combined_consumption, "Acorn_grouped_ACORN-", [1.0])
daily_individual_combined_consumption = filter_rows(daily_individual_combined_consumption, "Acorn_grouped_ACORN-U", [1.0])

#Then finally drop the columns themselves
daily_individual_combined_consumption.drop(['Acorn_grouped_ACORN-', 'Acorn_grouped_ACORN-U', 'stdorToU'], axis=1, inplace=True)
```

Once completed, all households within each Acorn group were grouped together to be able to provide an insight to the average per household energy consumption usage within these groups. It can be seen from the graph below the differences in consumption between the Acorn groups: Affluent households on average were the group with the highest electricity consumption whereas those in the Adversity grouping had the lowest consumption.

With this small subtle distinction between the three groups, it will be interesting to see how each of the models perform being trained on a dataset with all the acorn types combined in one dataset, compared being trained specifically on each of the 3 individual acorn types.



4.3.9 Specific household selection for testing

In addition to the standard model performance metrics when making predictions on the X_test dataset when passed into each of the developed models, it would be important to check how well the models to be developed do with generalising to any type of household with unseen data.

Making use of a function I created named *get_households()* which returns households from specific Acorn groups, I looked to select 3 different households from each of the 3 Acorn groups for the models to be developed to make predictions on. When getting households within each of the groups, I made the decision to select 3 different Acorn classification types within a specific Acorn group. An example is seen below were in the Affluent Acorn grouping, there were 3 different classifications seen, Acorn-E, Acorn-C and Acorn-D.

```
acorn_affluent = get_households("Acorn_grouped_Affluent")

acorn_affluent = acorn_affluent.merge(information_households, on='LCLid', how='left')
acorn_affluent.head(50)
```

	LCLid	day	stdorToU	Acorn	Acorn_grouped	file
0	MAC004537	733	Std	ACORN-E	Affluent	block_21
1	MAC004518	733	Std	ACORN-E	Affluent	block_20
2	MAC004512	733	Std	ACORN-E	Affluent	block_20
3	MAC004511	733	Std	ACORN-E	Affluent	block_20
4	MAC004510	733	Std	ACORN-E	Affluent	block_20
5	MAC004504	733	Std	ACORN-C	Affluent	block_5
6	MAC004500	733	Std	ACORN-E	Affluent	block_20
7	MAC004496	733	Std	ACORN-E	Affluent	block_20
8	MAC004487	733	Std	ACORN-E	Affluent	block_20
9	MAC004482	733	Std	ACORN-E	Affluent	block_20
10	MAC004477	733	Std	ACORN-E	Affluent	block_20
11	MAC004476	733	Std	ACORN-E	Affluent	block_20
12	MAC004475	733	Std	ACORN-D	Affluent	block_8

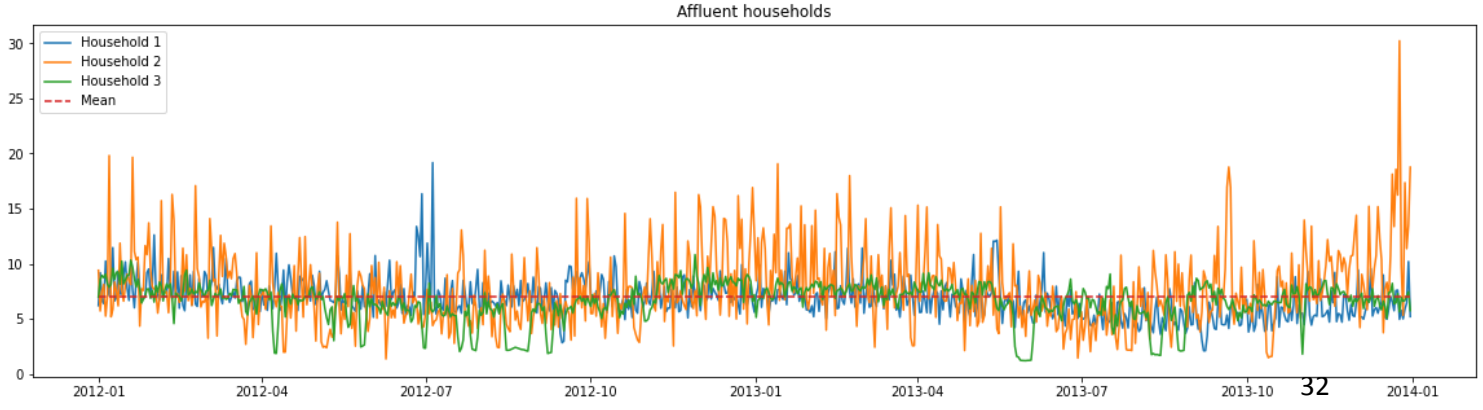
Figure 26: Selection of households within the Acorn Affluent group

The final selection of households from each group used for testing are as below. They are removed from the final dataset to avoid bias from happening.

Affluent households		**Comfortable households**		**Adversity households**	
LCLid	**Acorn Classification**	**LCLid**	**Acorn Classification**	**LCLid**	**Acorn Classification**
MAC004848	Acorn-C	MAC000244	Acorn-F	MAC004508	Acorn-K
MAC004475	Acorn-D	MAC004850	Acorn-G	MAC004507	Acorn-L
MAC004510	Acorn-E	MAC000239	Acorn-H	MAC004513	Acorn-Q

Table 4: Overview of selected households from each Acorn group to be passed as unseen data to developed models

In order of Acorn classification (C-Q), the selected household consumptions are represented in each of the Acorn groups graphically below.



As seen above, within each Acorn grouping they follow almost similar consumption usage patterns, which can especially be seen within the adversity households selected for testing. The average electricity seen in the above selected households follow the same pattern as observed previously when all household consumption was averaged by Acorn grouping.

Affluent households	Comfortable households	Adversity households
7.032278308594816 kWh	6.428361073306048 kWh	5.861303774715778 kWh

Table 5: Average electricity in all 3 households selected per Acorn group

Overall, after pre-processing the dataset, the final dataset size is 2783514 rows, 23 features with a total of 4390 unique households included in the final dataset of consumption data.

4.3.10 Manually splitting dataset between training and testing dataset

To ensure that bias did not happen when training the models, rather than using the train_test_split function in sklearn, I manually split households to be included either in the training dataset or test dataset. After getting some clarification on the approach that should be taken with time series data [37], the training data would be composed of household consumption in the full year of 2012, whereas the testing data would be the full year of 2013.

To also make sure that there was equal representation of all the acorn groups, I used the *get_households* function which would take in the name of the Acorn group and return households that were part of that group only. Then, using the .sample() function from the pandas library it randomly selected LCLids to be part of the training dataset, whereas the testing dataset would contain the rest of the LCLids that were not selected. Most importantly for an equal representation of each of the Acorn groups in the training and testing dataset in each group, 80% of LCLids in each group were allocated for training and 20% for testing.

```
acorn_affluent = get_households("Acorn_grouped_Affluent")
acorn_comfortable = get_households("Acorn_grouped_Comfortable")
acorn_adversity = get_households("Acorn_grouped_Adversity")

#select random households to be part of training dataset
acorn_affluent_training = acorn_affluent.sample(frac=0.8, random_state=25)
acorn_comfortable_training = acorn_comfortable.sample(frac=0.8, random_state=25)
acorn_adversity_training = acorn_adversity.sample(frac=0.8, random_state=25)

#then drop LCLids in not included in training dataset and keep the rest for testing
acorn_affluent_testing = acorn_affluent.drop(acorn_affluent_training.index)
acorn_comfortable_testing = acorn_comfortable.drop(acorn_comfortable_training.index)
acorn_adversity_testing = acorn_adversity.drop(acorn_adversity_training.index)
```

Figure 30: Code snippet of training and testing selection

Below were the number of training and testing samples in each of the Acorn groups totalling to 4380. The total unique households reduced by 10 from 4390 because of the filter of the date period that training, and testing would be set for between 2012 and 2013 only.

Acorn Group	Training samples	Testing samples
Affluent	1357	337
Comfortable	942	235
Adversity	1209	300

Table 6: Number of households from each group in training and testing samples

After LCLids were selected in each of the Acorn groups, they were saved into their respective Acorn training and testing files. Finally, they were then combined all together into a separate training and testing file. In total 8 separate .csv files were saved for later use.

File name	File name
daily_individual_consumption_final_training.csv	daily_comfortable_training.csv
daily_individual_consumption_final_testing.csv	daily_comfortable_training.csv
daily_affluent_training.csv	daily_adversity_training.csv
daily_affluent_testing.csv	daily_adversity_training.csv

Table 7: Overview of each of the files that will be used as part of training and testing of models developed

4.4 Correlation analysis and feature selection

4.4.1 Creating a grouped dataset to view average correlations

To be able to see general relationships between each of the features in the dataset, I created a separate grouped dataset based on the individual household daily consumption dataset created in the previous section. The purpose of this was to provide an insight into the average home in London by having consumption on a per household basis. It involved creating a new *house_count* column to total the number of houses which recorded readings on a specific day as well as a new *total_energy* column to sum all the consumption readings on a certain day into that column. Finally individual household features such as *LCLid* and *energy_sum* are removed.

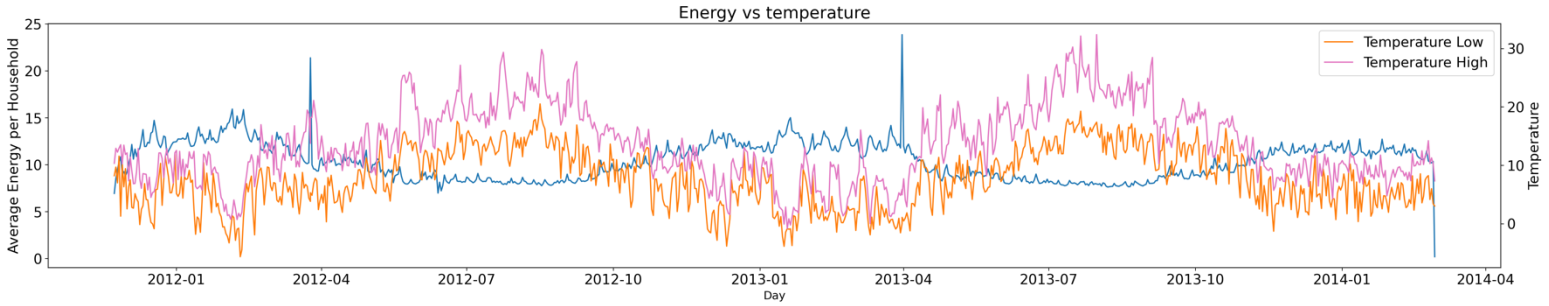
```
#Load in individual dataframe
daily_grouped_consumption = pd.read_csv(full_path + '/daily/daily_individual_combined_consumption_no_nulls.csv')

#Make date readable format
daily_grouped_consumption['day']=  pd.to_datetime(daily_grouped_consumption['day'],format='%Y-%m-%d').dt.date
```

```
#Creating a new column for total number of houses that recorded on a certain day
house_count = daily_grouped_consumption.groupby('day')[['LCLid']].nunique()

#Then make another column called total_energy to get the total electricity consumption of all households on a certain day
total_energy = daily_grouped_consumption.groupby('day')[['energy_sum']].sum()
total_energy = total_energy.merge(house_count, on = ['day']).reset_index()
total_energy.rename(columns={'LCLid':'house_count'}, inplace=True)

#Then get the average elctricity consumed by all households that recorded on each day
total_energy['total_energy_avg'] = total_energy['energy_sum']/total_energy['house_count']

#Then remove the irrelevant columns that won't be needed anymore
daily_grouped_consumption = daily_grouped_consumption.drop(['LCLid', 'energy_sum'], axis=1)

#This results in duplicate values of weather features. Drop them so that there is only one entry per day
daily_grouped_consumption.drop_duplicates(inplace=True)

#Merge the existing dataset with the new one by day
total_energy = total_energy.merge(daily_grouped_consumption, on = ['day'])

#Drop energy_sum as its no longer needed
total_energy.drop(['energy_sum'], axis=1, inplace=True)
```

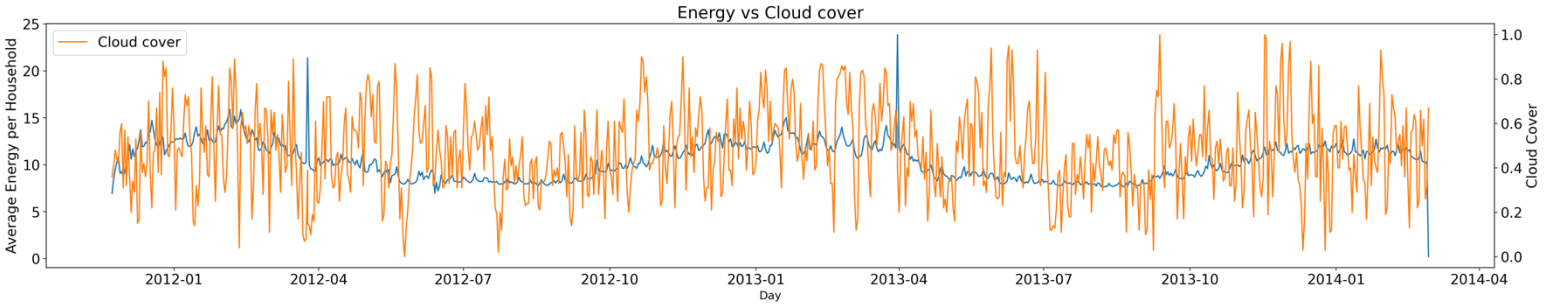
	day	house_count	total_energy_avg	temperatureLow	temperatureHigh	temperatureMin	temperatureMax	windBearing	dewPoint	cloudCover	windSpeed
0	2011-11-23	13	6.952692	8.24	10.36	3.81	10.36	229.0	6.29	0.36	2.04
1	2011-11-24	25	8.536480	9.71	12.93	8.56	12.93	204.0	8.56	0.41	4.04
2	2011-11-25	32	9.499781	7.01	12.27	7.46	13.03	243.0	7.24	0.48	5.02
3	2011-11-26	41	10.267707	11.59	12.96	7.01	12.96	237.0	6.96	0.44	5.75
4	2011-11-27	41	10.850805	1.31	13.54	4.47	13.54	256.0	5.76	0.42	5.48

4.4.2 Graphical representation of electricity consumption against weather features

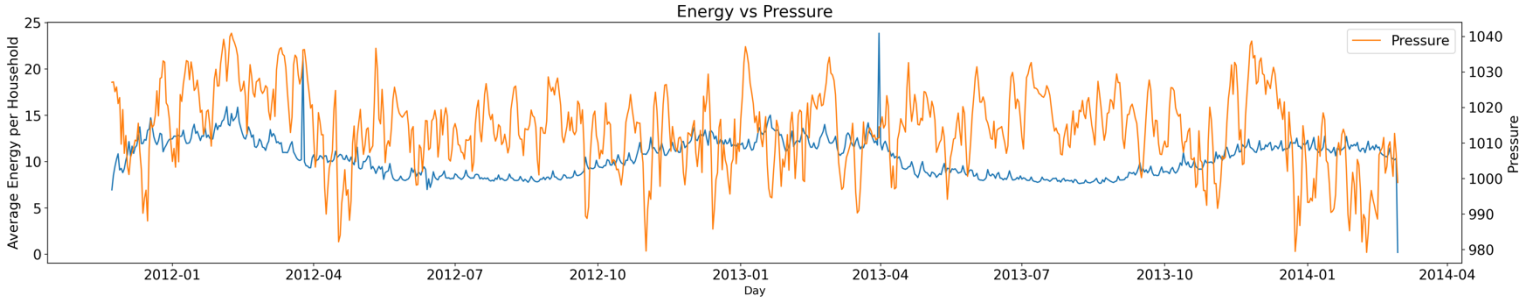
Consumption against low and high temperatures



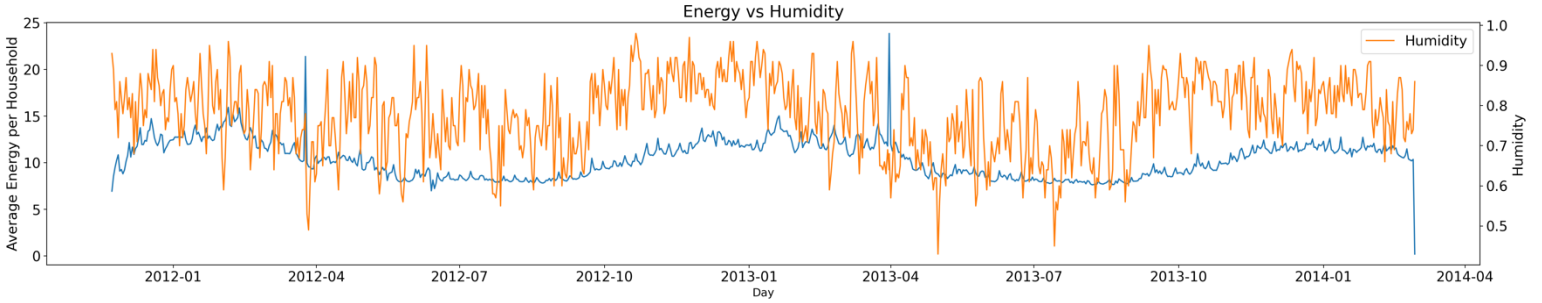
Consumption against cloud cover



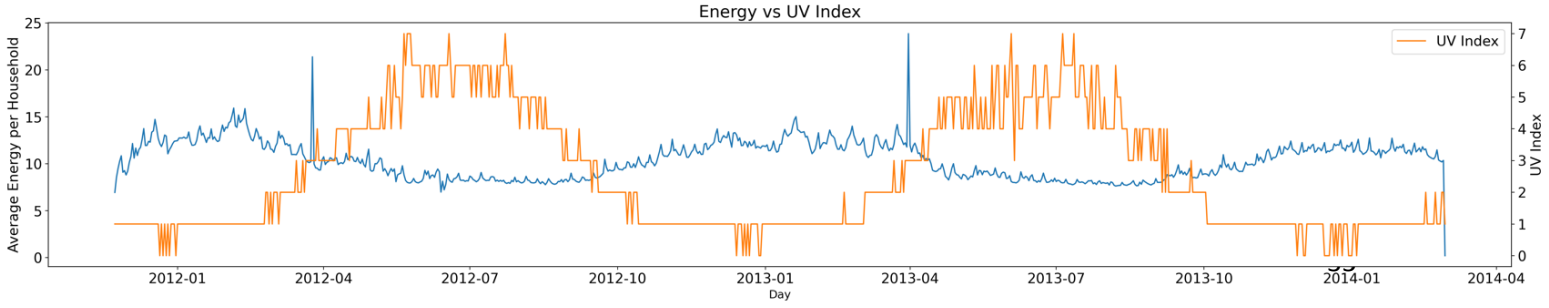
Consumption against pressure



Consumption against humidity



Consumption against UV Index



4.4.3 Observations from electricity consumption against weather features

From the graphical representations above on an average electricity consumed per household level there were some insights gained from this which provides expectations on which features would be impactful when it comes to feature selection.

There is a slight negative correlation between the average energy consumed at both high and low temperatures, where during winter periods there is higher energy consumption compared to summer months. An explanation for this would be through the increased use of electric heating as an example as well as people staying at home more generally.

With cloud cover, there isn't much of a distinct pattern and there is quite high variation seen between each day within the dataset period, so there is little to no correlation seen here.

With pressure a very weak positive correlation can be seen here: in periods of higher pressure, generally there is lower energy consumption and vice versa.

With humidity, there is also a weak positive correlation, as it increases, energy consumption appeared to also increase during this period.

Finally with UV index, this is a very clear strong negative correlation that also appears to also correspond to seasonal weather also. The higher the UV index, the lower energy consumption on average is.

4.4.4 Feature selection using Pearson's correlation method

Pearson's correlation coefficient helps to find the relationship between two features by providing a value between -1 and +1. A value between 0 to 1 indicates that there is a positive correlation whereas one feature increases in value the other also goes in the same direction. On the other hand, a value between 0 to -1 would indicate a negative correlation, whereas one feature decreases, the other increases or vice versa. Pearson's correlation coefficient can be calculated using the formula below [38]

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Where,

r = Pearson Correlation Coefficient

x_i = x variable samples

y_i = y variable sample

$\bar{x}$ = mean of values in x variable

$\bar{y}$ =mean of values in y variable

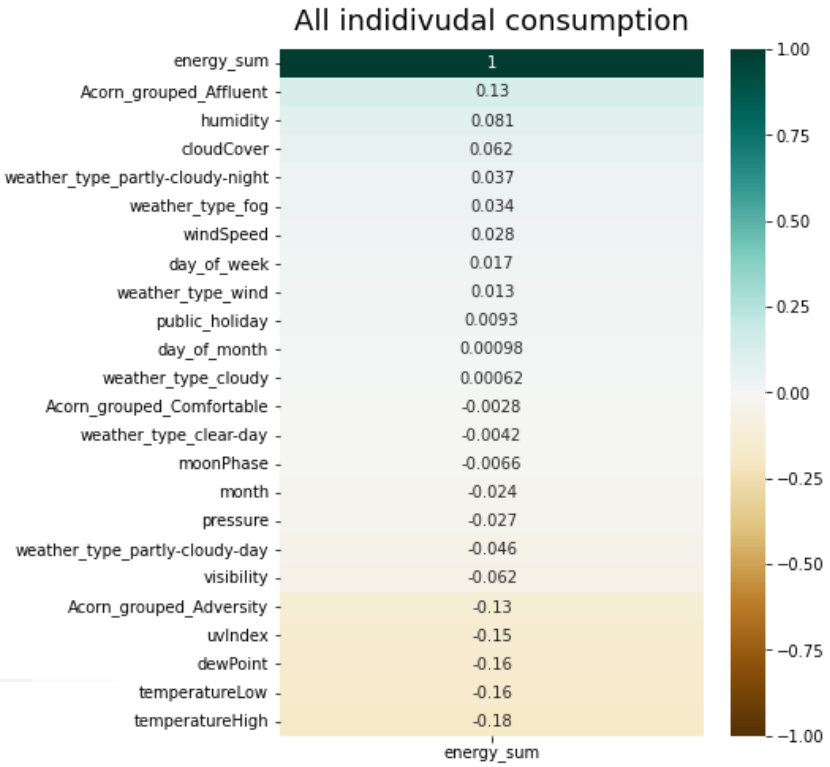
Figure 34: Pearson's correlation method formula [38]

There are 4 different variants made of the consumption dataset to be able to provide comparison as to which model has the best performance for generalisation in electricity forecasting. They are as follows:

1. All ACORN groups together with individual household consumption
2. Affluent Acorn group consumption only with individual household consumption
3. Comfortable Acorn group consumption only with individual household consumption
4. Adversity Acorn group consumption only with individual household consumption

Currently, there are 26 features in the dataset with 797577 individual rows at most for training as well as 303895 rows for testing. This is quite large and should be reduced so that model performance is not unnecessarily long and that different combinations of features can also be used when finding ways to improve model performance.

Applying the pandas .corr() function on the energy_sum provided the following outcome on the right with feature correlation.



With all acorn groups combined in one dataset, a household being part of the affluent acorn group had the highest weak positive relationship to energy consumed, whereas in the acorn specific datasets, humidity was the highest feature with a weak positive correlation and temperature being one with a weak negative correlation. As a result of this the 8 following features were selected based on positive and negative correlation:

- All acorn combined dataset: *Acorn_grouped_Affluent, humidity, cloudCover, weather_type_partly-cloudy-night, Acorn_grouped_Adversity, uvIndex, dewPoint* and *temperatureHigh*
- Acorn specific datasets: *humidity, cloudCover, weather_type_partly-cloudy-night, weather_type_fog, visibility, uvIndex, dewpoint* and *temperatureHigh*

Overall, from the values seen from Pearson's correlation above, it does indicate that the dataset is quite noisy so there may be challenges with generalisation for forecasting.

4.4.5 Feature selection using Mutual Information

Mutual information also known as information gain is another feature selection technique that can be used to measure how much information can be found for one feature by observing another separate feature. The main advantage to using this is that it can capture all types of relationship and is not based on linear relationships. Mutual information is always larger or greater than zero. [39] Features which are closer to 0 indicate that they have no relation to the target feature and are redundant to keep in in the dataset, whereas the larger the value from mutual information that features have, the stronger the relationship it is to the target feature. [40]

Using the mutual information technique, the top 8 features were selected by it in each of the datasets as seen below along with their corresponding scores from highest to lowest.

All acorn combined dataset		**Affluent dataset**		**Comfortable dataset**		**Adversity dataset**	
Feature name	Scores	Feature name	Scores	Feature name	Scores	Feature name	Scores
dewPoint	0.026561	month	0.028040	month	0.035340	month	0.030166
temperatureHigh	0.026387	dewPoint	0.025104	temperatureHigh	0.034368	temperatureHigh	0.027577
month	0.025686	temperatureLow	0.025092	temperatureLow	0.030599	dewPoint	0.024777
temperatureLow	0.025472	temperatureHigh	0.025074	dewPoint	0.030046	temperatureLow	0.023567
pressure	0.023209	uvIndex	0.021918	uvIndex	0.028778	pressure	0.020758
windSpeed	0.019747	pressure	0.018425	pressure	0.024097	uvIndex	0.019946
uvIndex	0.018390	windSpeed	0.016760	windSpeed	0.022941	windSpeed	0.018771
visibility	0.015709	visibility	0.014666	humidity	0.018933	visibility	0.013928
Acorn_grouped_Affluent	0.013212	humidity	0.012375	visibility	0.018581	humidity	0.012149

Table 8: Mutual information correlation scores for energy_sum

From the above results, it can be concluded that a similar pattern was seen across the datasets where features were about the same in terms of impact. Most notably a household being affluent or not had been the acorn group the most influential towards forecasting the electricity consumed in the acorn combined dataset. Although there are 9 features shown above, it is important to note that temperatureLow will be removed in the feature selection as it is redundant to keep

Chapter 5: Implementation of machine learning models

Within this chapter, the approach taken for implementing the selected models will be discussed as well as the process of getting them to run and be able to generate electricity consumption forecasts.

5.1 Python libraries required

5.1.1 Pandas

Pandas is a popular python package which is commonly for analysis and manipulation of data. It is primarily used for the data structure that it provides, also known as Series and DataFrame structures. It also provides the ability to handle missing data, modification of columns and rows, merging and joining of datasets, correlation analysis and more. It is essential for any data science project. [41]

5.1.2 NumPy

NumPy is another popular library used for data projects in Python. It is known for its mathematical functions and how quick it is to be able to perform computation. One of the most common operations is shape manipulation which is needed for data input in TensorFlow models. [42]

5.1.3 Matplotlib

Matplotlib is a library specifically for visualisation of data. It allows for a wide variety of graphical representations such as line graphs, bar charts and scatter graphs. It is used in throughout this dissertation to provide outputs after performing certain actions such as correlation analysis or comparing between actual values and forecasted ones. [43]

5.1.4 Sklearn

Sklearn is a library which provides tools for predictive data analysis and machine learning in Python. It is built on NumPy, SciPy and Matplotlib. Some of the tools that are included within the library are classification, regression, and clustering. Machine leaning models include linear regression, random forest, and support vector machine [44]. Within this project, Multiple Linear Regression, kNN and SVR are implemented using the sklearn library, specifically with the use of: *LinearRegression, KNeighborsRegressor* and *LinearSVR.*

5.1.5 TensorFlow

TensorFlow is an open source library which was developed by Google, which is designed for the use of creating deep learning models and machine learning. It is most used for building neural networks [5]. Within this project, it is used to develop and build the LSTM model, specifically with the use of *Sequential()* and *model.add(LSTM)*.

5.2 Feature combinations

To test for any difference performance of the machine learning models that will be developed, in addition to the 8 features that were selected through Pearson's correlation and Mutual Information in section 4, calendar attributes will be added: with one feature set including public holidays, one including the month number and day of the week and one including a combination of both. For each of the algorithms, they will be looped with performance metrics and predictions saved in a separate array so that they can be compared when evaluating the results. (Please refer to the appendix which contains the full code snippet)

```
#For use when iterating through each of the feature selection groups
#For all acorn groups combined
individual_feature_groups_pearson = [pearson_0, pearson_1, pearson_2, pearson_3]
individual_feature_groups_mutual_information = [mutual_information_0, mutual_information_1, mutual_information_2, mutual_information_3]

#For all specific acorn groups
acorn_groups_pearson = [pearson_4, pearson_5, pearson_6, pearson_7]
acorn_groups_mutual_information = [mutual_information_4, mutual_information_5, mutual_information_6, mutual_information_7]
```

5.3 Helper functions

For each of the machine learning models developed, I have created several functions which are used to provide consistency in model development for each of the algorithms as well as the output that is provided from running the models for ease of comparison. Below are a description of what they do.

5.3.1 run_model()

This function takes in as parameters: the training and test dataset along with a feature group and acorn type household arrays which will be iterated on so that performance metrics can be compared and evaluated on later. On each iteration, the performance metrics get printed out in a tabular format for ease of comparison and on completion, the household predictions that the model made on that iteration with the specific feature group selected at the time are returned by the function. (Please refer to the appendix which contains the full code snippet)

5.3.2 get_performance()

This function calculates the performance metrics of how well a model has performed. It takes in the household consumption data and the predicted electricity consumption in 2D array format by the model. As the models are expected to have been trained with multiple feature groups, a loop is done until the end of the 2D array. This function prints each of the performance metrics for each feature group.

```
#Get the performance metrics
#MAE, MAPE, MSE, RMSE, R2
def get_performance(household_consumption, household_predictions):
  household_consumption = household_consumption['energy_sum']

  print("-- Household forecasting performance per feature group: --\n")
  print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format('Feature group', 'MAE', 'MAPE', 'MSE', 'RMSE', 'R2'))

  for i in range(0, len(household_predictions)):
    mae = mean_absolute_error(household_consumption, household_predictions[i])
    mape = mean_absolute_percentage_error(household_consumption, household_predictions[i])
    mse = mean_squared_error(household_consumption, household_predictions[i])
    rmse = np.sqrt(mse)
    r2 = r2_score(household_consumption, household_predictions[i])
    print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format(i, mae, mape, mse, rmse, r2))
```

Figure 38: get_performance function code snippet

5.3.3 visualise_results()

Making use of the matplotlib library, the function takes in the actual household electricity data as well as the predictions that the model has made in 2D array format. There is a loop within the function which allows matplotlib to be able to add multiple lines to represent each feature group that the model trained on as well as the results.

```
#Function to visualise results quickly
def visualise_results (household_data, household_predictions, title):
  household_data_consumption = household_data['energy_sum']
  household_data['day']=  pd.to_datetime(household_data['day'],format='%Y-%m-%d')

  #Clear any existing plots
  plt.clf()
  plt.figure(figsize = (25,5))
  plt.plot(household_data['day'], household_data_consumption, label='Actual consumption')

  #Loop through each of the household predictions in the 2D array for each feature group, and plot
  for i in range(0, len(household_predictions)):
    plt.plot(household_data['day'], household_predictions[i], label='Feature group: ' +str(i))

  plt.title(title)
  plt.xlabel('Day')
  plt.ylabel('Electricity consumed (kWh)')
  plt.legend()
  plt.show()
```

Figure 39: visualise_results function code snippet

5.4 Additional functions for kNN and LSTM

5.4.1 Function for kNN

5.4.1.1 optimise_k()

As kNN works by predicting new values based on the mean of the nearest K values set, this function would be used to provide the various performance metrics when the model makes predictions with the test dataset provided with a range of values with K. (Please refer to the appendix which contains the full code snippet)

5.4.2 Functions for LSTM

5.4.2.1 reshape_predicitions()

When making predictions, the LSTM model developed outputs a vector output, and this function converts the output into a numpy 2D array to cater for the each of the predictions made in each household group.

```
def reshape_predictions(household_predictions):
  #Reshape each of the household predictions before returning
  household_predictions = np.array(household_predictions)
  household_predictions_reshaped = household_predictions.reshape(household_predictions.shape[0],
                                                                 household_predictions.shape[1])
  return household_predictions_reshaped
```

Figure 40: reshape_predictions() function code snippet

5.4.2.2 create_sliding_window()

With the nature of LSTM remembering previous historic information in the model, lags of 7 days of previous consumption data are provided as additional input into the model. To be able to achieve this, this function was created. It takes in the input dataset (either the training or test dataset), the number of previous days to include, the number of future days to include as well as the number of features that will be included for prior historic data. Making use of the pandas shift() function, it provides the ability to get a certain number of previous days, and this gets saved to a new array *consumption_data* to hold the values.

Due to time constraints within this dissertation, only historic energy consumption will be used for LSTM, though this function does leave the possibility for future work to experiment the model performance changes on other features.

```
#Takes in the data consumption, number of days to use before, number of days ahead to include,
#How many features to perform this on
def create_sliding_window(input_data, historic_days=1, future_days=1, n_features=1):
  #Just perform sliding scale on consumption data not other features
  consumption_data = input_data['energy_sum']
  energy_values, column_names = list(), list()

  #Go through each of the previous energy consumed and save it to the energy_values list
  for i in range(historic_days, 0, -1):
    energy_values.append(consumption_data.shift(i))
    column_names += [('energy_sum%d(t-%d)' % (j+1, i)) for j in range(n_features)]

  #The future energy consumed
  for i in range(0, future_days):
    energy_values.append(consumption_data.shift(-i))
    if i == 0:
      column_names += [('energy_sum%d(t)' % (j+1)) for j in range(n_features)]
    else:
      column_names += [('energy_sum%d(t+%d)' % (j+1, i)) for j in range(n_features)]

  #Put it all together and combine into final output
  combined_output = pd.concat(energy_values, axis=1)
  combined_output.columns = column_names
  combined_output.dropna(inplace=True)

  #Final output is the historic energy consumption combined with the original dataset
  final_output = combined_output.merge(input_data, left_index=True, right_index=True)
  final_output.reset_index(inplace=True)
  final_output.drop(['index'], axis=1, inplace=True)
  return final_output
```

Figure 41: create_sliding_window code snippet

5.5 General machine learning model flow

Within each of the machine learning models, the following flow of data was as follows below.

1. The specific dataset to train and test on each model is loaded
2. The run_model() function is then called with the datasets to train and test on, as well as the household groups that will be tested and the feature group combinations to use
3. Within this function, the data passed in is scaled using MinMaxScaler() for both training and test data, except the target feature
4. The data is then fit to the model
5. Predictions are made with the test dataset
6. Model performance metrics are then outputted
7. The trained model will then make predictions on the household groups passed into the function and return their predictions
8. This is then repeated until all feature group combinations have been used

5.6 LSTM network structure

With Multiple Linear Regression, Support Vector Regression, and k-Nearest Neighbors, they all had a similar structure and did not require much designing of the structure for the algorithms, sklearn provided it internally. However, with LSTM and TensorFlow, the neural network had to be designed with neurons as well as the input data being reshapes to fit the requirements of the LSTM model. Below is the structure of the LSTM network that was used to train on the datasets and make predictions on households for electricity forecasting.

```
n_timesteps, n_features, n_outputs = X_train_scaled.shape[1], X_train_scaled.shape[2], y_train.shape[0]
model.add(LSTM(200, activation='relu', input_shape=(n_timesteps, n_features)))
model.add(RepeatVector(1))
model.add(LSTM(200, activation='relu', return_sequences=True))
model.add(TimeDistributed(Dense(100, activation='relu')))
model.add(TimeDistributed(Dense(1)))
model.compile(loss='mse', optimizer='adam')
```

Hidden layer 1: Contains 200 units which reads the input data and outputs 200 vectors once data passes through.

Hidden layer 2: RepeatVector takes the output of the first LSTM layer and repeats it another time.

Hidden layer 3: Another fully connected LSTM layer is then used to interpret the results from the previous layer and predicts one single step forward i.e., the electricity predicted.

Hidden layer 4 & 5: The TimeDistributed wrapper is used to apply the Dense layer to every sample that it receives. The final layer outputs the final electricity consumed prediction.

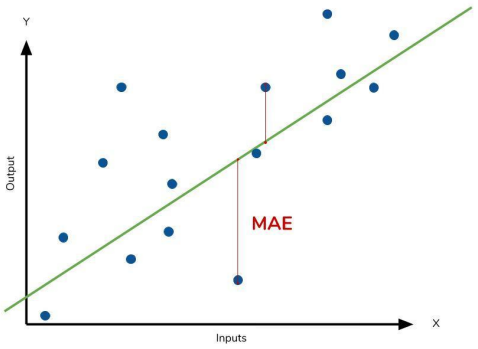
Chapter 6: Model performance and evaluation

This chapter provides the performance metrics used within the dissertation as well as the results gained from each of the model predictions against test data.

6.1 Performance metrics used

6.1.1 Mean Absolute Error

MAE is the difference between every data point predicted, taking the absolute value of each one so to make sure that positive and negatives do not cancel out. The end value is the average of all the differences together. The lower the value, the better a model is at predicting values and is stable, whereas a higher MAE value indicates that a model is not so great and will not always be reliable. Below is the graphical representation of MAE as well as the equation.



$$mae = \frac{\sum_{i=1}^{n} abs\,(y_i - \lambda(x_i))}{n}$$

6.1.2 Mean Absolute Percent Error

MAPE is the same as MAE, however the difference is that it measures the size of an error in percentage terms. Like before, a lower MAPE indicates a better performing model. However, a disadvantage to this is that MAPE can be biased towards predictions which are lower than the actual expected value, resulting in a lower MAPE value which may provide a misrepresentation of how a model is truly performing.

$$MAPE = \frac{100\%}{n} \Sigma \left| \frac{\overbrace{y - \hat{y}}^{\text{The residual}}}{y} \right|$$

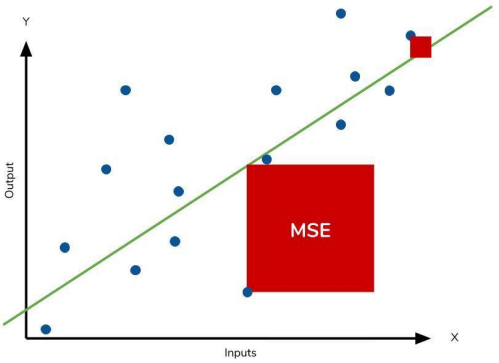
Multiplying by 100% converts to percentage

Each residual is scaled against the actual value

Figure 44: Mean Absolute Percentage Error formula [47]

6.1.3 Mean Squared Error

MSE is the average difference between actual values against predicted values. It is the same principal as MAE where the difference between every data point predicted is averaged, however this is not in absolute terms.



$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

Figure SEQ Figure * ARABIC 46: Mean Squared Error formula [47]

6.1.4 Root Mean Squared Error

RMSE is another type of commonly used measure for measuring how well predictions a model makes are. It shows how far predictions are from actual expected values using Euclidean distance. The lower the RMSE, the better a model can make predictions and fit to a dataset. However, one issue with using RMSE as a performance metric on its own is that it can be quite easily influenced by some predictions which are worse than others, creating a false representation of how a model is performing.

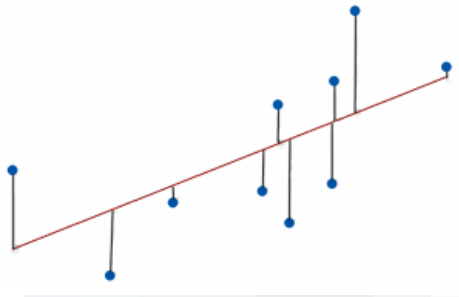
$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Figure 47: RMSE formula [48]

6.1.5 R Squared

R squared is the final statistical measure in this dissertation. It is used to measure how close predicted values are to the fitted regression line. It is also known as the coefficient of determination. If r squared is 1, it means that the model has performed perfectly, however if it is 0, it means that the model will likely perform badly on an unseen dataset. Its formula can be calculated as below.

$$R^2 = \frac{\text{Variance explained by the model}}{\text{Total variance}}$$



6.2 Discussion of model performance and results

Within this section, the models developed are trained using household consumption data during 2012 and tested with data during 2013 with 4 different datasets: a combination of all acorn groups together and each of the 3 acorn groups separated (affluent, comfortable, adversity). The results shown are the best feature group in that dataset as well as performance metrics achieved by the model when making predictions on the 2013 test datasets.

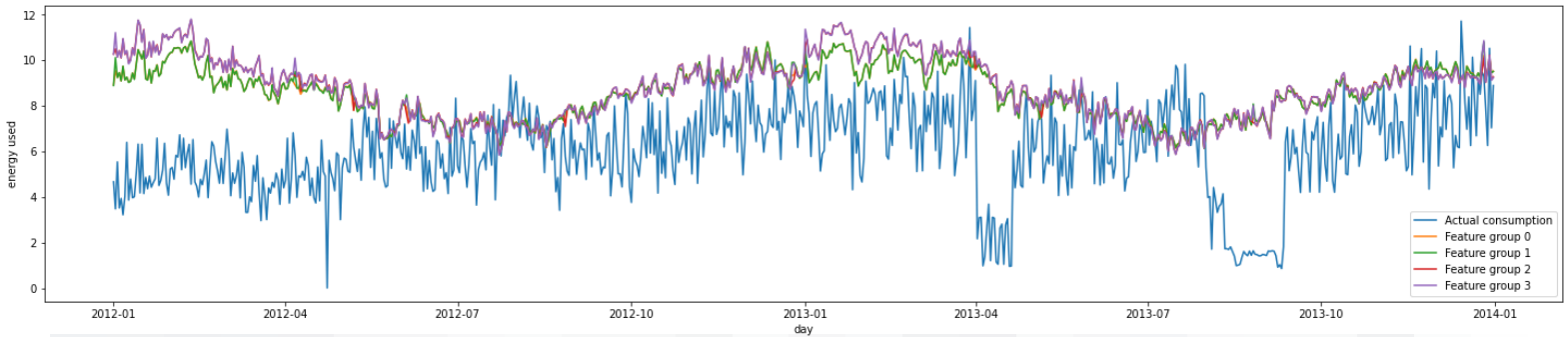
6.2.1 Multiple Linear Regression

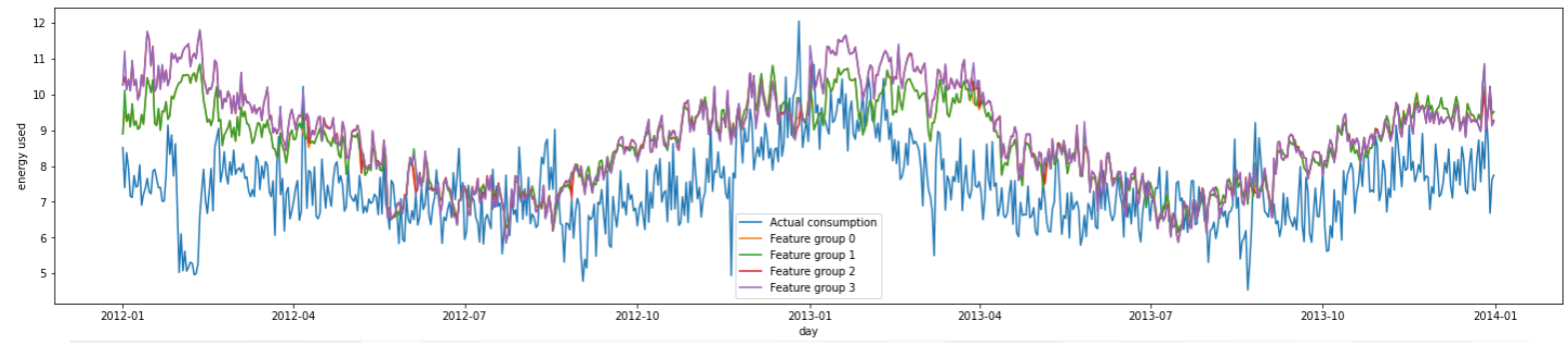
Dataset:	**All acorn groups**	**Affluent only**	**Comfortable only**	**Adversit**

Best feature group	0 – Pearson	0 – Pearson	2 – Pearson	0 - Pearson
MAE	5.815650291300232	7.37928060717557	5.0789187125006405	4.617104899
MAPE	155789447501784.5	171176309698622.34	94196670293075.88	1826669560
MSE	81.15178939742358	132.8713676295043	58.15865717486562	41.0704609
RMSE	9.008428797377686	11.526984325030737	7.62618234602777	6.40862395
R2	0.05610916824813339	0.04265200018316317	0.03971616140063205	0.02956649

Table 9: Multiple Linear Regression model results

From the results above, the best performing model for forecasting electricity consumption was on adversity dataset, achieving a MAE value of 4.62. The model also performed quite well on the comfortable household's dataset, achieving a MAE value of 5.08. Below are two households from the two best performing datasets and the predictions made by the model on the datasets.





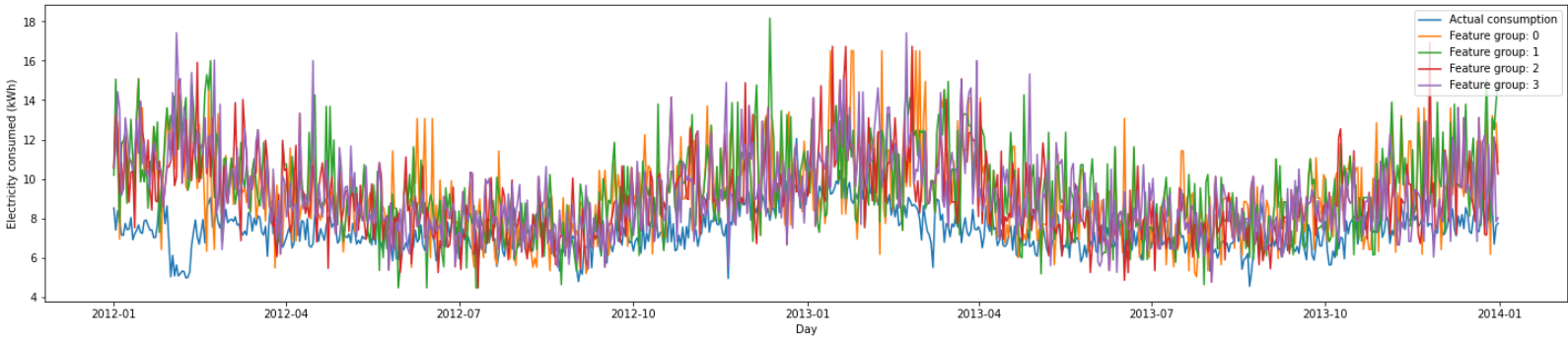
6.2.2 K Nearest Neighbours – K = 15

Dataset:	**All acorn groups**	**Affluent only**	**Comfortable only**	**Advers**
Best feature group	2 – Pearson	3 – Mutual Information	0 – Pearson	2 – Pearson
MAE	6.2778629424964105	7.8019986156474195	5.164829662654916	4.8882838
MAPE	169350359221078.28	176958539019120.72	93051823196309.83	192046032

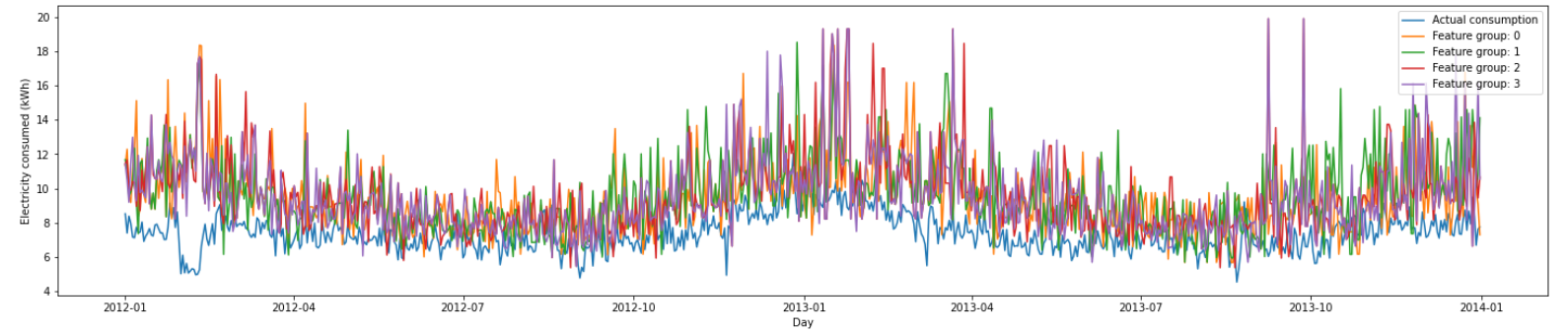
MSE	85.83656057947948	141.46335119543656	60.868770804976506	43.913502
RMSE	9.264802241790134	11.89383668945545	7.801844064384811	6.6267263
R2	0.00161976511319394 9	-0.0192538748601363	-0.00503174795942196	-0.037610

Table 10: kNN Model results

With the KNN machine learning algorithm, results achieved were like multiple linear regression. The best performing model was again the adversity dataset, which achieved a MAE value of 4.88. The second best performing model was the comfortable household's dataset like in multiple linear regression. Due to time constraints in the dissertation as well as kNN taking at least 20 minutes per run, optimisation at different K levels was not possible.



6.2.3 Support Vector Regression – Linear Kernel



Due to long training times when attempting to fit and predict the model with the RBF kernel (up to 5 hours+), I had decided to switch and use a linear kernel which sped up the training and testing time.

Dataset:	**All acorn groups**	**Affluent only**	**Comfortable only**	**Adversit**
Best feature group	0 – Pearson	1 – Mutual Information	2 – Pearson	1 – Pearson
MAE	5.485071137890237	6.821205813754089	4.925832527537741	4.428571945
MAPE	126349053784630.83	126172220355137.3	83128497084155.16	157177117846

MSE	85.90400226430432	142.98008080651405	60.23256583992725	42.33827048
RMSE	9.268441199268857	11.957427850776021	7.7609642339033655	6.506786494
R2	0.0008353389353272	-0.03018202353003518	0.0054729193704403	-0.00038994

Table 11: SVR - linear model results

With SVR, improved performance was outputted across all the datasets used to train and test the model. The best performing model was the adversity only dataset achieving a MAE of 4.428, followed by the comfortable only dataset with an MAE of 4.925

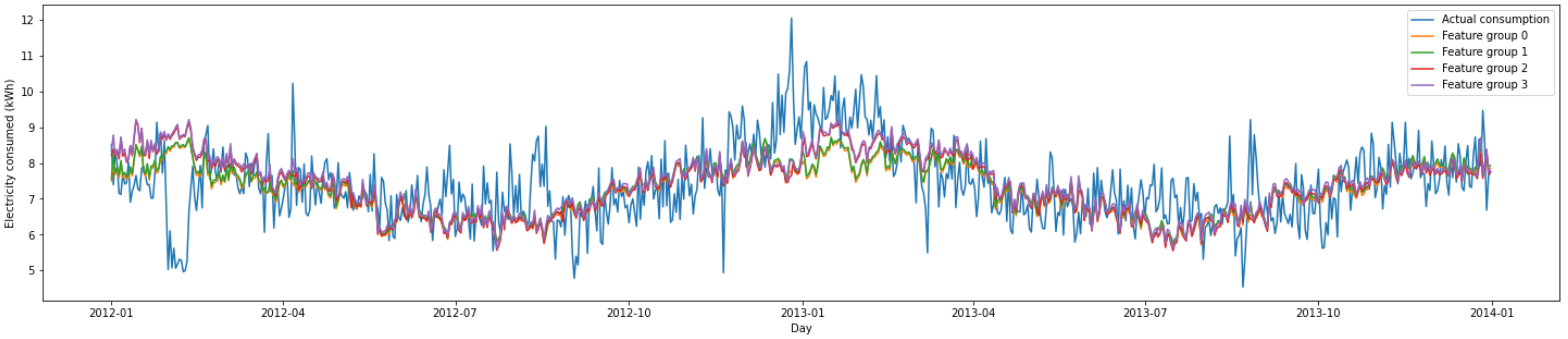
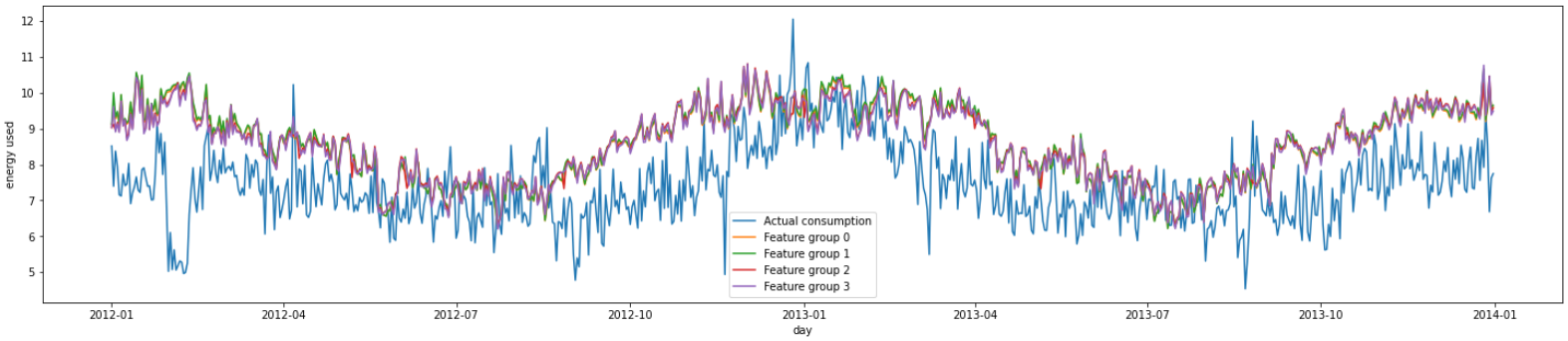


Figure : SVR: Adversity Household 3 [MAC000239]: Acorn-H predictions. Lowest RMSE:0.95, pearson feature group 1



6.2.4 Long Short Term Memory

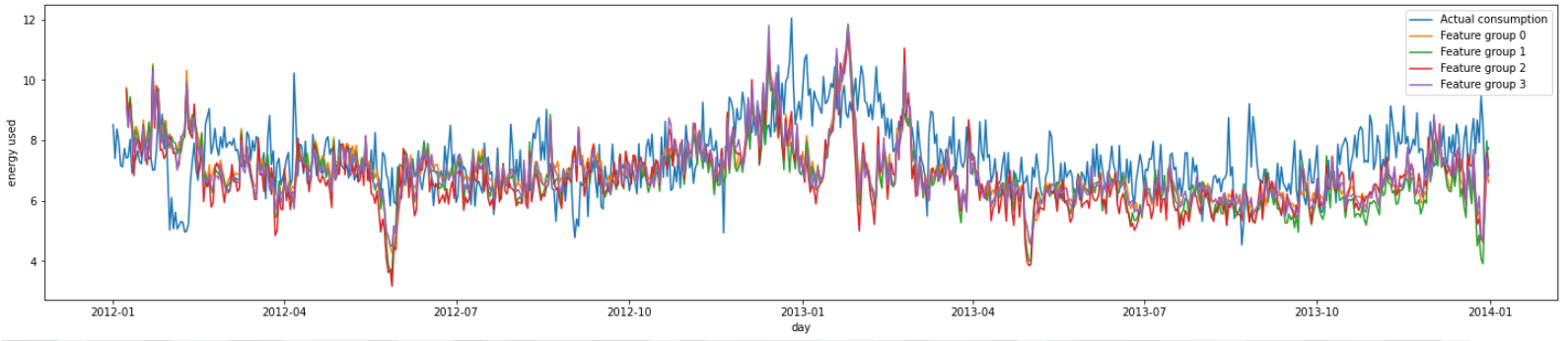
With LSTM, due to the long training times experienced (up to 10 hours at times) with the full dataset size, a subset of the training and test dataset was used rather than the full size. The below were the results of using a subset of the first 500 rows from each of the datasets.

Dataset:	**All acorn groups**	**Affluent only**	**Comfortable only**	**Adversit**
Best feature group	0 – Mutual information	2 – Mutual information	1 – Pearson	3 – Pearson
MAE	8.894320335164371	9.587787292135623	1.0430490718262337	8.197724919

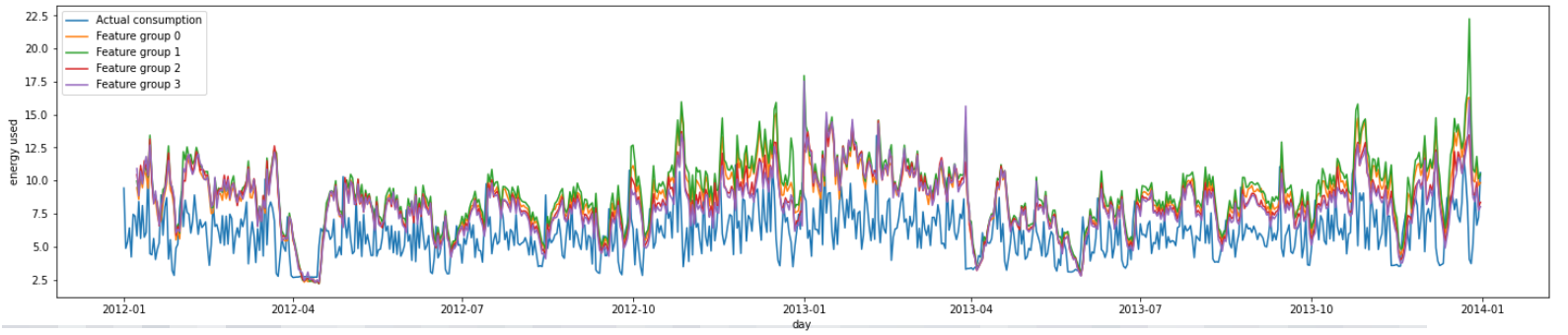
MAPE	0.7216488189942679	0.7811194250439806	0.15094202284116578	1.552546230
MSE	125.83061748463261	141.6318435090871	2.1783078638145343	77.74585620
RMSE	11.217424726051547	11.900917759109467	1.4759091651638099	8.817361068
R2	-1.691903382348893	-2.029940138670804	0.6936894525819193	-24.5844521

Table 12: LSTM model results

The results seen here, most notably with the comfortable and adversity datasets they saw significant improvements compared to the other models. The LSTM model was able to get much closer predictions to actual consumption in the select households which will be discussed further down. This is most likely due to the added historical information of prior electricity consumption being passed in as input to the model as well as each of the LSTM units keeping historical data which has proved beneficial as seen above. In addition to this, a key difference with LSTM compared to the other models was that the MAPE value from each of the test datasets were significantly lower. This model was able to adapt much better than the previous three models thanks to its memory unit and meant that there was higher accuracy when forecasting.



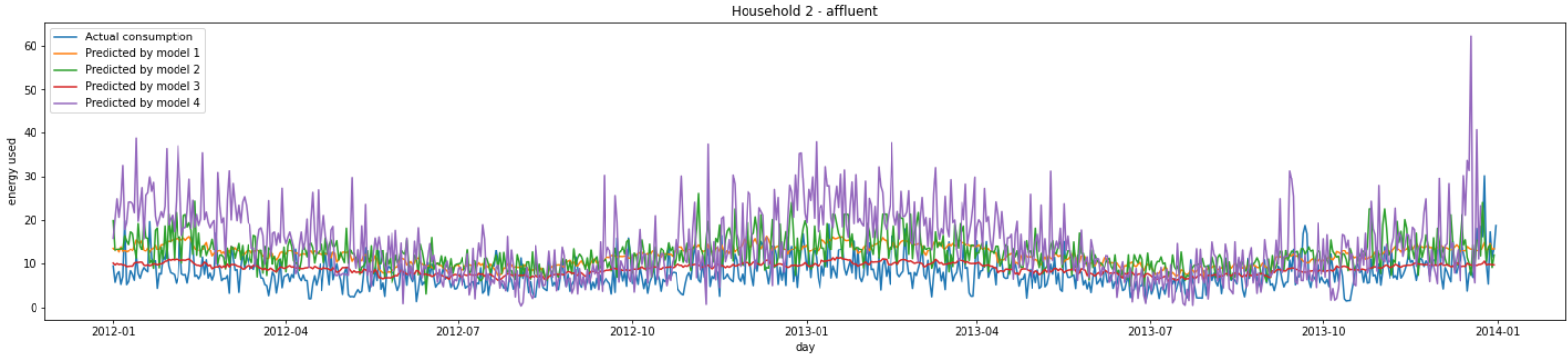
6.3 Household electricity graphical forecast against actual consumption

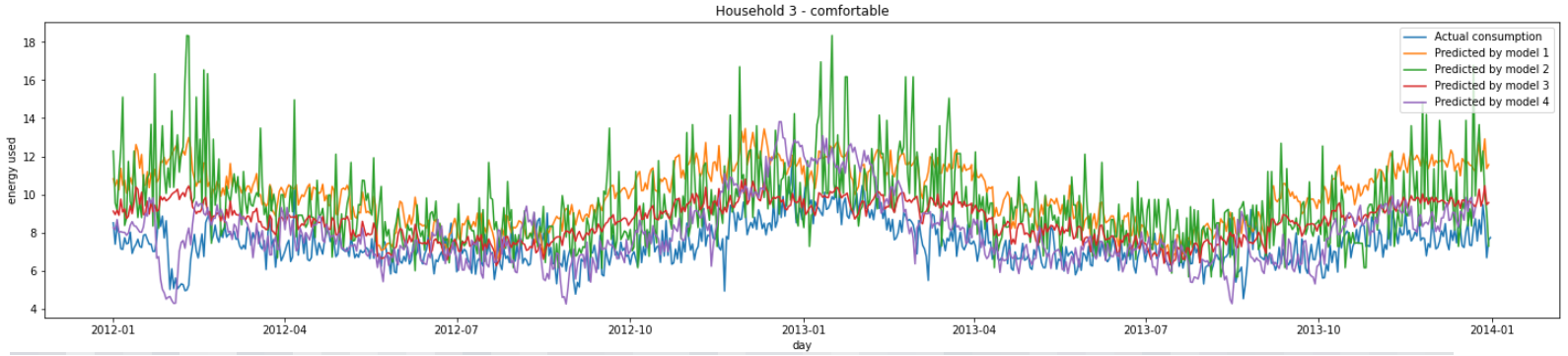


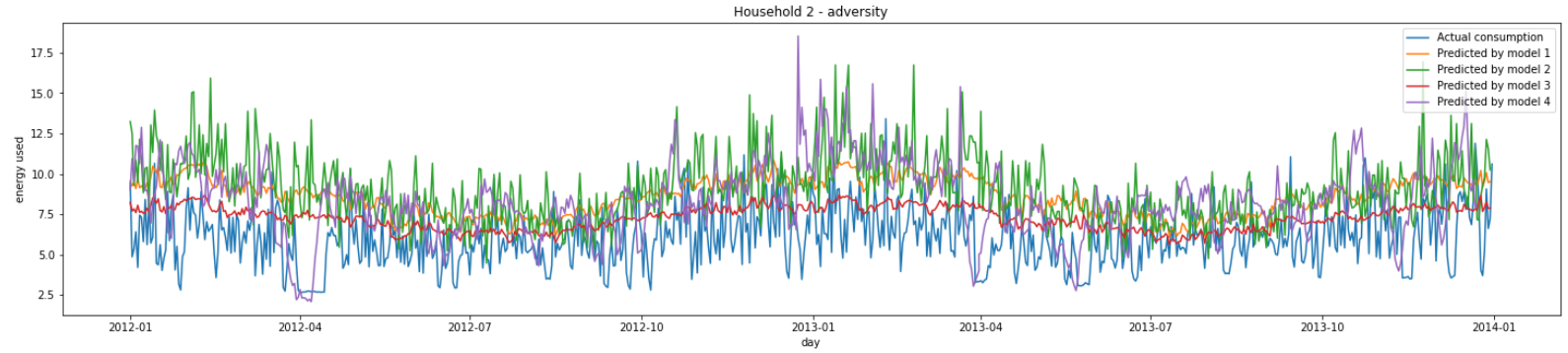
Below are the graphical representations of the results from the models developed where predictions were closest to the actual consumption of that household. (Please refer to the appendix which contains the full results)

For reference in the graph legends, models referred to are as follows:

- Model 1: Multiple Linear Regression
- Model 2: k-Nearest Neighbors
- Model 3: Support Vector Regression
- Model 4: Long Short Term Memory


Household 2 - affluent
Actual consumption
Predicted by model 1
Predicted by model 2
Predicted by model 3
Predicted by model 4
energy used
day
2012-01
2012-04
2012-07
2012-10
2013-01
2013-04
2013-07
2013-10
2014-01


Household 3 - comfortable
Actual consumption
Predicted by model 1
Predicted by model 2
Predicted by model 3
Predicted by model 4
energy used
day
2012-01
2012-04
2012-07
2012-10
2013-01
2013-04
2013-07
2013-10
2014-01


Household 2 - adversity
Actual consumption
Predicted by model 1
Predicted by model 2
Predicted by model 3
Predicted by model 4
energy used
day
2012-01
2012-04
2012-07
2012-10
2013-01
2013-04
2013-07
2013-10
2014-01

6.4 Discussion and evaluation of the best results and model for each of the datasets

Dataset:	All acorn groups	Affluent only	Comfortable only	Adversity only
Best Model	Support Vector Regression	Support Vector Regression	Long Short Term Memory	Long Short Term Memory
Best feature group	0 – Pearson	1 – Mutual Information	0 – Pearson	3 – Pearson
MAE	5.485071137890237	6.821205813754089	1.0791791736428158	1.893357803915
MAPE	126349053784630.83	126172220355137.3	0.16056443479841012	13692450602601
MSE	85.90400226430432	142.98008080651405	2.2068430091495754	9.450546520536
RMSE	9.268441199268857	11.957427850776021	1.4855446843328461	3.074174120074
R2	0.0008353389353272	-0.030182023530035	0.6896768811114546	0.776709251388

Table 13: Datasets and the machine learning model with the best performance metrics for electricity forecasting

6.4.1 Performance metrics

Overall, from the results gained from each of the models when looking at the highest performance metrics, there is a split between which model was best for which type of dataset. As seen in the table above, with the all combined acorn groups dataset and the affluent only, support vector regression was better suited for forecasting. In figure 27, it could be seen that the variation between the households were quite large and because of this, it can be concluded that households within the Affluent Acorn grouping are likely to have very unpredictable and quite varied electricity consumption patterns and behaviours which is a difficult problem when attempting to have a model that is able to generalise for forecasting electricity consumption. On the other hand, LSTM was the best model for households in the Acorn comfortable and Adversity groupings. Again, referring to figure 28 and 29, the electricity pattern and consumption between the households were quite similar to one another, stable and within a smaller range compared to the Affluent households in figure 27, which is where LSTM were able to recognise this and create better more accurate forecasts.

6.4.2 Evaluation of performance metrics

The results gained from each of the machine learning models developed within the dissertation have provided insight on how capable they are for electricity forecasting, especially with the LSTM model developed, it has shown that they can certainly provide forecasts with high accuracy which is promising. More can be done within this area however as there was not enough time available to be able to optimise each of the models:

- With kNN, different values of K could be explored that could bring improvements to the model
- With SVR the rbf kernel could bring differences in the model, this may have been quicker if a subset of data was used like with LSTM
- With LSTM, different variations of prior historic days to provide to the model could also be another way for optimisation

Chapter 7: Conclusion

This chapter provides the overall dissertation outcome against the initial project objectives as well as a reflection on how it went overall and what could be done to further extend this work.

7.1 Project objectives and outcomes

The main aim of the work within this dissertation was to investigate the London smart meter dataset with household electricity consumption and apply a variety of machine learning techniques to be able to see how well they would generalise for forecasting electricity consumption on an individual household basis rather than on average for the whole of the London area covered in the dataset. Below are the set objectives in this dissertation against how well objectives were met during this dissertation.

7.1.1 Objective 1

"Review and summarise published findings in the types of machine learning algorithms that have been applied to forecasting short term electricity consumption"

This objective was achieved in chapter 1. Background research was undertaken which discussed several academic papers that focused on machine learning algorithms being applied to a dataset with household electricity consumption data over a certain period and setting a focus on which features to use for forecasting. The insights provided and process of each paper and results were discussed within chapter 1 as well as how the models developed within the papers had performed.

7.1.2 Objective 2

"Evaluate the types of machine learning algorithms referenced and identify those that will be implemented and applied to the crowdsourced London household smart meter dataset"

This objective was partly achieved in chapter 3. Looking back at my evaluation approach to each of the models, it could have been expanded more to include other factors such as considering the general shape of the London Smart meter dataset and seeing which algorithm it would suit best from their strengths. In addition to this it may have been beneficial to take a small subset sample of the training dataset containing the consumption dataset and check the typical training time to fit on each of the models to provide expectations ahead of going into the dissertation. Academic papers involving a wider variety of other models within the deep learning area could also have been considered.

7.1.3 Objective 3

"Pre-process and analyse the dataset and investigate correlations between features to be provided as input for training each of the machine learning algorithms"

This was achieved in in chapter 4 which involved the process of pre-processing the data to a readable format for the Pandas library, dealing with null data, analysing trends within the dataset, and combining the 4 individual datasets together into one large dataset for training and testing. Correlation analysis with Pearson's method and mutual information were identify and select the top 8 features were impactful towards influencing the energy consumed by one household. The dataset was then reduced to only be set during a period between 2012 and 2013, segregated between acorn types combined and separated. 3 specific households from each acorn group and acorn type were then selected to provide a wide spectrum of different consumption patterns to see how well the models developed were able to generalise and make predictions on each household.

7.1.4 Objective 4

"Train each of the chosen machine learning algorithms and develop working models"

This objective was partly achieved in chapter 5, where 4 of the chosen machine learning algorithms were developed with a consistent flow in each of them involving scaling the dataset, fitting it to the

model and then making predictions on the 2013 test dataset as well as the selected households from each Acorn grouping. With the SVR model, a linear kernel had to be used as a rbf kernel took too long to fit to the model and make predictions which did not provide any results even after 8 hours running. With the LSTM model only a subset of the dataset was used to be able to train and make predictions on the datasets. Each of the models developed were able to make predictions and output results.

7.1.5 Objective 5

"Evaluate the effectiveness of generalisation in the dataset for predicting short term electricity consumption for households in each type of algorithm chosen"

This was achieved in chapter 6, where performance metrics involving MAE, MAPE, MSE, RMSE and R2 were used to determine and discuss how well each model did with forecasting electricity consumption for households in each of the Acorn groups and how close to generalisation of forecasting models were able to achieve. By using these performance metrics, it allowed for simple and straightforward comparison between models.

7.2 What could have gone differently

Dataset pre-processing and analysis took much longer than I had expected. The large size of the dataset meant that: some of the datasets initially had inaccurate data such as the bank holidays dataset that had to be corrected, that null values were checked and dealt with and that combining all four datasets together took longer than expected. In addition to this, going into this dissertation, I did not have much prior experience on working with machine learning models which meant that there was a steep learning curve for me and longer than expected time was spent on understanding how each of the models worked behind the scenes first and then applying it to the electricity consumption dataset. These various factors lengthened the dissertation completion and something that I would do differently in future would be to plan an overview of how the dataset should be structured and with what features.

With the machine learning models, making use of a pipeline would prevent the need for me to use different files for each of the models and would also improve my efficiency and workflow as there would not need to be stopping and starting of the Colab session each time. In addition to this, I would certainly consider using a subset of the dataset again to reduce training times and to get results effectively and quicker. Finally, one thing that became quite apparent was that there were subtle difference in energy forecasting as to what short term means, and I did not set a definition or goal as to which period of time should be focused on and instead this looked at checking how well models were performing with predictions against actual consumption.

7.3 The future and extensions to this work

This dissertation provides a strong starting point for the possibilities that machine learning algorithms can bring towards generalising for forecasting electricity consumption. For future extensions on this work, other approaches such as clustering methods to group households together based on load profiles as well as clustering with Acorn groupings could prove useful towards improving efficiency and performance in electricity forecasting.

As mentioned in the performance evaluation section, further focusing on the optimisation of the models in kNN and LSTM and making use of a wider variety of machine learning algorithms would provide more insight around this area. It would also be beneficial to explore providing an application to allow consumers to provide their electricity consumption as direct input into a machine learning model such as LSTM to get their forecasting consumption and plan accordingly to allow them to save money in the long run by changing tariffs for example.

8 References

[1] "Global electricity demand is growing faster than renewables, driving strong increase in generation from fossil fuels - News - IEA", IEA, 2022. [Online]. Available: https://www.iea.org/news/global-electricity-demand-is-growing-faster-than-renewables-driving-strong-increase-in-generation-from-fossil-fuels [Accessed: 24- Feb- 2022]

[2] "EE: Average UK Smart Home will have 50 connected devices by 2023 - Smart Cities Association", Smart Cities Association. [Online]. Available: https://smartcitiesassociation.org/index.php/media-corner/news/169 [Accessed: 22- Feb- 2022]

[3] "As it happens: the 2022 UK energy price crisis", Power Technology, 2022. [Online]. Available: https://www.power-technology.com/features/uk-power-utility-crisis-gas-price-energy-bills/ [Accessed: 26- Feb- 2022]

[4] "About the rollout | Smart Energy GB", Smartenergygb.org. [Online]. Available: https://www.smartenergygb.org/faqs/about-the-rollout [Accessed: 21- Feb- 2022]

[5] C. Smith, "Cost of living: Warning UK faces biggest income squeeze in nearly 50 years", BBC News, 2022. [Online]. Available: https://www.bbc.co.uk/news/business-60649217 [Accessed: 16- Apr- 2022]

[6] "2017 Smart Meter Customer Experience Study Executive Summary", GOV.UK, 2022. [Online]. Available: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/758010/ces-executive-summary.pdf [Accessed: 20- Feb- 2022]

[7] T. Hargreaves, M. Nye and J. Burgess, "Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors", Energy Policy, vol. 38, no. 10, pp. 6111-6119, 2010. Available: 10.1016/j.enpol.2010.05.068 [Accessed 20- Feb- 2022]

[8] A. Kell, A. McGough and M. Forshaw, "Segmenting Residential Smart Meter Data for Short-Term Load Forecasting", Proceedings of the Ninth International Conference on Future Energy Systems, 2018. Available: 10.1145/3208903.3208923 [Accessed 22- Feb- 2022]

[9] S. Sulaiman, P. Jeyanthy and D. Devaraj, "Artificial neural network based day ahead load forecasting using Smart Meter data", 2016 Biennial International Conference on Power and Energy Systems: Towards Sustainable Energy (PESTSE), 2016. Available: 10.1109/pestse.2016.7516422 [Accessed 23- Feb- 2022]

[10] K. Gajowniczek and T. Ząbkowski, "Short Term Electricity Forecasting Using Individual Smart Meter Data", Procedia Computer Science, vol. 35, pp. 589-597, 2014. Available: 10.1016/j.procs.2014.08.140 [Accessed 27- Feb- 2022]

[11] Z. Camurdan and M. Ganiz, "Machine learning based electricity demand forecasting", 2017 International Conference on Computer Science and Engineering (UBMK), 2017. Available: 10.1109/ubmk.2017.8093428 [Accessed 28 -Feb- 2022]

[12] S. Chadoulos, I. Koutsopoulos and G. Polyzos, "One model fits all", Proceedings of the Twelfth ACM International Conference on Future Energy Systems, 2021. Available: 10.1145/3447555.3466587 [Accessed 1- Mar- 2022]

[13] "Computational complexity of machine learning algorithms", The Kernel Trip, 2022. [Online]. Available: https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/ [Accessed: 17- Apr- 2022]

[14] I. Education, "What are Neural Networks?", Ibm.com, 2022. [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks [Accessed: 17- Apr- 2022]

[15] J. Brownlee, "Why Training a Neural Network Is Hard", Machine Learning Mastery, 2022. [Online]. Available: https://machinelearningmastery.com/why-training-a-neural-network-is-hard/ [Accessed: 18- Apr- 2022]

[16]"17: Decision Trees", Cs.cornell.edu, 2018. [Online]. Available: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote17.html [Accessed: 18- Apr- 2022]

[17] J. Brownlee, "K-Nearest Neighbors for Machine Learning", Machine Learning Mastery, 2016. [Online]. Available: https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/ [Accessed: 16- Apr- 2022].

[18] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm", Medium, 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 [Accessed: 16- Apr- 2022]

[19] S. Dobilas, "Support Vector Regression (SVR) — One of the Most Flexible Yet Robust Prediction Algorithms", Medium, 2020. [Online]. Available: https://towardsdatascience.com/support-vector-regression-svr-one-of-the-most-flexible-yet-robust-prediction-algorithms-4d25fbdaca60 [Accessed: 18- Apr- 2022]

[20] P. Bassey, "Logistic Regression Vs Support Vector Machines (SVM)", Medium, 2019. [Online]. Available: https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16 [Accessed: 20- Apr- 2022]

[21] J. Brownlee, "A Gentle Introduction to Long Short-Term Memory Networks by the Experts", Machine Learning Mastery, 2017. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ [Accessed: 28- Apr- 2022]

[22] "Frequently Asked Questions", Machine Learning Mastery. [Online]. Available: https://machinelearningmastery.com/faq/single-faq/what-is-the-difference-between-samples-timesteps-and-features-for-lstm-input/ [Accessed: 28- Apr- 2022]

[23] M. Alam, "Multiple regression as a machine learning algorithm", Medium, 2020. [Online]. Available:

https://towardsdatascience.com/multiple-regression-as-a-machine-learning-algorithm-a98a6b9f307b [Accessed: 25- Apr- 2022]

[24] "The Ultimate Guide to Linear Regression for Machine Learning", Keboola.com, 2020. [Online]. Available: https://www.keboola.com/blog/linear-regression-machine-learning [Accessed: 21- Apr- 2022]

[25] A. Singh, "K-Nearest Neighbors Algorithm | KNN Regression Python", Analytics Vidhya, 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/ [Accessed: 17- Apr- 2022]

[26] A. Sethi, "Support Vector Regression In Machine Learning", Analytics Vidhya, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/ [Accessed: 20- Apr- 2022]

[27] "What is LSTM - Introduction to Long Short Term Memory", Intellipaat Blog, 2022. [Online]. Available: https://intellipaat.com/blog/what-is-lstm/ [Accessed: 28- Apr- 2022]

[28]"Understanding LSTM Networks -- colah's blog", Colah.github.io, 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed: 30- Apr- 2022].

[29] "Smart meters in London", Kaggle.com. [Online]. Available: https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london [Accessed: 28- Mar- 2022]

[30]"Google Colab", Research.google.com. [Online]. Available: https://research.google.com/colaboratory/faq.html [Accessed: 01- Mar- 2022].

[31] "Energy Smart Meters", Parliament UK, 2019. [Online]. Available: https://commonslibrary.parliament.uk/research-briefings/cbp-8119/ [Accessed: 28- February- 2022]

[32] "Acorn consumer classification (CACI)", GOV.UK, 2020. [Online]. Available: https://www.gov.uk/government/statistics/quality-assurance-of-administrative-data-in-the-uk-house-price-index/acorn-consumer-classification-caci [Accessed: 31- Apr- 2022]

[33] "United Kingdom Bank Holidays", Ukbankholidays.co.uk, 2020. [Online]. Available: https://www.ukbankholidays.co.uk/ [Accessed: 01- Mar- 2022]

[34] "pandas.DataFrame.interpolate — pandas 1.4.2 documentation", Pandas.pydata.org. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html [Accessed: 16- Mar- 2022]

[35] "SmartMeter Energy Consumption Data in London Households – London Datastore", Data.london.gov.uk, 2022. [Online]. Available: https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households [Accessed: 02- May- 2022]

[36] Octopus energy, "Introducing Agile Octopus: The 100% green smart tariff with Plunge Pricing", Octopus Energy. [Online]. Available: https://octopus.energy/agile/ [Accessed: 04- Apr- 2022]

[37] D. Radečić, "Time Series From Scratch — Train/Test Splits and Evaluation Metrics", Medium, 2021. [Online]. Available: https://towardsdatascience.com/time-series-from-scratch-train-test-splits-and-evaluation-metrics-4fd654de1b37 [Accessed: 04- Apr- 2022]

[38] S. Chatterjee, "Data Correlation can make or break your Machine Learning Project", Medium, 2017. [Online]. Available: https://towardsdatascience.com/data-correlation-can-make-or-break-your-machine-learning-project-82ee11039cc9 [Accessed: 13- Mar- 2022]

[39] J. Brownlee, "Information Gain and Mutual Information for Machine Learning", Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/information-gain-and-mutual-information/ [Accessed: 28- May- 2022]

[40] P. Paznar, "What is Mutual Information?", Quantdare, 2021. [Online]. Available: https://quantdare.com/what-is-mutual-information/ [Accessed: 18- Apr- 2022]

[41] "Package overview — pandas 1.4.2 documentation", Pandas.pydata.org. [Online]. Available: https://pandas.pydata.org/docs/getting_started/overview.html [Accessed: 20- Feb- 2022].

[42] "What is NumPy? — NumPy v1.22 Manual", Numpy.org. [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html [Accessed: 19- Feb- 2022].

[43] "Matplotlib — Visualization with Python", Matplotlib.org. [Online]. Available: https://matplotlib.org/ [Accessed: 20- Feb- 2022]

[44] "About us", scikit-learn. [Online]. Available: https://scikit-learn.org/stable/about.html [Accessed: 02- Mar- 2022]

[45] "Why TensorFlow", TensorFlow. [Online]. Available: https://www.tensorflow.org/about [Accessed: 28- May- 2022]

[46] "Mean Absolute Error ~ MAE [Machine Learning(ML)]", Medium, 2022. [Online]. Available: https://medium.com/@20__80__/mean-absolute-error-mae-machine-learning-ml-b9b4afc63077 [Accessed: 17- Apr- 2022]

[47] C. Pascual, Dataquest.io, 2018. [Online]. Available: https://www.dataquest.io/blog/understanding-regression-error-metrics/ [Accessed: 17- Apr- 2022]

[48] J. Moody, "What does RMSE really mean?", Medium, 2019. [Online]. Available: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e [Accessed: 17- Apr- 2022]

[49] J. Frost, "How To Interpret R-squared in Regression Analysis", Statistics By Jim, 2018. [Online]. Available: https://statisticsbyjim.com/regression/interpret-r-squared-regression/ [Accessed: 17- Apr- 2022]

9 Appendices

Additional information is supplied here that may be useful for understanding some of the data included within this dissertation.

Feature selection groups based on Pearson's correlation method

```
#---- Pearson correlation selection ----
#For all acorn combined dataset only
pearson_0 = ['Acorn_grouped_Affluent', 'humidity', 'cloudCover', 'weather_type_partly-cloudy-night',
             'Acorn_grouped_Adversity','uvIndex', 'dewPoint', 'temperatureHigh']

pearson_1 = ['Acorn_grouped_Affluent', 'humidity', 'cloudCover', 'weather_type_partly-cloudy-night',
             'Acorn_grouped_Adversity', 'uvIndex', 'dewPoint', 'temperatureHigh', 'public_holiday']

pearson_2 = ['Acorn_grouped_Affluent', 'humidity', 'cloudCover', 'weather_type_partly-cloudy-night',
             'Acorn_grouped_Adversity', 'uvIndex', 'dewPoint', 'temperatureHigh', 'month', 'day_of_week']

pearson_3 = ['Acorn_grouped_Affluent', 'humidity', 'cloudCover', 'weather_type_partly-cloudy-night',
             'Acorn_grouped_Adversity', 'uvIndex', 'dewPoint', 'temperatureHigh',  'month', 'day_of_week', 'public_holiday']

#For specific acorn datasets
pearson_4 = ['humidity', 'cloudCover', 'weather_type_partly-cloudy-night', 'weather_type_fog',
             'visibility', 'uvIndex', 'dewPoint', 'temperatureHigh']

pearson_5 = ['humidity', 'cloudCover', 'weather_type_partly-cloudy-night', 'weather_type_fog',
             'visibility', 'uvIndex', 'dewPoint', 'temperatureHigh', 'public_holiday']

pearson_6 = ['humidity', 'cloudCover', 'weather_type_partly-cloudy-night', 'weather_type_fog',
             'visibility', 'uvIndex', 'dewPoint', 'temperatureHigh', 'day_of_week', 'month']

pearson_7 = ['humidity', 'cloudCover', 'weather_type_partly-cloudy-night', 'weather_type_fog',
             'visibility', 'uvIndex', 'dewPoint', 'temperatureHigh', 'day_of_week', 'month', 'public_holiday']
```

Feature selection groups based on Mutual Information

```
#----- Mutual information feature selection ----
#For all acorn combined dataset only
mutual_information_0 = ['month', 'Acorn_grouped_Affluent', 'temperatureHigh', 'dewPoint', 'windSpeed',
                        'pressure', 'visibility', 'uvIndex']

mutual_information_1 = ['month', 'Acorn_grouped_Affluent', 'temperatureHigh', 'dewPoint', 'windSpeed',
                        'pressure', 'visibility', 'uvIndex', 'public_holiday']

mutual_information_2 = ['month', 'Acorn_grouped_Affluent', 'temperatureHigh', 'dewPoint', 'windSpeed',
                        'pressure', 'visibility', 'uvIndex', 'day_of_week', 'month']

mutual_information_3 = ['month', 'Acorn_grouped_Affluent', 'temperatureHigh', 'dewPoint', 'windSpeed',
                        'pressure', 'visibility', 'uvIndex', 'day_of_week', 'month', 'public_holiday']


#----
#For specific acorn datasets
mutual_information_4 = ['month', 'temperatureHigh', 'dewPoint', 'windSpeed', 'pressure', 'visibility',
                        'humidity', 'uvIndex']

mutual_information_5 = ['month', 'temperatureHigh', 'dewPoint', 'windSpeed', 'pressure', 'visibility',
                        'humidity', 'uvIndex', 'public_holiday']

mutual_information_6 = ['day_of_week', 'month', 'temperatureHigh', 'dewPoint', 'windSpeed', 'pressure', 'visibility',
                        'humidity', 'uvIndex', 'day_of_week', 'month']

mutual_information_7 = ['day_of_week', 'month', 'temperatureHigh', 'dewPoint', 'windSpeed', 'pressure', 'visibility',
                        'humidity', 'uvIndex', 'day_of_week', 'month', 'public_holiday']
```

Run_model function for MLR

```
#Run the model and see performance for each group
def run_model(X_train, X_test, y_train, y_test, feature_groups, household_groups):
  #Store household predictions in arrays
  household_1_predictions = []
  household_2_predictions = []
  household_3_predictions = []

  print("-- Overall model performance per feature group: --\n")
  print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format('Feature group', 'MAE', 'MAPE', 'MSE', 'RMSE', 'R2'))
  #Loop through each of the feature groups passed into the function
  for i in range(0, len(feature_groups)):
    #Fit the data to the model
    model = LinearRegression()
    model.fit(X_train[feature_groups[i]], y_train)

    #Create predictions for the test set
    y_pred = model.predict(X_test[feature_groups[i]])

    #Get performance metrics for each feature group in order of: #MAE, MAPE, MSE, RMSE, R2
    mae = mean_absolute_error(y_test, y_pred)
    mape = mean_absolute_percentage_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format(i, mae, mape, mse, rmse, r2))
    household_1_predictions.append(model.predict(household_groups[0][feature_groups[i]]))
    household_2_predictions.append(model.predict(household_groups[1][feature_groups[i]]))
    household_3_predictions.append(model.predict(household_groups[2][feature_groups[i]]))

  return household_1_predictions, household_2_predictions, household_3_predictions
```

Run_model function for KNN

```
def run_model(X_train, X_test, y_train, y_test, k_neighbors, feature_groups, household_groups):
  #Apply feature scaling
  scaler = MinMaxScaler()

  #Arrays to store results
  household_1_predictions = []
  household_2_predictions = []
  household_3_predictions = []


  print("-- Model performance per feature group: --\n")
  print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format('Feature group', 'MAE', 'MAPE', 'MSE', 'RMSE', 'R2'))

  for i in range(0, len(feature_groups)):
    #Scale the dataset first
    X_train_scaled = scaler.fit_transform(X_train[feature_groups[i]])
    X_test_scaled = scaler.fit_transform(X_test[feature_groups[i]])

    #Fit to model
    model = KNeighborsRegressor(n_neighbors = k_neighbors)
    model.fit(X_train_scaled, y_train)

    y_pred = model.predict(X_test_scaled)
    mae = mean_absolute_error(y_test, y_pred)
    mape = mean_absolute_percentage_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format(i, mae, mape, mse, rmse, r2))

    #Scale the dataset for each household also
    household_1_scaled = scaler.fit_transform(household_groups[0][feature_groups[i]])
    household_2_scaled = scaler.fit_transform(household_groups[1][feature_groups[i]])
    household_3_scaled = scaler.fit_transform(household_groups[2][feature_groups[i]])

    #Then make predictions for that household
    household_1_predictions.append(model.predict(household_1_scaled))
    household_2_predictions.append(model.predict(household_2_scaled))
    household_3_predictions.append(model.predict(household_3_scaled))

  return household_1_predictions, household_2_predictions, household_3_predictions
```

Run model function for SVR

```
def run_model(X_train, X_test, y_train, y_test, feature_groups, household_groups):

  scaler = MinMaxScaler()

  household_1_predictions = []
  household_2_predictions = []
  household_3_predictions = []

  print("-- Model performance per feature group: --\n")
  print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format('Feature group', 'MAE', 'MAPE', 'MSE', 'RMSE', 'R2'))

  for i in range(0, len(feature_groups)):

    #Scale the dataset first
    X_train_scaled = scaler.fit_transform(X_train[feature_groups[i]])
    X_test_scaled = scaler.fit_transform(X_test[feature_groups[i]])

    #then fit the dataset to the model itself
    #model = SVR(kernel = 'rbf') #linear vs rbf #Also try linear to see results
    #model = SVR(kernel = 'linear')

    model = LinearSVR(verbose=0, dual=True)
    model.fit(X_train_scaled, y_train)

    y_pred = model.predict(X_test_scaled)

    mae = mean_absolute_error(y_test, y_pred)
    mape = mean_absolute_percentage_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format(i, mae, mape, mse, rmse, r2))

    #Scale the dataset for each household also
    household_1_scaled = scaler.fit_transform(household_groups[0][feature_groups[i]])
    household_2_scaled = scaler.fit_transform(household_groups[1][feature_groups[i]])
    household_3_scaled = scaler.fit_transform(household_groups[2][feature_groups[i]])

    #Then make predictions for that household
    household_1_predictions.append(model.predict(household_1_scaled))
    household_2_predictions.append(model.predict(household_2_scaled))
    household_3_predictions.append(model.predict(household_3_scaled))

  return household_1_predictions, household_2_predictions, household_3_predictions
```

Run_model function for LSTM – part 1

```
#Run the model and see performance for each group
def run_model(training_dataset, testing_dataset, subset, feature_groups, household_groups):
  training_dataset_subset = training_dataset[:subset]
  testing_dataset_subset = testing_dataset[:subset]

  scaler = MinMaxScaler(feature_range=(0, 1))

  household_1_predictions = []
  household_2_predictions = []
  household_3_predictions = []
  model_history = []


  print("-- Model performance per feature group: --\n")
  print("{:<15} {:<23} {:<23} {:<23} {:<23} {:<23}".format('Feature group', 'MAE', 'MAPE', 'MSE', 'RMSE', 'R2'))

  #Make training and test datasets have prior energy consumption of 7 days
  X_train = create_sliding_window(training_dataset_subset, 7, 1)
  y_train = X_train['energy_sum']

  X_test = create_sliding_window(testing_dataset_subset, 7, 1)
  y_test = X_test['energy_sum']

  X_train.drop(['energy_sum'], axis=1, inplace=True)
  X_test.drop(['energy_sum'], axis=1, inplace=True)

  household_1 = create_sliding_window(household_groups[0], 7, 1)
  household_2 = create_sliding_window(household_groups[1], 7, 1)
  household_3 = create_sliding_window(household_groups[2], 7, 1)

  for i in range(0, len(feature_groups)):

    #Scale the datasets first
    X_train_scaled = scaler.fit_transform(X_train[feature_groups[i]])
    X_test_scaled = scaler.fit_transform(X_test[feature_groups[i]])
    household_1_scaled = scaler.fit_transform(household_1[feature_groups[i]])
    household_2_scaled = scaler.fit_transform(household_2[feature_groups[i]])
    household_3_scaled = scaler.fit_transform(household_3[feature_groups[i]])


    #Reshape to the expected input shape for LSTM
    #[samples, timesteps, features]
    X_train_scaled = X_train_scaled.reshape((X_train_scaled.shape[0], 1, X_train_scaled.shape[1]))
    X_test_scaled = X_test_scaled.reshape((X_test_scaled.shape[0], 1, X_test_scaled.shape[1]))

    household_1_scaled = household_1_scaled.reshape((household_1_scaled.shape[0], 1, household_1_scaled.shape[1]))
    household_2_scaled = household_2_scaled.reshape((household_2_scaled.shape[0], 1, household_2_scaled.shape[1]))
    household_3_scaled = household_3_scaled.reshape((household_3_scaled.shape[0], 1, household_3_scaled.shape[1]))
```

Run_model function for LSTM – part 2

```
    # design network
    model = Sequential()
    # define parameters
    #verbose - if output is needed on progress of model
    verbose, epochs, batch_size = 0, 50, 72

    n_timesteps, n_features, n_outputs = X_train_scaled.shape[1], X_train_scaled.shape[2], y_train.shape[0]
    model.add(LSTM(200, activation='relu', input_shape=(n_timesteps, n_features)))
    model.add(RepeatVector(1))
    model.add(LSTM(200, activation='relu', return_sequences=True))
    model.add(TimeDistributed(Dense(100, activation='relu')))
    model.add(TimeDistributed(Dense(1)))
    model.compile(loss='mse', optimizer='adam')

    # fit network
    model_history.append(model.fit(X_train_scaled, y_train, epochs=epochs, batch_size=batch_size, verbose=verbose))

    # Make predictions on the test dataset
    y_pred = model.predict(X_test_scaled)
    y_pred = y_pred.reshape(-1)

    mae = mean_absolute_error(y_test, y_pred)
    mape = mean_absolute_percentage_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print("{:<15} {:<23} {:<23} {:<23} {:<23}".format(i, mae, mape, mse, rmse, r2))
    #Then make predictions for that household
    household_1_predictions.append(model.predict(household_1_scaled))
    household_2_predictions.append(model.predict(household_2_scaled))
    household_3_predictions.append(model.predict(household_3_scaled))

  #Set correct shape for households
  household_1_predictions = reshape_predictions(household_1_predictions)
  household_2_predictions = reshape_predictions(household_2_predictions)
  household_3_predictions = reshape_predictions(household_3_predictions)

  # plot error history and show outcome
  plt.clf()
  plt.figure(figsize = (25,5))
  for history in model_history:
    plt.plot(history.history['loss'], label='train')
  plt.legend()
  plt.show()
  return household_1_predictions, household_2_predictions, household_3_predictions
```

Get_performance function

```
#Get the performance metrics: MAE, MAPE, MSE, RMSE, R2
def get_performance(household_data, household_predictions):
  household_consumption = household_data['energy_sum']

  print("-- Test household forecasting performance per feature group: --\n")
  print("{:<15} {:<23} {:<23} {:<23} {:<23}".format('Feature group', 'MAE', 'MAPE', 'MSE', 'RMSE', 'R2'))

  for i in range(0, len(household_predictions)):
    mae = mean_absolute_error(household_consumption, household_predictions[i])
    mape = mean_absolute_percentage_error(household_consumption, household_predictions[i])
    mse = mean_squared_error(household_consumption, household_predictions[i])
    rmse = np.sqrt(mse)
    r2 = r2_score(household_consumption, household_predictions[i])

    print("{:<15} {:<23} {:<23} {:<23} {:<23}".format(i, mae, mape, mse, rmse, r2))
```

Visualise results function

```
#Visualise the results against household prediction against each feature group
def visualise_results (household_data, household_predictions):
  household_data_consumption = household_data['energy_sum']
  household_data['day']=  pd.to_datetime(household_data['day'],format='%Y-%m-%d')

  plt.clf()
  plt.figure(figsize = (25,5))
  plt.plot(household_data['day'], household_data_consumption, label='Actual consumption')

  for i in range(0, len(household_predictions)):
    plt.plot(household_data['day'], household_predictions[i], label='Feature group '+str(i))

  plt.xlabel('day')
  plt.ylabel('energy used')
  plt.legend()
  plt.show()
```

Multiple Linear Regression results

Acorn combined dataset results

Pearson

-- Overall model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.815650291300232	155789447501784.5	81.15178939742358	9.008428797377686	0.05610916824813339
1	5.8164594152963245	155822207022624.7	81.15745710555075	9.008743369946263	0.05604324613134459
2	5.835792009952841	156754174084337.34	81.09949696028866	9.005525912476665	0.056717390843693094
3	5.836897390292406	156820951602419.16	81.10391615786524	9.00577126946189	0.056665990374187736

Mutual Information

-- Overall model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.895387566864048	160403260267505.66	81.54565604781948	9.030263343215383	0.05152803549675489
1	5.896259219865349	160434741894554.56	81.55651742934543	9.03086471105317	0.05140170484503881
2	5.893171423565357	160453782283624.53	81.52563822827287	9.029154901111891	0.051760866300388564
3	5.893866455115475	160501166783012.47	81.53581719021716	9.029718555426696	0.05164247299166047

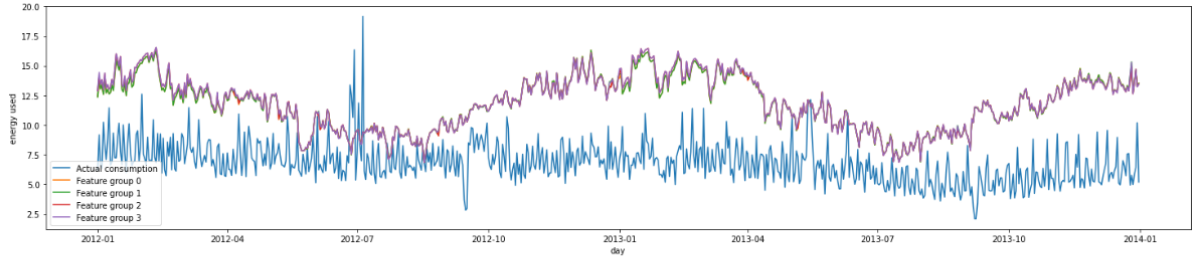
MLR - Affluent households results

MLR - Affluent households Pearson's correlation method

-- Overall model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	7.37928060717557	171176309698622.34	132.8713676295043	11.526984325030737	0.042652000183163175
1	7.380803233332948	171218488334481.38	132.88102827991082	11.527403362419085	0.04258239448474499
2	7.3973851957668995	171992979961275.56	132.78384859643384	11.523187432148877	0.04328258126884543
3	7.3989312035831984	172039678053591.9	132.79463740775836	11.523655557493827	0.04320484708783867

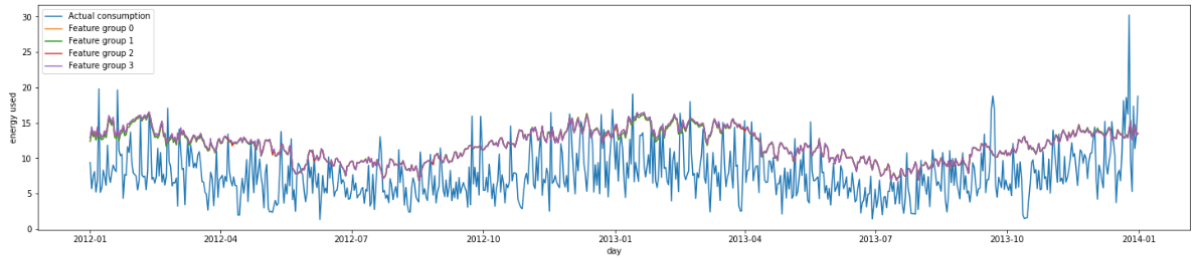
Household 1 - MAC004848 Acorn C



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.078743577070778	0.8249289750871017	30.83796209329797	5.553193864191846	-9.222311834382277
1	5.081912081583361	0.8252603253940461	30.869504165714428	5.556033132164929	-9.232767547998119
2	5.146825424935025	0.8341489354994759	31.68884723371687	5.629284788826807	-9.504367218401784
3	5.150155037558692	0.8345412341111644	31.72363770757675	5.632374073832166	-9.515899727313547

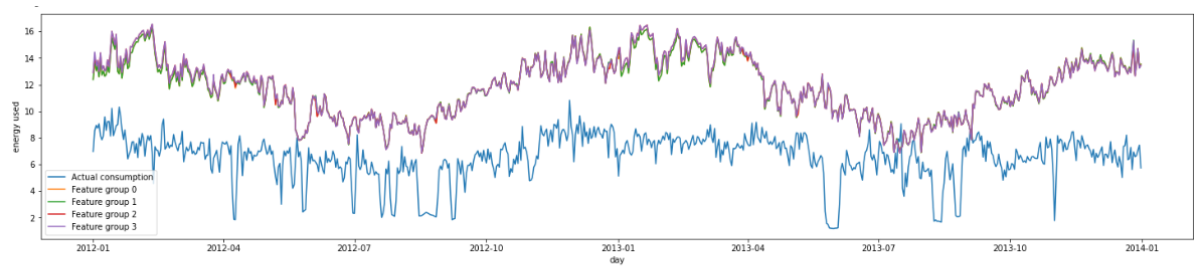
Household 2 - MAC004475 Acorn D



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.631044915288172	0.8703389178235555	27.97850304180266	5.289470960483918	-1.3790931588809494
1	4.629987575179804	0.8704914464819712	27.985067517813633	5.29009144701806	-1.3796513552914575
2	4.689625507915545	0.8786454830690069	28.663645358773625	5.353843979681667	-1.4373527947424551
3	4.686739928429907	0.8785873811727264	28.650771713739104	5.352641564100767	-1.436258111414125

Household 3 - MAC004510 Acorn E



-- Test household forecasting performance per feature group: --

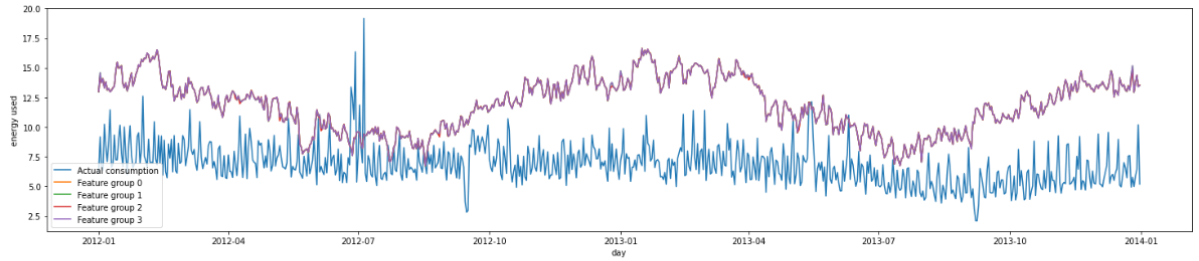
Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.254954154091922	0.9954818781599644	31.136291724724813	5.579990297905975	-9.802586143171176
1	5.257742032896015	0.9965819495343501	31.193944916604035	5.585153974296862	-9.822588639846446
2	5.3236074659798645	1.0044218804744163	31.98936828253457	5.655914451486565	-10.098556937764611
3	5.326472598197446	1.005655929571942	32.05069673126618	5.661333476422863	-10.119834547067205

MLR - Affluent households Mutual Information

-- Overall model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	7.3981949753105845	1719323891886627.25	132.78656573516386	11.523305330293208	0.043263004083972 32
1	7.399177508433423	1719649087295 21.16	132.79554964990749	11.523695138709089	0.043198274315898 51
2	7.39801151040673	1719345404282 25.38	132.78574679887836	11.523269796324234	0.043268904580286094
3	7.399006992239761	1719668642310 54.72	132.79451485114782	11.523650239882667	0.043205730117253 15

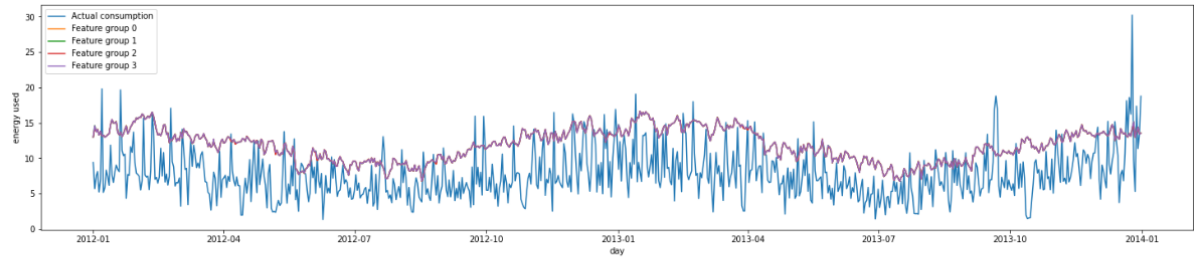
Household 1 - MAC004848 Acorn C



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.146547831627051	0.8345044770269269	31.69085650198714	5.629463251677476	-9.505033260040834
1	5.148632750897292	0.8347359848278876	31.71094225424944	5.631246953761613	-9.511691379096568
2	5.146343917027838	0.8345335918538291	31.690376066899166	5.629420580033008	-9.50487400317196
3	5.148543755829149	0.8348075635667158	31.715941176207018	5.631690791956446	-9.513348445115765

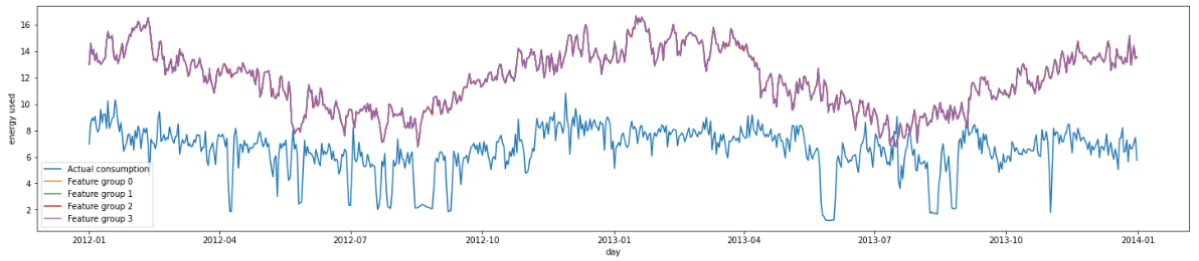
Household 2 - MAC004475 Acorn D



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.682415708904106	0.8781110864018175	28.617370632957545	5.349520598423521	-1.4334179207624702
1	4.681370885048993 5	0.8781946967555496	28.61838943025447	5.349615820809422	-1.4335045520477743
2	4.680453792073581	0.8778639704472095	28.592715510753806	5.347215678346424	-1.4313214243030754
3	4.677866747049814	0.8777454102970524	28.577251886728888	5.345769531763307	-1.4300065075588089

Household 3 - MAC004510 Acorn E



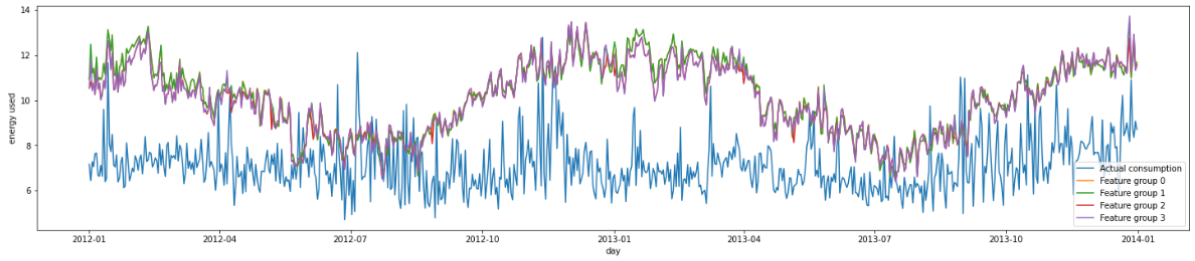
-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.322698963276779	1.0035827384500235	31.95423051755855	5.652807312969243	-10.08636605978891
1	5.3244491512710415	1.0043760849234757	31.99297491160485	5.656233279454168	-10.099808240314145
2	5.322550972115845	1.0036604036530659	31.952504516656575	5.652654643320833	-10.085767232106038
3	5.324392084562193	1.004536832892309	31.995864070121353	5.656488669671438	-10.10081061991772

MLR - Comfortable households results

MLR - Comfortable households Pearson's correlation method

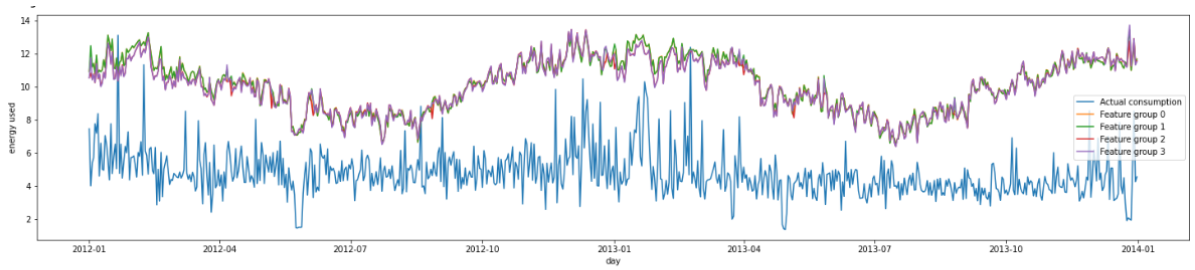
Household 1 - MAC000244 Acorn F



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.151442318182913	0.4658176486349187	12.385268295252876	3.5192709891755816	-8.09316391217897
1	3.159131973617376	0.4668733369395326	12.453074857475073	3.528891448808687	-8.142946942300659
2	3.0477344859346553	0.4509457098135252	11.522670888791618	3.3945059859708038	-7.459851865949087
3	3.0587164705440975	0.4524587583591964	11.613344712231871	3.407835781288745	-7.5264238544947215

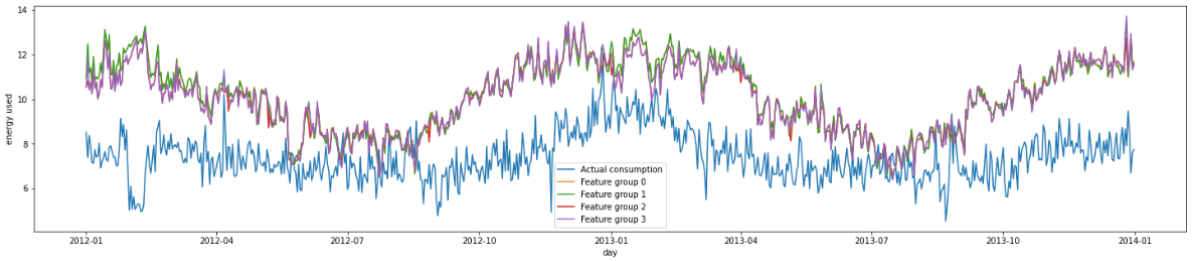
Household 2 - MAC004850 Acorn G



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.40267734568248	1.2893454529785229	32.22967413481364	5.677118471091972	-15.44984772266103
1	5.409076512683393	1.2912163506751446	32.33144735994611	5.6860748640820855	-15.501792214829035
2	5.295642140048543	1.2672558231965847	30.98526733495888	5.5664411732235 96	-14.814709362991941
3	5.303573907010438	1.2696139339536403	31.11696687377105	5.578258408658661	-14.881928080424297

Household 3 - MAC000239 Acorn H



-- Test household forecasting performance per feature group: --

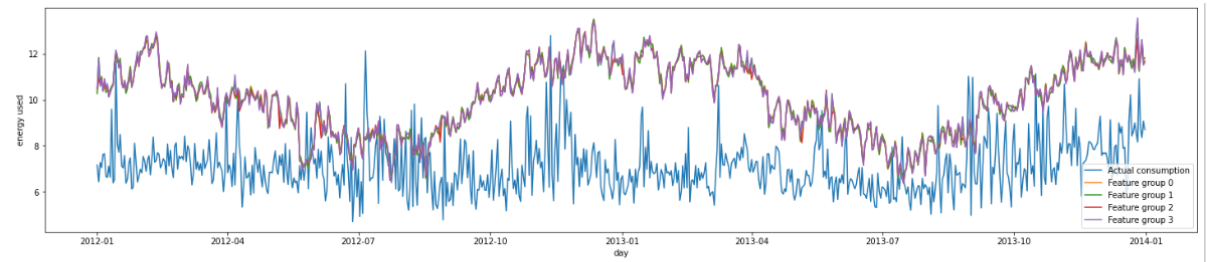
Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.7887884835964774	0.38982826720971664	9.565913843170913	3.092881155681691	-7.5384100778323955
1	2.792340384620425	0.39020161295439787	9.594595050967927	3.097514334263512	-7.5640105502716555
2	2.6856004068614947	0.37657298504032716	8.897125212937926	2.982804923714913	-6.9414580590351855
3	2.691396821646542	0.37720506759516387	8.936457969715255	2.9893909027952925	-6.976565965332734

MLR - Comfortable households Mutual Information

-- Overall model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.083130737269007	93884264649116.4	58.22153123514904	7.630303482506383	0.038678019413004194
1	5.084470563011371	93931863284488.22	58.22323525079777	7.630415142755849	0.03864988364151822
2	5.078774998367078	93938916925659.2	58.176694799745164	7.6273648660428695	0.03941833403494055
3	5.082435025543177	94193144985901.73	58.1696128563161	7.6269006061647415	0.039535267199342194

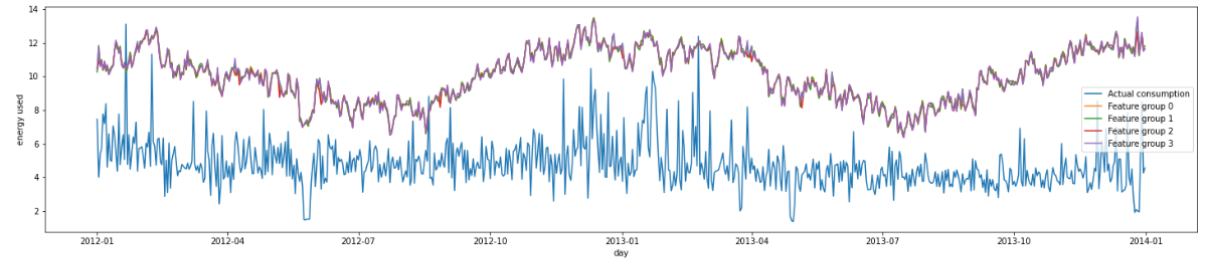
Household 1 - MAC000244 Acorn F



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.0475366020210615	0.45085590027833644	11.493420502671274	3.3901947588112504	-7.438376468796102
1	3.053785446070203	0.4517217025407611	11.546489033892572	3.3980125123213676	-7.477338955636924
2	3.043257904801837	0.4502645407387651	11.467534726584216	3.3863748650414083	-7.419371341144261
3	3.063271463549297	0.45319324477340167	11.609627722112037	3.4072903783082586	-7.523694870381187

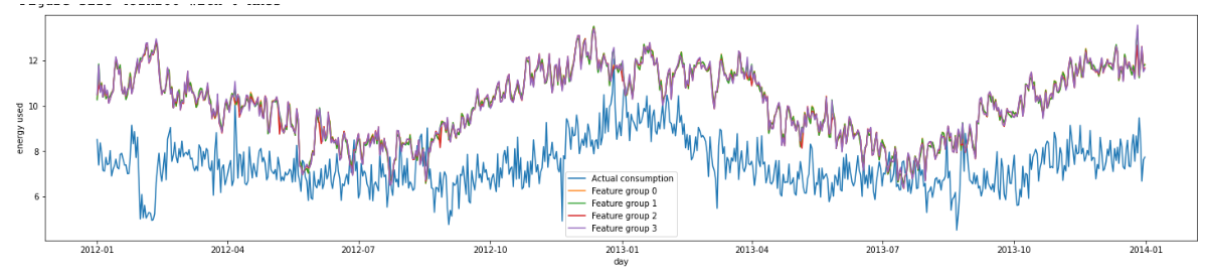
Household 2 - MAC004850 Acorn G



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.298970248508717	1.268376345891289	30.994735841276526	5.567291607350609	-14.819542036336227
1	5.304429637019337	1.269944923310922	31.07648539774371	5.574628722860717	-14.861266558577912
2	5.2885747696413485	1.2660943706662615	30.913696172509695	5.5600086486002605	-14.778179836857168
3	5.3082937445236755	1.2710556137974378	31.148978550690654	5.5811269964668115	-14.898266663571981

Household 3 - MAC000239 Acorn H



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.6839887497667205	0.37632690705873745	8.87036968030719	2.978316584969971	-6.917576419162885
1	2.6866470436022762	0.37661384473472925	8.892008340196544	2.981947071997849	-6.936890805085558
2	2.679038034604108	0.37571622226248175	8.848485147408207	2.9746403391684524	-6.898042570194892
3	2.6953105397318464	0.3779002889973064	8.942341695454607	2.9903748419645666	-6.98181770227828

MLR - Adversity households results

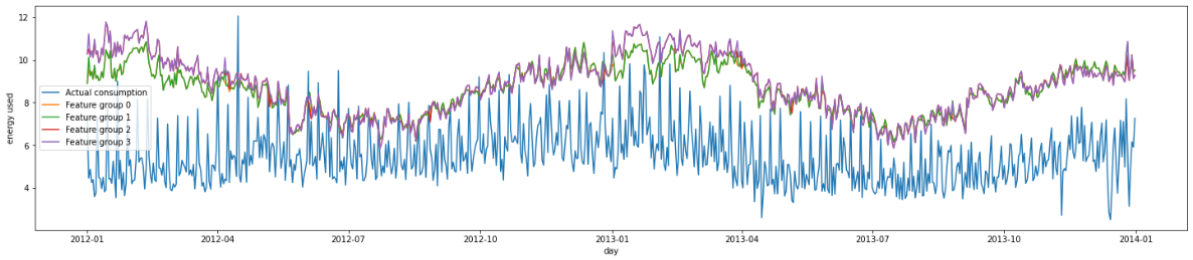
MLR - Adversity households Pearson's correlation

-- Overall model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.617104899888959	182666956056674.6	41.07046096246888	6.4086239523371065	0.02956649798322375
1	4.618982071255672	182804036598508.34	41.07267878964583	6.408796984586563	0.02951409404755423
2	4.672079818695016	185770471804606.72	41.11870650803817	6.412386958694724	0.028426527974454996
3	4.674673008253793	185979023829093.3	41.12389379705027	6.412791420048704	0.028303960101487058

<Figure size 432x288 with 0 Axes>

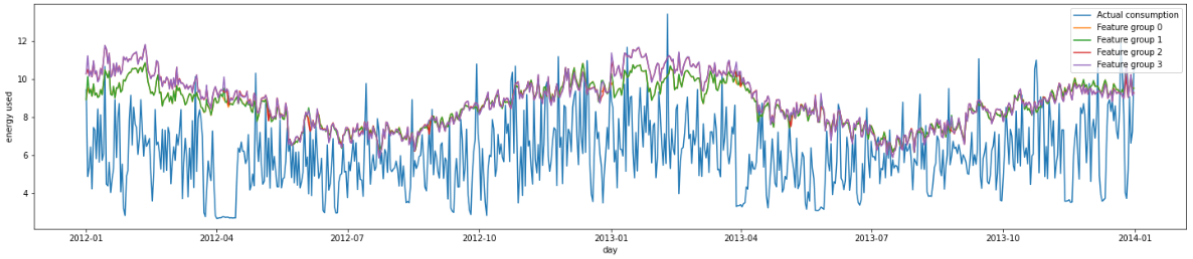
Household 1 - MAC004508 Acorn K



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.1542516102945966	0.6449016750028064	11.843730558926556	3.4414721499565495	-5.612101744746796
1	3.1587031808185375	0.6459468011762807	11.89155298806403	3.44841311157234	-5.638799985268622
2	3.343422125714972	0.680775831274033	13.534197873753474	3.6788854118813585	-6.555853531926166
3	3.3478592762659947	0.6817399315627429	13.58628703262519	3.6859580888318835	-6.584933796500837

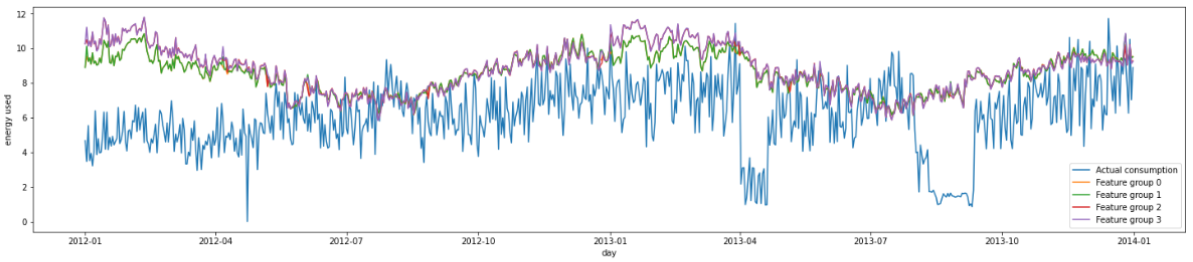
Household 2 - MAC004507 Acorn L



-- Test household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.7307015895440725	0.574869842861508	10.10946328468708	3.1795382187806895	-2.123662862627314
1	2.7363214871127646	0.5764509602889657	10.177940709486803	3.1902884994129925	-2.1448212943611966
2	2.9221725724798984	0.609546199466652	11.599184944708693	3.405757616846609	-2.5839630876758153
3	2.9283124034562005	0.6114986901846713	11.694791360656387	3.4197648107225715	-2.6135039448424924

Household 3 - MAC004513 Acorn Q



-- Test household forecasting performance per feature group: --

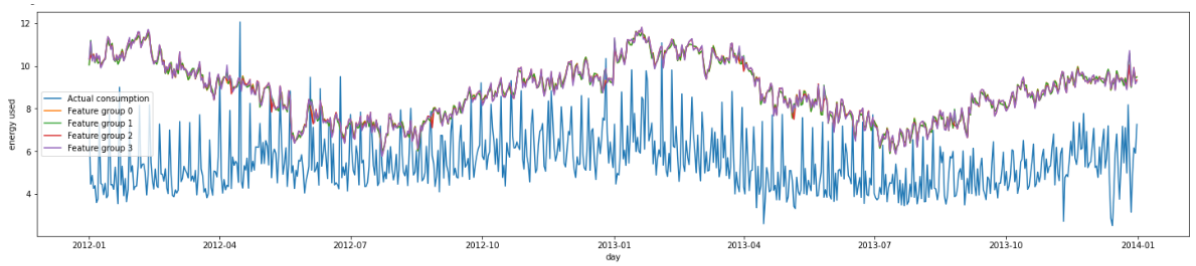
Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.6965909849163725	3.6751614020712386	10.457313567230514	3.2337769816780058	-1.7065659774603459
1	2.6998357787237923	3.6698989500757144	10.470957297666244	3.235885859801956	-1.7100972530949123
2	2.8994336341853617	3.7401127311394036	12.144441261176288	3.4848875535914052	-2.1432290254513697
3	2.9052496114683635	3.722983666769924	12.161650702415857	3.4873558324919838	-2.147683179747372

MLR - Adversity households Mutual information

```
-- Overall model performance per feature group: --
```

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.676029166179607	185975799299903.0	41.14105488297821	6.4141293160473625	0.02789846933451756
1	4.677715824898773	186100125457577.94	41.14515942758384	6.414449269234565	0.02780148509079394 3
2	4.678283908609679	186439648997943.97	41.119934569886546	6.412482714977604	0.028397510711627105
3	4.678641881003521	186496175614164.56	41.11800509690439	6.412332266570751	0.02844310126939653

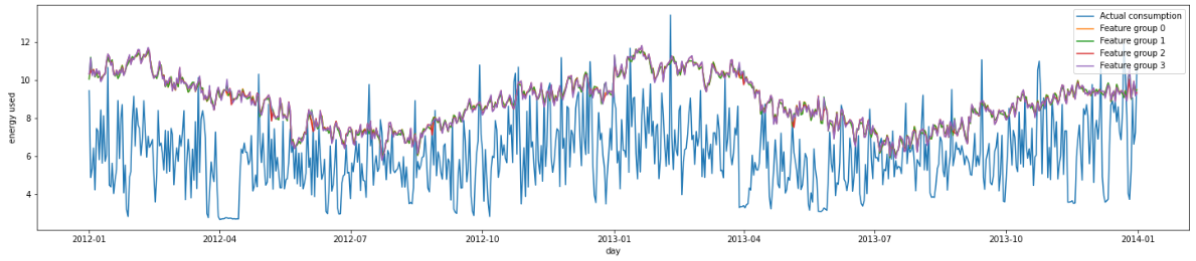
Household 1 - MAC004508 Acorn K




```
-- Test household forecasting performance per feature group: --
```

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.3538769230733183	0.6837525368825974	13.63811154890639	3.6929813902735	-6.613866316775484
1	3.35794292273711	0.6847401773224681	13.685274319256223	3.699361339374166	-6.640196283888943
2	3.3594647211033437	0.68395169894277	13.651957712602963	3.694855573984315	-6.62159633416153
3	3.3576839516176125	0.6836993704000336	13.654691906608502	3.6952255555795914	-6.623122776262231

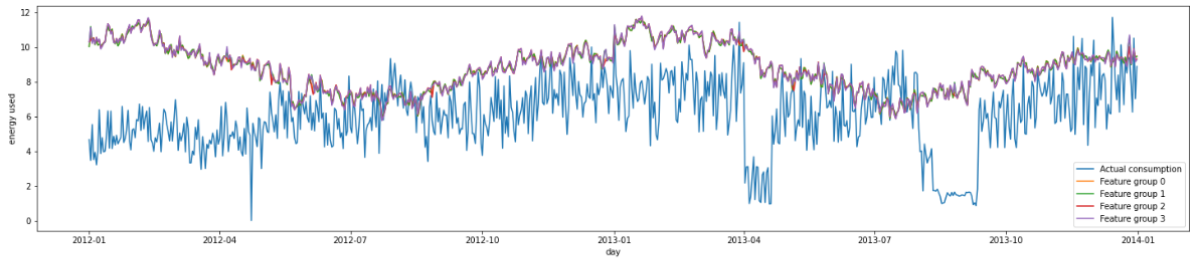
Household 2 - MAC004507 Acorn L




```
-- Test household forecasting performance per feature group: --
```

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.927605518260076	0.6102613527181656	11.59558170007053	3.4052285826461826	-2.5828497425709136
1	2.9329777462634743	0.611745424718592	11.660780329212868	3.414788475032219	-2.602995078758504
2	2.938533423848202	0.6129739541236692	11.716365870180402	3.4229177422457004	-2.620170124072961
3	2.9394594112670904	0.6138551631987923	11.7691286884189	3.430616371502197	-2.636472993099529

Household 3 - MAC004513 Acorn Q




```
-- Test household forecasting performance per feature group: --
```

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.9067856537905405	3.811345360713119	12.167727679117121	3.488227010834748	-2.1492560252281656
1	2.91099413802246	3.803178851939671	12.183087665377842	3.4904280060442217	-2.153231502864965
2	2.9146505161109504	3.750607149862616	12.241872057058883	3.4988386726253733	-2.1684461020549692
3	2.914069929059714	3.7255965484665734	12.221269998551044	3.4958933048007976	-2.1631138692339418

KNN results

Acorn combined dataset results

Pearson's correlation method

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	6.319104236438485	171814987625190.8	86.4213676668882	9.296309357314234	-0.00518222967025217
1	6.30758713009702	172212419423963.88	86.1556033893268	9.282004276519528	-0.0020910782999616906
2	6.2778629424964105	169350359221078.28	85.83656057947948	9.264802241790134	0.0016197651131939494
3	6.305320197735272	170536696150358.03	86.23236380844789	9.286138261325204	-0.0029838923264646766

Mutual Information

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	6.398829834343043	169768603662339.2	89.13451858243116	9.441107910750262	-0.036739368377306736
1	6.39350942395731	1719352733200048.88	88.8836709349551	9.427813687963669	-0.033821714972775974
2	6.323098489822452	168480520363915.9	88.10774937440986	9.386572823688626	-0.02479683391229126
3	6.506002768907045	1744506746285 61.72	90.89431626398034	9.533851072047451	-0.05720788681277589

<Figure size 432x288 with 0 Axes>

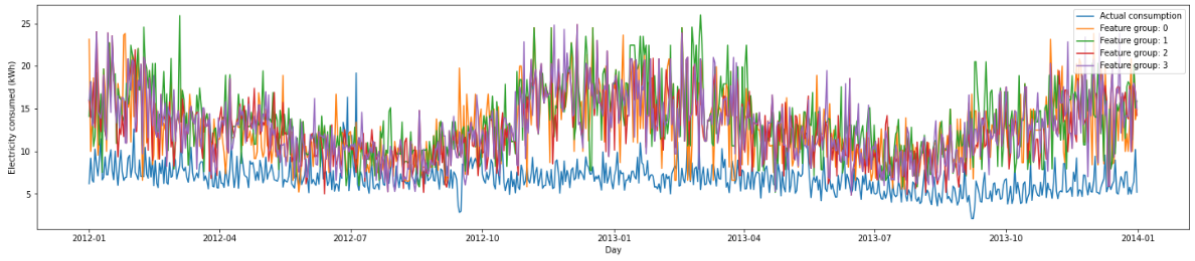
KNN - Affluent households results

KNN - Affluent households Pearson's correlation method

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	8.038129283855318	186671599070059.6	146.18495222639265	12.090696928895069	-0.0532733583919307
1	8.281463297475414	1971428078427 97.84	147.12449168453816	12.129488517020746	-0.06004280945615559
2	7.982385933785563	184389682908430.6	142.79440948401256	11.949661479891912	-0.028844247962647174
3	8.062584148766927	187599871831779.53	144.03960864232377	12.00165024662541	-0.03781599970156857

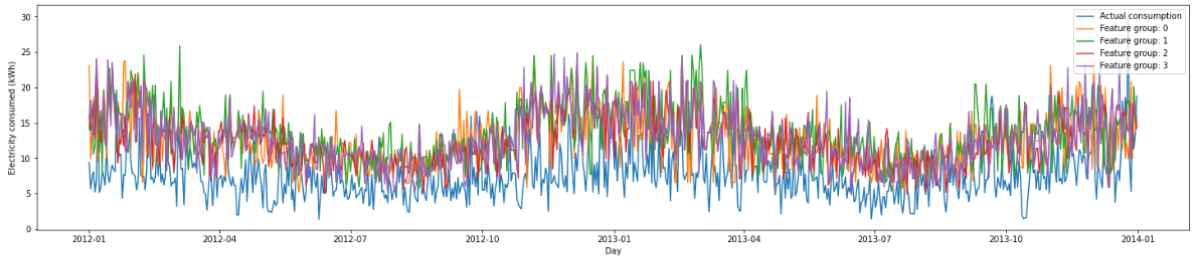
Household 1 - MAC004848 Acorn C



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.997567075216007	0.9710109840565823	50.60592745951741	7.1137843275936765	-15.775089404229568
1	6.692080309549795	1.0692844767344785	60.17237058574758	7.7570851861860834	-18.946218692421432
2	5.884806365138701	0.9428140013160683	48.03139827235819	6.930468834960459	-14.92167243399534
3	6.140543154834016	0.9834589377549232	51.92603877837729	7.205972438080601	-16.212686075392742

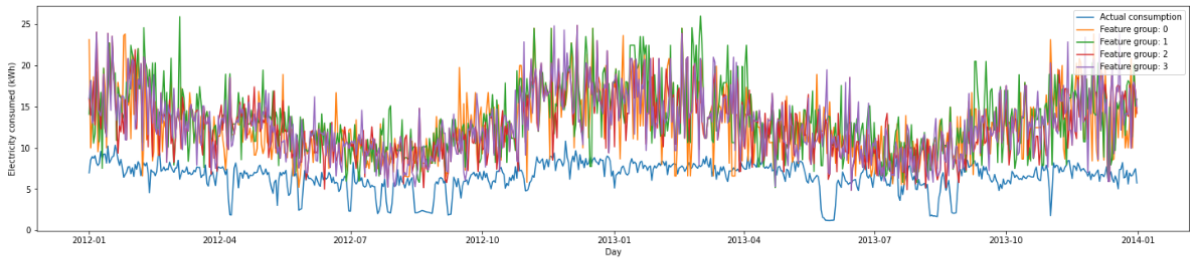
Household 2 - MAC004475 Acorn D



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.657997182355616	1.0146985094092666	46.23321457721505	6.79950105354908	-2.9313441590991003
1	6.203553617107778	1.1216801240379666	54.58986099495339	7.388495177974565	-3.641934010669205
2	5.408364440118236	0.982249709425321	42.46573486856381	6.516573859672259	-2.6109844462265754
3	5.705379448603911	1.0254020529589845	46.53141366898754	6.821393821572505	-2.9567008916648296

Household 3 - MAC004510 Acorn E



-- Household forecasting performance per feature group: --

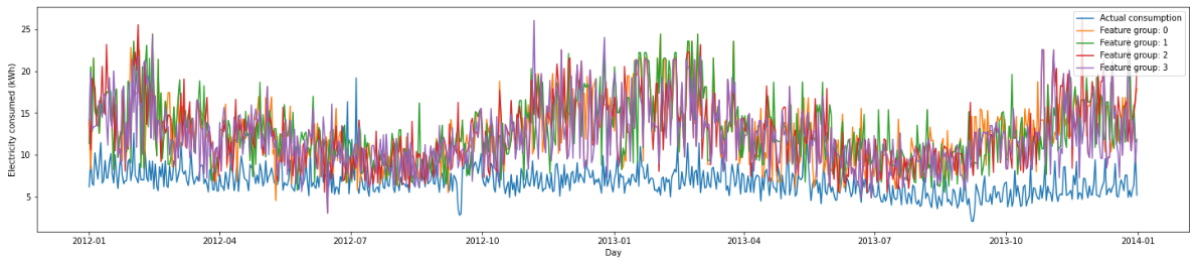
Feature group	MAE	MAPE	MSE	RMSE	R2
0	6.147562892905866	1.1439294867973886	50.45498948370524	7.103167566917258	-16.505115094284363
1	6.841693589504322	1.2537216163328633	60.523255087207296	7.779669343050982	-19.998251253713107
2	6.034824739245112	1.1066070051477375	47.90726127166767	6.92150715319053	-15.621193087025272
3	6.258422556625739	1.1456123620108623	51.382069549505424	7.1681287341610584	-16.826761048817563

KNN - Affluent households Mutual Information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	8.025567374629803	187255474533121.62	142.76474809489332	11.948420317970628	-0.02863053546723382
1	8.131468076002992	188231367330147.78	144.10002640380813	12.004167043314922	-0.03825131412741034
2	8.023924616198196	181250535701706.12	144.8919868557181	12.037108741542468	-0.0439574475713298
3	7.8019986156474195	176958539019120.72	141.46335119543656	11.89383668945545	-0.0192538748601363

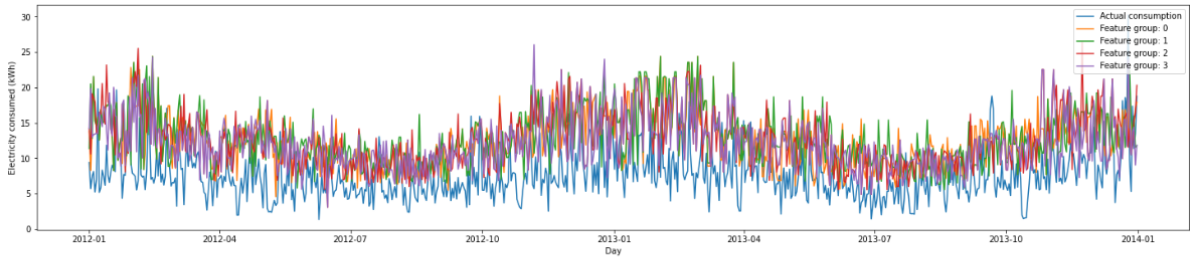
Household 1 - MAC004848 Acorn C



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.891555795916325	0.9480162240650531	47.43624780440002	6.887397752736517	-14.724389174698217
1	6.153533875661664	0.9804778569753665	51.127173380161096	7.15032680233296	-15.947874438697763
2	5.866364526757618	0.9367413014376771	48.55783814232674	6.968345437930495	-15.096179183066145
3	5.511475215434288	0.8804879808628129	42.83150985708618	6.544578661540114	-13.197989113933977

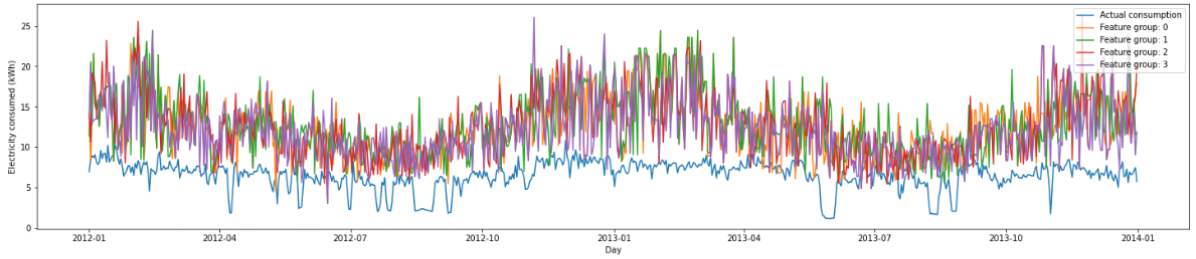
Household 2 - MAC004475 Acorn D



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.408234741982719	0.9685425332066191	41.10028077485015	6.41095006803595	-2.494875929331783
1	5.683864300545703	1.019553275186619	45.078791262323314	6.714074118024265	-2.833180200619689
2	5.422909775034107	0.972138538715003	42.94383415709816	6.553154519549967	-2.6516386136392662
3	5.140310959718054	0.9318393260429567	37.92808280155736	6.15857798534348	-2.225134746767405

Household 3 - MAC004510 Acorn E



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	6.020240472014553	1.0905047178925589	46.97199053561863	6.853611495818728	-15.296705418979244
1	6.239290493551615	1.1549324905970737	51.47733820147955	7.174770951150952	-16.859814047832117
2	6.016149157180537	1.1069156212073639	48.44806263278291	6.960464254112861	-15.808821509235631
3	5.644323055825376	1.0473087126760472	43.31005990927981	6.581037905169656	-14.0262162614689

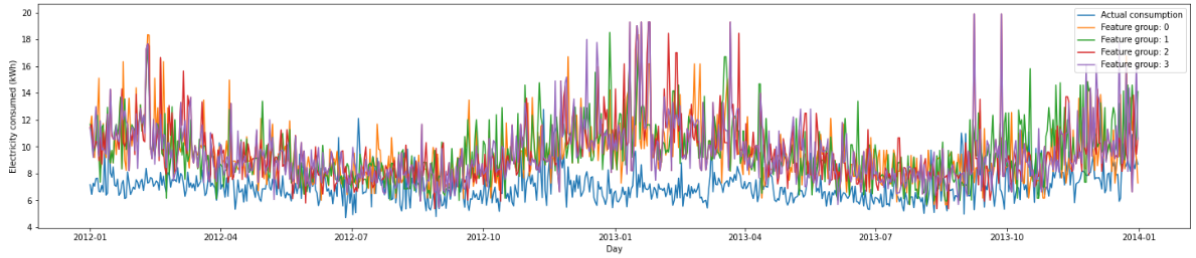
Comfortable households results

KNN - Comfortable households Pearson's correlation results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.164829662654916	93051823196309.83	60.868770804976506	7.801844064384811	-0.005031747959421962
1	5.251032663945549	91114531237934.23	61.555352366375494	7.845721914927618	-0.016368206009156783
2	5.299463584131855	93503821508592.98	63.07752614163627	7.942136119561051	-0.04150150424759036
3	5.2654374814730565	91790770098989.95	62.70257180303233	7.9184955517467035	-0.03531046392677473

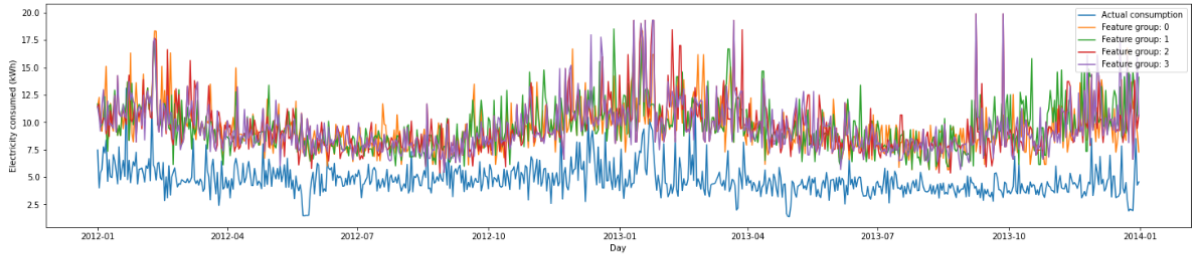
Household 1 - MAC000244 Acorn F



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.5736756710050024	0.37928395453484454	10.485137892764488	3.2380762641983107	-6.6981035233003485
1	2.7323989083128697	0.4023571570463197	11.331632402188502	3.3662490107222465	-7.31959295263329
2	2.7287571623829017	0.40114825951612776	12.03170503492571	3.4686748240395366	-7.83358062315877
3	2.682257935843565	0.39399506810862545	11.743857002111453	3.4269311347197293	-7.62224492321422

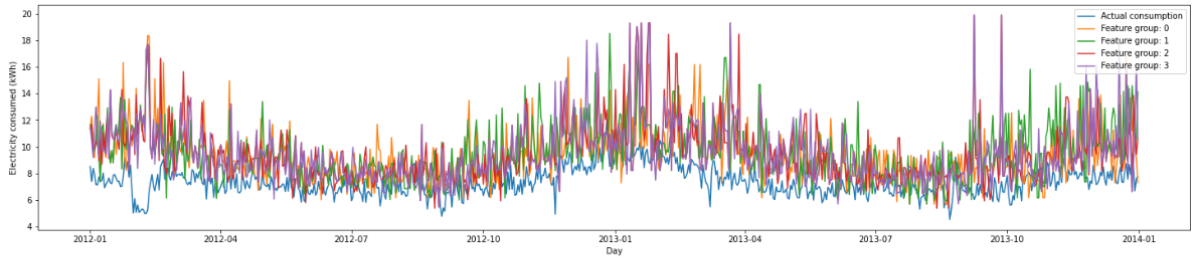
Household 2 - MAC004850 Acorn G



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.656754160181901	1.1289423429292362	26.59259327231317	5.156800681848502	-12.57271277551318
1	4.843821735734425	1.1756910285756093	28.7479506774189	5.361711543660187	-13.67279529429984
2	4.796454478917689	1.1555763591995285	28.64940133127132	5.3525135526471415	-13.622496252165114
3	4.754038199408822	1.1446376286929476	28.09354962077299	5.300334859305871	-13.338792608255124

Household 3 - MAC000239 Acorn H



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.1613424281673486	0.3039465520258308	8.022136178462748	2.832337582009381	-6.1604542456578315
1	2.3302222825557073	0.3268757749318958	8.892499365442744	2.9820294038528097	-6.937329087823555
2	2.2975697136516597	0.32050580463940875	9.25682434315614	3.0425029734013638	-7.262520816738705
3	2.285914324802183	0.3192505556101332	9.199725664020557	3.033104954336489	-7.2115552795872055

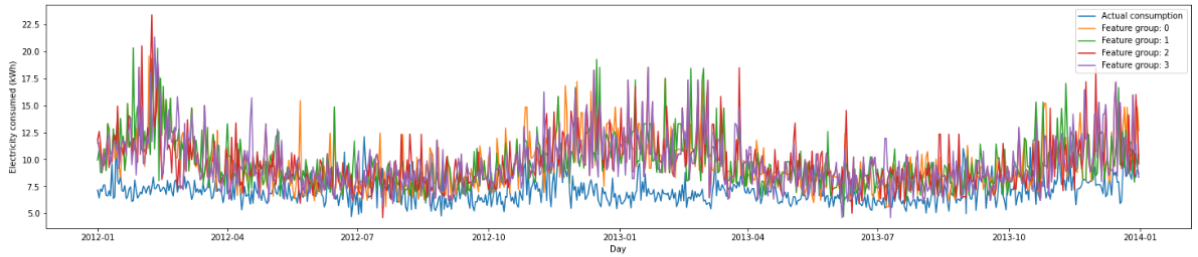
KNN – Comfortable households Mutual information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.1724698965440945	90300982344546.14	61.64949845064549	7.8517194582234975	-0.017922694499495684
1	5.188186564242328	90296313248626.33	61.65164779472688	7.8518563279473526	-0.017958183289732643
2	5.330202728320283	93889960453374.73	63.41465570711008	7.963331947565044	-0.04706800266699407
3	5.28274933727613	92903895593301.3	62.75757887879575	7.92196811902167	-0.03621871058221937

<Figure size 432x288 with 0 Axes>

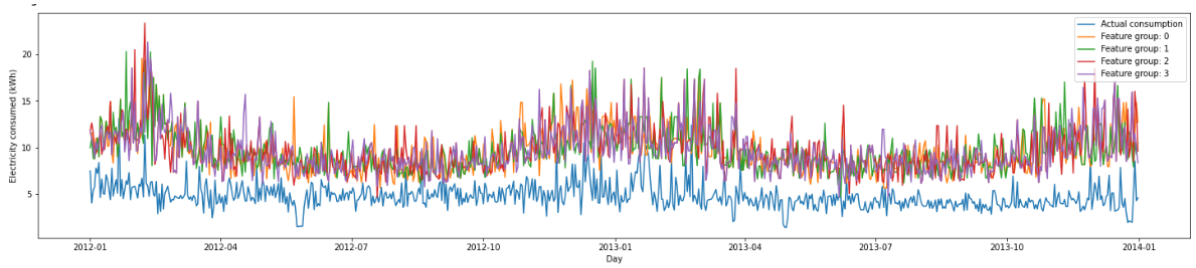
Household 1 - MAC000244 Acorn F



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.8251466126875853	0.41415065858358047	12.554992281477055	3.5433024541347096	-8.217773891532941
1	2.808109139918145	0.4147524175549068	12.68502108487777	3.5616037237286475	-8.313239988386117
2	2.8510008186539335	0.42003566594718666	13.657855370822373	3.6956535783028115	-9.02748706084336
3	2.7952893125511595	0.41094446428801684	12.772830525435479	3.5739096974371747	-8.377708970162962

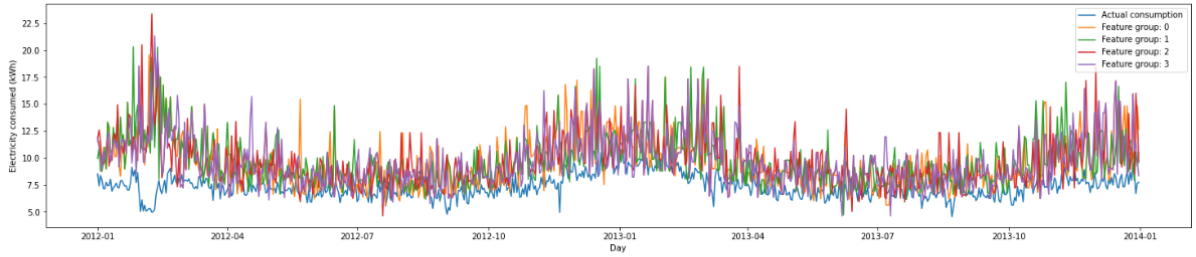
Household 2 - MAC004850 Acorn G



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.868749886202819	1.1608793301129738	29.288874516072088	5.411919670142202	-13.94887983484413
1	4.954088220309232	1.1878579489089178	30.166852937361767	5.492435974807696	-14.396995173323369
2	4.9730881302592085	1.190171756935056	31.01068725363536	5.5687240238348465	-14.82768354912865
3	4.904603545829922	1.1787718355277519	30.275794691541417	5.5023444722719255	-14.452598443136111

Household 3 - MAC000239 Acorn H



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.3746884034106412	0.3318200534009091	9.790847786865507	3.129033043428194	-7.739183185679233
1	2.4186937690950434	0.33848663673777285	10.041695939363697	3.168863509109172	-7.963086978710489
2	2.4747619824192815	0.34745076474754005	11.106224156699136	3.3326002095509653	-8.913270997514168
3	2.4097429734333784	0.33739990275660414	10.35512464620028	3.217937949401803	-8.242849359284927

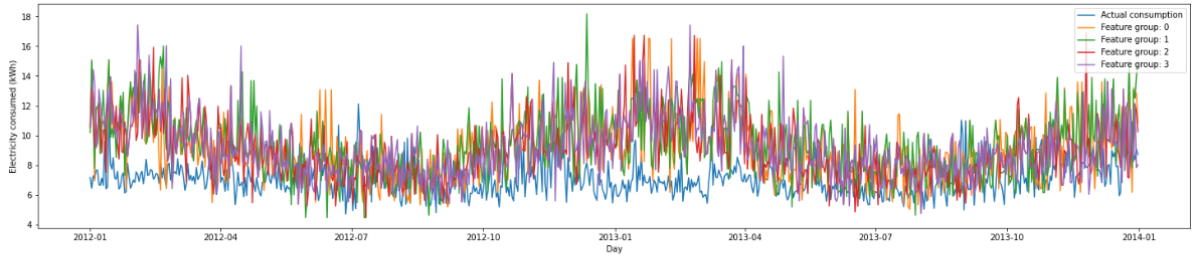
Adversity households results

KNN – Adversity households Pearson's correlation results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.00469569638067	197686561645363.94	45.06959665987096	6.713389357088635	-0.06492709105675942
1	5.063628469565286	203641959213358.66	45.23242432510619	6.72550550703114	-0.06877446500135265
2	4.88828382807417	192046032717366.5	43.913502503103935	6.626726379073149	-0.03761031703180295
3	4.973526425532703	195262501725514.5	44.44950275256747	6.667046028982211	-0.050275200429242606

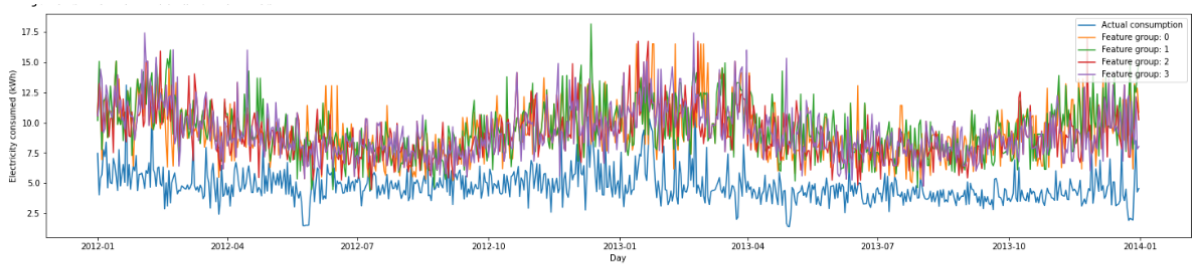
Household 1 - MAC004508 Acorn K



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.513519964283766	0.36803930667537516	9.812039873944611	3.132417576560413	-6.203920396364131
1	2.6944809468303776	0.39406151110202875	10.980479054354253	3.3136805902733375	-7.061779001893839
2	2.3400780350613917	0.34262605076939445	8.620822666055172	2.93612374842328	-5.329338347100122
3	2.5060844013915418	0.3697583296889853	9.791481049630342	3.129134233239338	-6.1888262736632935

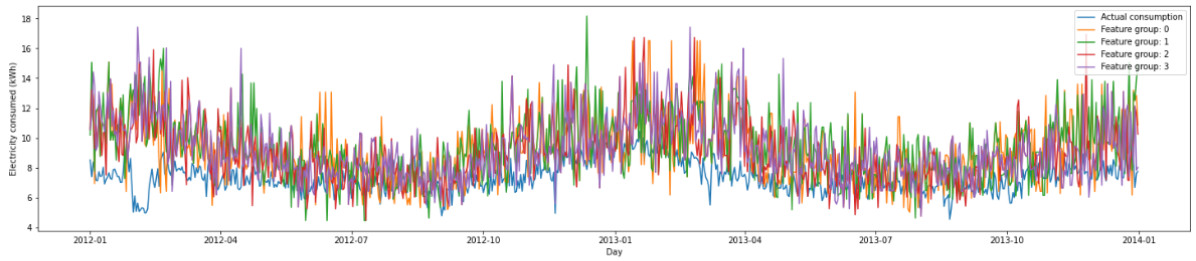
Household 2 - MAC004507 Acorn L



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.488818553615279	1.0842802635722226	25.171777226065007	5.017148316131885	-11.847536110526377
1	4.678311232169167	1.1312932392556365	27.30412150216409	5.225334582796023	-12.935872855334754
2	4.3154024545520695	1.0378413232588874	23.113475527830648	4.807647608532748	-10.796990288634976
3	4.50341809895407	1.0889962363608354	25.174135382012533	5.01738332021907	-11.848739700301465

Household 3 - MAC004513 Acorn Q



-- Household forecasting performance per feature group: --

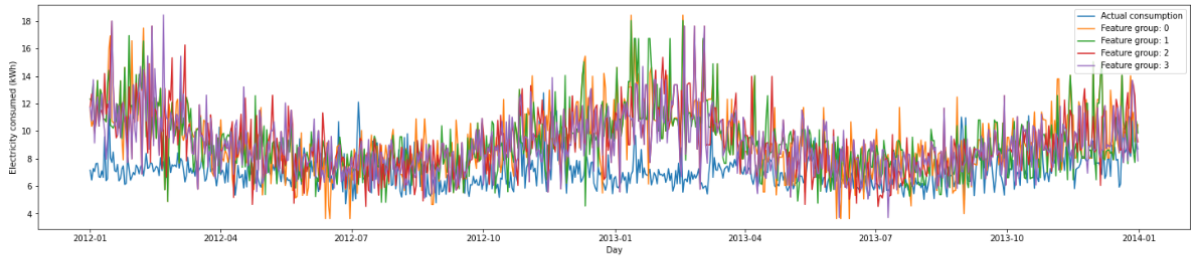
Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.207272943183265	0.3079298564525112	7.829125994048694	2.798057539445659	-5.988175869462571
1	2.3333999097589815	0.32621600874791146	8.725193468858444	2.9538438463904018	-6.7879940240861805
2	1.9346168257116871	0.27057017115071036	6.500596813899866	2.549626798945263	-4.802348030486655
3	2.06076743899045	0.28914052228517273	7.352939157395524	2.71163035043413	-5.563137702522235

KNN - Adversity households Mutual information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.067348611354855	2020440681180 34.06	45.62772289192535	6.754829597549101	-0.0781147782958902
1	4.983728315715052	195820574346559.94	45.17140236319393	6.720967368109588	-0.06733260740319813
2	4.955241374937352	195738746002455.97	44.582285456314885	6.67699673927694	-0.05341265691707253
3	4.872508072277596	191806451677126.84	43.95329562628619	6.629728171372201	-0.038550568954498265

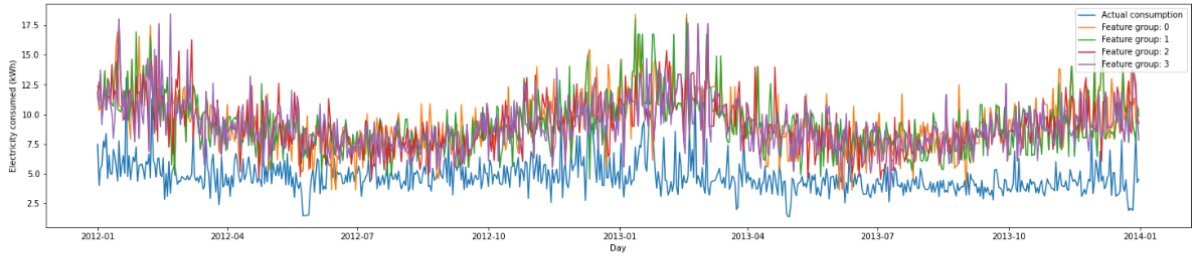
Household 1 - MAC004508 Acorn K



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.546672031723511	0.37441917025735005	10.349015451882567	3.216988568814407	-6.598163527043698
1	2.4581849017280586	0.3630248579725939	9.934904563695016	3.151968363371532	-6.294126658859686
2	2.475758070904957	0.36485394008446626	9.838498250175501	3.1366380489587096	-6.223345922415093
3	2.340825283392451	0.3457270613623428	8.999004533538928	2.9998340843351534	-5.6069964185818515

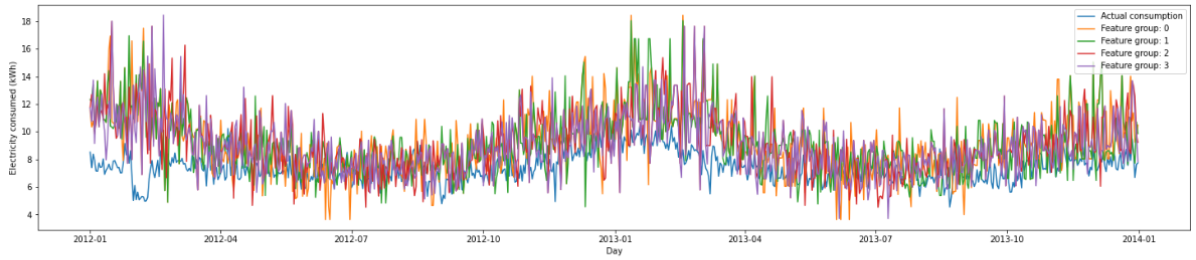
Household 2 - MAC004507 Acorn L



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.529306502328331	1.0842212768206287	25.848342485825253	5.0841265214218705	-12.192851283461277
1	4.4235729872487495	1.0629798785466131	24.56708632503888	4.956519577792353	-11.538905213435987
2	4.422951340854934	1.0713899559338642	24.858072582379194	4.985787057464368	-11.687422992505455
3	4.278377989758981	1.0362663736887479	23.074837048521843	4.803627488525921	-10.77726942213765

Household 3 - MAC004513 Acorn Q



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.225771986412006	0.31054675517761265	8.202753949236401	2.864045032683041	-6.321671314878076
1	2.0461296033196907	0.28565523666119846	7.2757667395639185	2.697362923220366	-5.49425460771844
2	2.142289767567076	0.2984824471781434	7.481353468573991	2.735206293604559	-5.67775864378096
3	1.944048384820373	0.27320944883479625	6.669652866317118	2.5825671078051617	-4.9532452605206165

SVR results

Acorn combined dataset results

Pearson's correlation method

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.488749286126235	127524473332315.11	85.679127068529	9.256302019085645	0.0034509021552736607
1	5.48562537313994	126420291823510.05	85.9005131982252	9.268252974440502	0.0008759208806906615
2	5.492676439429185	129370737646703.0	85.08158275514226	9.223967842265186	0.010401045869407377
3	5.487449544171924	127747842895529.2	85.4550800096963	9.244191690445211	0.00605683316780492

Mutual Information

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	5.509168838409498	127782488195654.12	86.21906363834039	9.2854221034016 75	-0.002829195692928721
1	5.510794961487411	128444712334564.45	86.037171973859	9.275622457488177	-0.0007135815364092135
2	5.50637504088172	126982991236526.86	86.46186696876866	9.29848734842225	-0.005653284221497135
3	5.509118203376109	128210327472134.56	86.15023686567204	9.281715189859687	-0.0020286593131013664

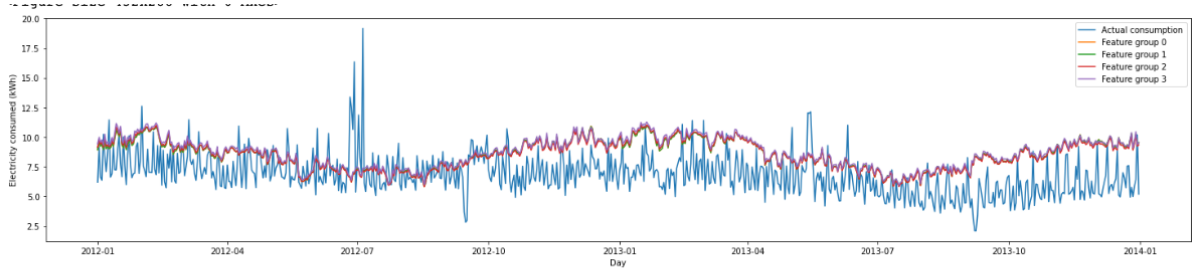
Affluent households results

SVR – Affluent households dataset Pearson's correlation method

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	6.826454078495283	128002450106187.12	142.06851293666912	11.919249680104413	-0.023614109821742835
1	6.8267739731573185	128156179071659.11	142.08170935392783	11.919803243087859	-0.0237091909810081
2	6.822430613738264	127302755777822.62	142.24041858914205	11.92645876147409	-0.024852702721690045
3	6.829208569694998	130164167491728.98	141.19906990386332	11.882721485579948	-0.017349708670004738

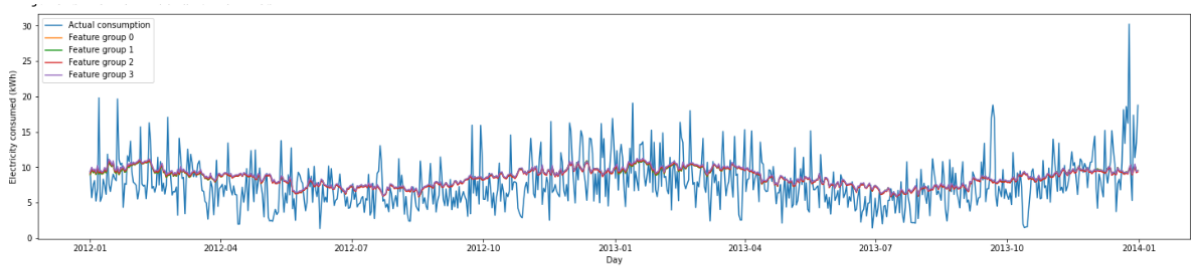
Household 1 - MAC004848 Acorn C



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.1205304790371424	0.3582454169236094	6.43433412125324	2.536598927945299	-1.1328831534023882
1	2.12562590768381	0.3593698231011356	6.454187625559489	2.5405093240449816	-1.1394642858199493
2	2.1022922262135113	0.35447139755520285	6.328104092793098	2.5155723191339776	-1.0976695269698271
3	2.236292223043755	0.37782169182344694	7.030679730185936	2.651542896161768	-1.3305625836167292

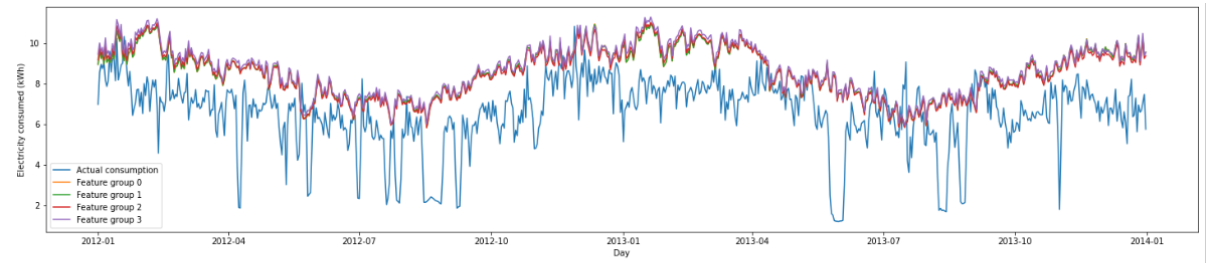
Household 2 - MAC004475 Acorn D



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.6181751493226053	0.4727757415216844	10.817455402519862	3.2889900277318964	0.08016043223674174
1	2.6219834369803814	0.4740640497874258	10.830175034272306	3.2909231279797933	0.0790788451039165
2	2.6078521265236736	0.46955117350726117	10.774551199819038	3.2824611497806093	0.08380870087285353
3	2.6674821917279847	0.4876836033772258	11.0522191408874	3.324487801284192	0.060197791524011834

Household 3 - MAC004510 Acorn E



-- Household forecasting performance per feature group: --

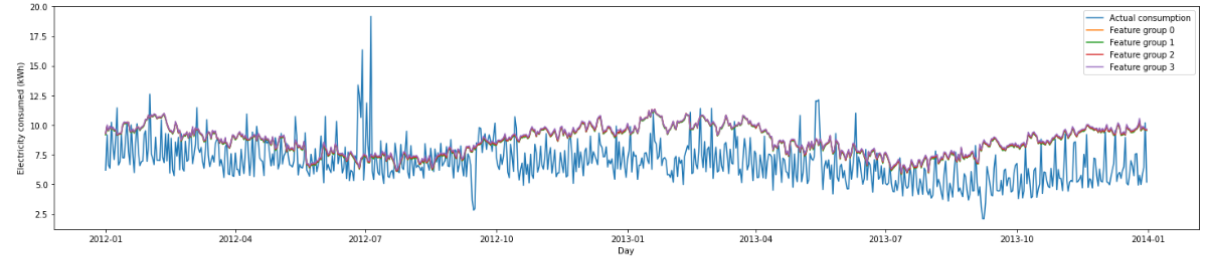
Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.0559297868485022	0.46647441549648094	6.00283503692567	2.4500683739287092	-1.0826546450341232
1	2.0634409485735414	0.4684918078735606	6.040137311365413	2.4576690809312414	-1.0955964891218555
2	2.0314639823470184	0.46123403988397876	5.889456184564569	2.426820179692877	-1.0433183993991912
3	2.2004471291428147	0.49184375880163567	6.6771846979332725	2.584024902730868	-1.3166170053581125

SVR – Affluent households dataset Mutual Information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	6.824762957541388	128423839884658.02	142.16645813752288	11.923357670451846	-0.024319812214981473
1	6.823122519428956	127458156203731.77	142.48656930765944	11.93677382325976	-0.026626236796263525
2	6.824933832404007	128474505620891.95	142.12428757502641	11.921589137989383	-0.024015970203136394
3	6.8287358309602935	129707859797854.47	141.78041184499745	11.907158008735646	-0.021538320215768625

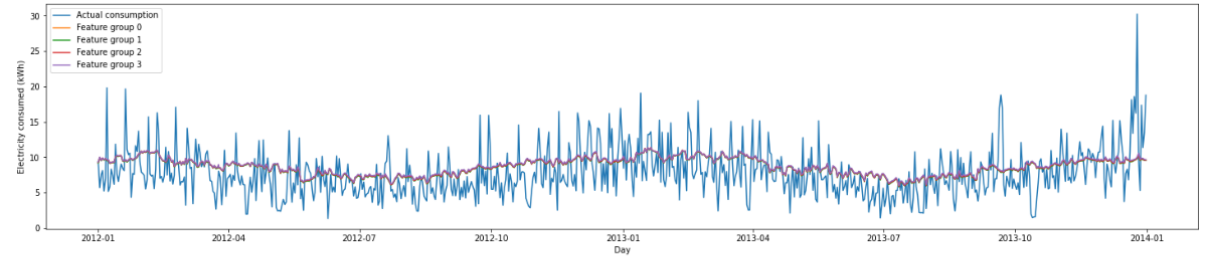
Household 1 - MAC004848 Acorn C



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.216174750666931	0.3756850976631284	6.912725025182741	2.6292061587450197	-1.2914624065937712
1	2.1718591767083826	0.36785325762177046	6.700557846552381	2.588543576328662	-1.221132238393325
2	2.221162375609456	0.37648250695052876	6.949983152381488	2.636282069957896	-1.3038129047699605
3	2.275324132394206	0.38621624928903037	7.228798029881146	2.688642413910996	-1.3962357637527858

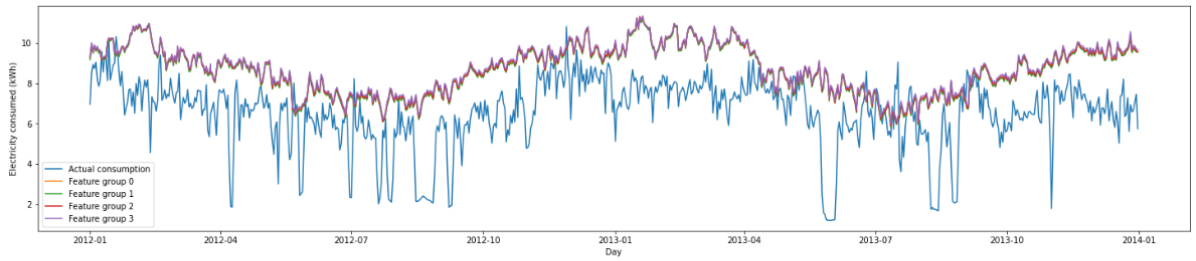
Household 2 - MAC004475 Acorn D



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.688247444335503	0.4901600126227437	11.144255033584486	3.338301219720067	0.05237171296791521
1	2.6605031759256645	0.4828107649980297	11.002141300691584	3.316947587872257	0.06445605533617482
2	2.679466679314386	0.4892867474743656	11.095233604148588	3.3309508558591174	0.05654014711306077
3	2.7087443275784993	0.4975363665472946	11.24485646588858	3.3533351257947035	0.043817281767264515

Household 3 - MAC004510 Acorn E



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.1827580404150027	0.49071678462613366	6.595248455394833	2.568121581116212	-1.2881896214520707
1	2.1259489752159	0.4797905734822124	6.328037444535259	2.515559071962982	-1.1954820508545674
2	2.1872954069916153	0.49126690406552476	6.618057550312315	2.5725585611045503	-1.2961031268518872
3	2.2552446904429737	0.5044054882030968	6.9446888388253	2.63527775363913	-1.4094262760056648

Comfortable households results

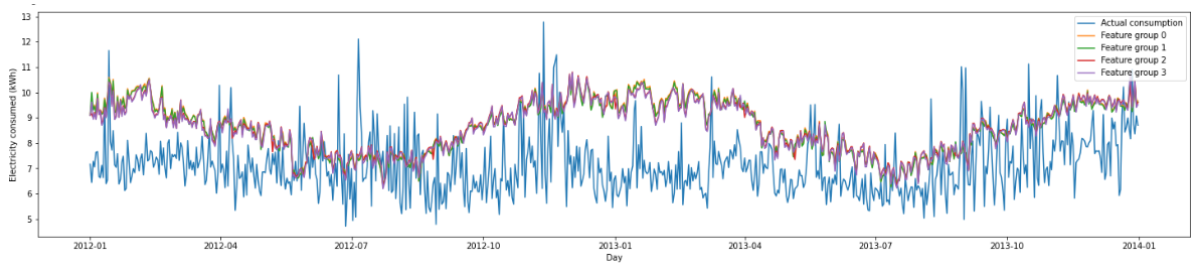
SVR - Comfortable households dataset Pearson's correlation results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.929165038598086	84021881058775.19	59.90844148367514	7.740054359219652	0.010824682910466454
1	4.928161430685689	83699156027069.66	60.011421840279155	7.746703933950178	0.009124327762260354
2	4.928665898855345	84187991577694.23	60.033626305775435	7.7481369570868734	0.008757699810437058
3	4.9269133754715275	83720543464483.97	60.11797814243685	7.753578408866248	0.007364928563009965

<Figure size 432x288 with 0 Axes>

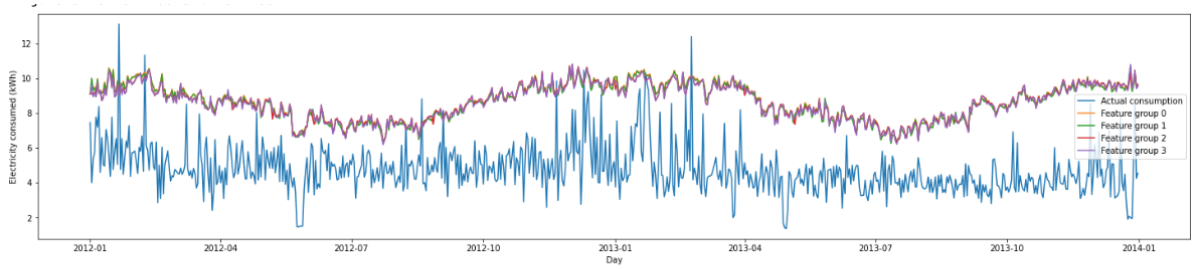
Household 1 - MAC000244 Acorn F



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.809514856513005	0.2692062541080956	4.298938973943483	2.073388283448974	-2.1562462602046057
1	1.7841725437843585	0.26532497550042783	4.192591503983246	2.047581867467879	-2.078166807023554
2	1.7840451061712204	0.2658747232039158	4.161971776101681	2.040091119558556	-2.055686050213456
3	1.7605074108108436	0.262093486980087	4.081714830136745	2.020325426790631	-1.9967620489441607

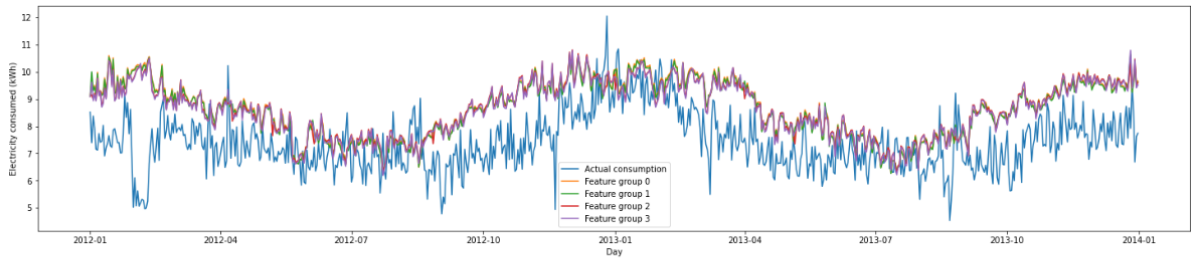
Household 2 - MAC004850 Acorn G



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.9189168339676264	0.9583141414296393	17.304620862766832	4.159882313571723	-7.832182941105728
1	3.886855428801235	0.9512675603858525	17.050716821453232	4.129251363316749	-7.702591720347295
2	3.901328835107692	0.9553083609591388	17.134719221919845	4.139410492077325	-7.745466081727237
3	3.8659022271132715	0.9470762957956634	16.873287186048014	4.107710698923187	-7.612032614112012

Household 3 - MAC000239 Acorn H



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.3818172052972764	0.19864262600623012	2.6803722063478377	1.6371842310344422	-1.3924653132184104
1	1.3517626627836037	0.194475910104846	2.580012044029397	1.6062415895591164	-1.302885139760 71
2	1.366648193313244	0.1968676837679624	2.6184848450113356	1.6181733049989842	-1.3372254607182463
3	1.3360210174601295	0.19253099956553915	2.5272575083566524	1.5897350434448667	-1.255797128471428

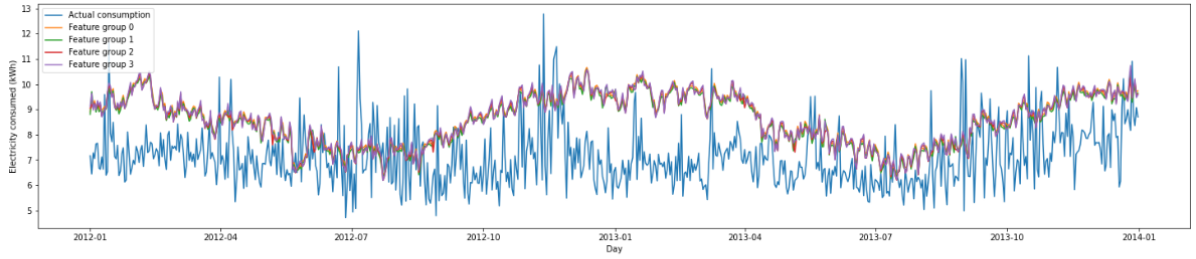
SVR – Comfortable households dataset Mutual information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.930136462761355	83944479280591.0	59.99376387716871	7.745564141956912	0.009415886361121 45
1	4.92797647599053	83122568112097.8	60.206591299774445	7.759290644110094	0.005901796726545561
2	4.927107029324339	83613565912610.12	60.05999338231936	7.749838281042989	0.008322341108806142
3	4.9276935444287	84012606713318.0	59.974344077756115	7.744310432682571	0.009736535434374471

<Figure size 432x288 with 0 Axes>

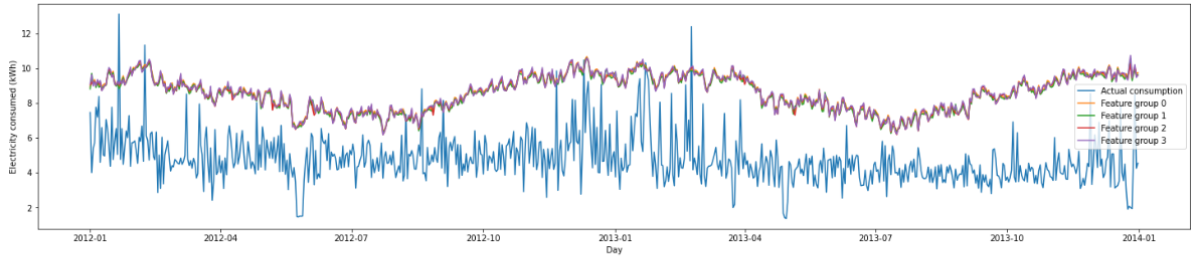
Household 1 - MAC000244 Acorn F



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.7719826320342629	0.2637725729234473	4.09696118988258	2.0240951533667038	-2.0079558030823783
1	1.7152250986455604	0.25499828516333845	3.8738939371539423	1.968221008208667	-1.8441816284624863
2	1.748446805692065	0.26018620481280497	4.020823170044904	2.0051990350199413	-1.9520558840957154
3	1.7782274601566783	0.26475897752276517	4.145126125564173	2.0359582818820656	-2.043318109697904

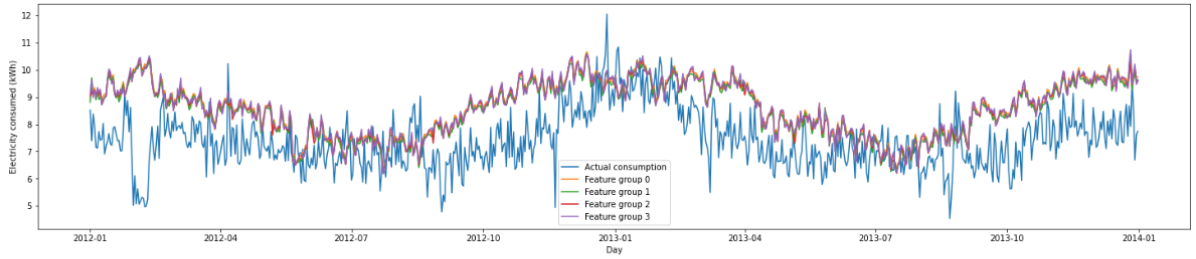
Household 2 - MAC004850 Acorn G



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.880415110144245	0.950744550696083	16.982630928817688	4.120998778065542	-7.66784105668153
1	3.805828214047888	0.933707774881932	16.401996072131013	4.049937786204994	-7.371488231796842
2	3.8467175900706554	0.9428377226852107	16.73183450128998	4.090456515022495	-7.539835944840766
3	3.883766950826001	0.9514780069257776	17.026786466376436	4.126352683227215	-7.690377799247258

Household 3 - MAC000239 Acorn H



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.3497646653082462	0.19439608583580545	2.5611626587817358	1.600363289625745	-1.2860604240463331
1	1.288022849305704	0.18564429695404117	2.370771871275675	1.539731103561812	-1.1161200874073267
2	1.3282345747503188	0.19127409254376954	2.500395908910368	1.5812640225181778	-1.231820814741341
3	1.3546566185106808	0.1951738356536412	2.5836398296891843	1.6073704705789467	-1.3061232539801746

Adversity households results

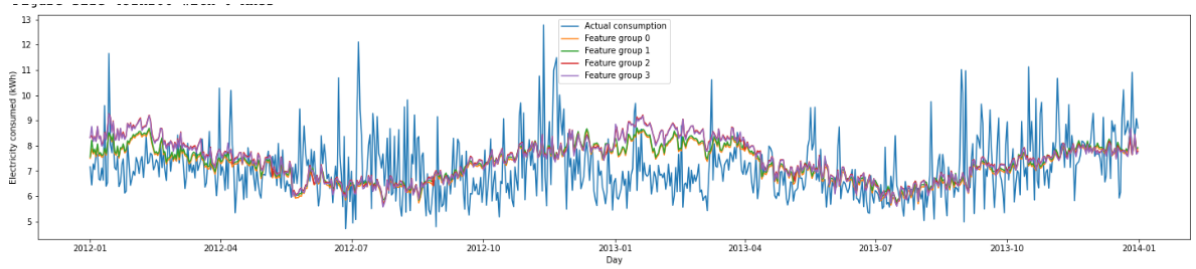
SVR – Adversity households dataset Pearson's correlation results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.421555023446005	155066947727877.8	42.56641160721894	6.5242939546911085	-0.005780576021026329
1	4.426997559315053	156739314589281.47	42.38340381298365	6.5102537441319175	-0.001456375841768498
2	4.433080383623846	158543046161620.03	42.028133812208594	6.482910905774395	0.006938122367752264
3	4.429865272201193	157818902496573.4	42.09073231714461	6.4877370721342125	0.005459013418340319

<Figure size 432x288 with 0 Axes>

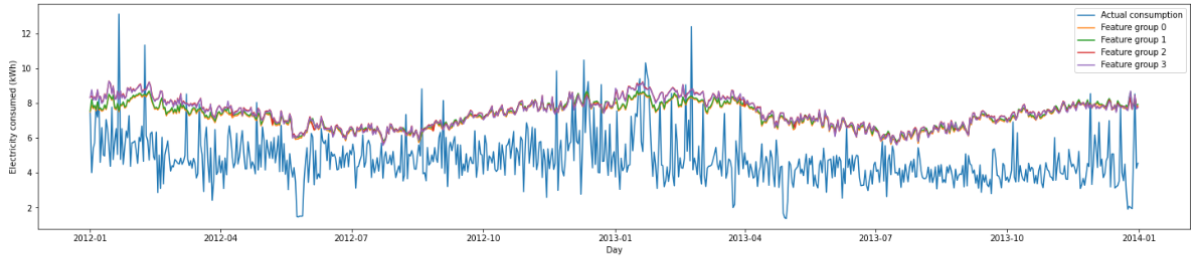
Household 1 - MAC004508 Acorn K



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	0.9495906739597819	0.13296822189189766	1.5405536193457579	1.2411904041466635	-0.1310620199952255
1	0.9712565594614365	0.13713910347087943	1.5656367885856912	1.251254086341256	-0.14947788018479136
2	1.0677395852060607	0.1520251306644784	1.8198071938642268	1.3490022957223708	-0.33608773810030734
3	1.0592587802833442	0.15041621035337707	1.8101403974796852	1.3454145820079717	-0.3289904543003279

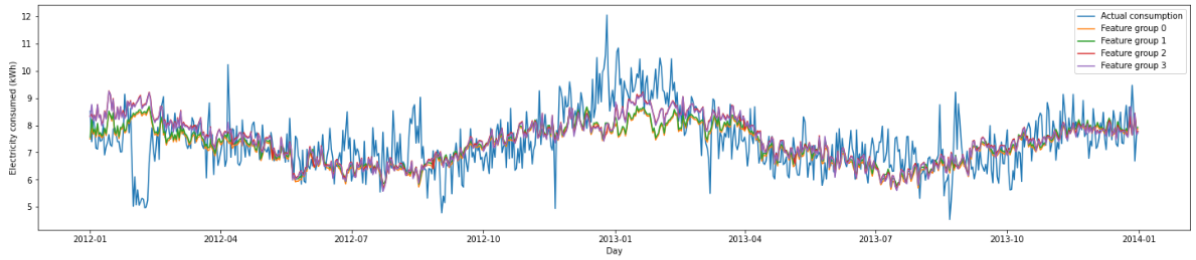
Household 2 - MAC004507 Acorn L



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.5780334304762182	0.6495756379335603	8.013188738411943	2.8307576262216343	-3.0898872873627603
1	2.6535274205327717	0.6677739641048936	8.423658403434883	2.9023539417918833	-3.2993887379872815
2	2.769480923335798	0.6933670848754865	9.143650610187287	3.023846988554032	-3.6668687848856276
3	2.74189283935733	0.6869768477708207	8.993058529681283	2.9988428651200256	-3.5900074184876765

Household 3 - MAC004513 Acorn Q



-- Household forecasting performance per feature group: --

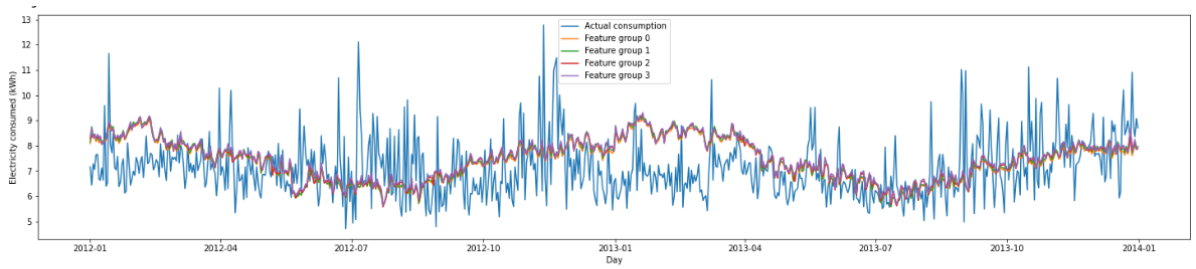
Feature group	MAE	MAPE	MSE	RMSE	R2
0	0.7131151358609001	0.09692935684631457	0.9385680443991805	0.968797215313494	0.16224639063099044
1	0.7030667659125445	0.09643000809075007	0.907065285839152	0.9523997510705009	0.19036534252396553
2	0.7232440595046299	0.10102921169893643	0.9688306907826738	0.9842919743565289	0.13523434671140555
3	0.7197518124795372	0.10028488952363103	0.9573078574467416	0.9784211043547362	0.1455194776349864

SVR - Adversity households dataset Mutual information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	4.432650408550924	158178306828732.56	42.0628014580542	6.485584126202836	0.006118977800678382
1	4.438507474453987	159581608271865.47	41.93190375158547	6.475484827531099	0.009211894625023143
2	4.437828273752207	159911941309685.12	41.900922664001314	6.4730922026494655	0.00994393133987892
3	4.441406986607315	160741391320152.25	41.83022549769149	6.467629047625682	0.011614399532271724

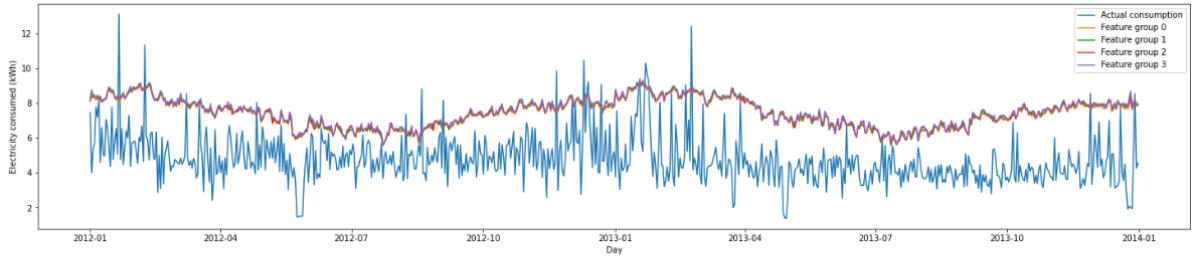
Household 1 - MAC004508 Acorn K



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.0476960321951718	0.14805595539126326	1.7675930656441623	1.3295085805079117	-0.29775254703960496
1	1.075187684662886	0.1527020834076334	1.828685461948429	1.3522889713180497	-0.3426060907933095
2	1.0742973168150747	0.15302290671016458	1.8284635292814027	1.3522069106765437	-0.342443149622345
3	1.0880237394539114	0.15564702135629704	1.855932639481151	1.3623261868881296	-0.36261074838673824

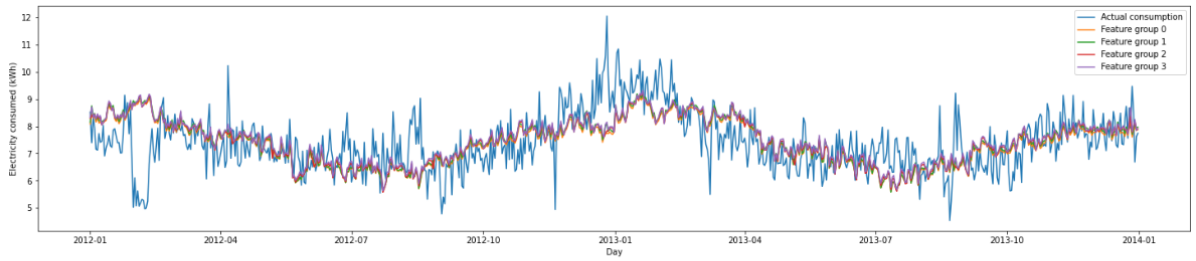
Household 2 - MAC004507 Acorn L



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.7020595081319176	0.678044812503978	8.775530832149682	2.9623522464672702	-3.478982482744043
1	2.759167956874117	0.6914185941341412	9.12819113843274	3.0212896482185783	-3.6589783558613895
2	2.7758292294449767	0.6952926690074128	9.204351579367916	3.0338674294319317	-3.6978502243957685
3	2.8219362084977297	0.7063190395271839	9.458970004588936	3.0755438550911505	-3.8278060627560855

Household 3 - MAC004513 Acorn Q



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	0.7172394361918787	0.09945712750014621	0.959043249142479	0.9793075355282829	0.14397048961493975
1	0.7207009260497659	0.10067996904511672	0.9623611007414481	0.9810000513463025	0.1410090185005396
2	0.7235170642299177	0.10122952272546511	0.9666494319473709	0.9831833155354961	0.1371813099317578
3	0.7246081096595535	0.1018584582394168	0.9608370140793299	0.9802229410084881	0.14236939840032425

LSTM results

Acorn combined dataset results

Pearson's correlation method

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	11.562142425680289	0.9796489653522797	204.15053271026022	14.288125584213635	-3.367406919688632
1	10.860187300409345	0.9444537303840514	168.90070178919993	12.996180276881354	-2.6133047704623724
2	11.09118734760835	0.9294472686757397	176.6551716423234	13.29116893438359	-2.779196697588097
3	11.391436535205505	0.974720605443493	184.48145616294264	13.582395081978092	-2.9466249610220947

Mutual Information

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	9.283179853239016	0.7688679426076721	131.26413795879776	11.457056251882408	-1.808143073728087
1	9.339001685709604	0.761870802936848	137.34297952431902	11.719342111412185	-1.9381881652813266
2	10.498553147986708	0.8940963673210491	158.62375632814627	12.594592344659127	-2.3934493425876417
3	9.793896483115166	0.8072752343376925	148.47348872667217	12.18496978767991	-2.1763039432058777

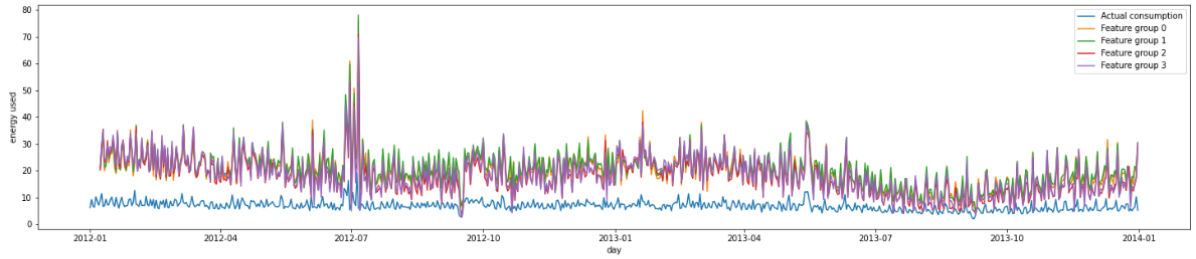
LSTM - Affluent households results

LSTM - Affluent households dataset Pearson's correlation method

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	11.24404652238814	0.9671183712038395	185.45521788511468	13.618194369486531	-2.9674567150571325
1	11.670579239448749	1.0251722484363495	192.06137981063662	13.858621136701755	-3.108782808714972
2	10.136373782604231	0.8335048849361286	160.82748584647658	12.681777708447525	-2.4405938854879414
3	9.846048962946009	0.849026544500718	143.38432569745152	11.97431942523046	-2.067431115228561

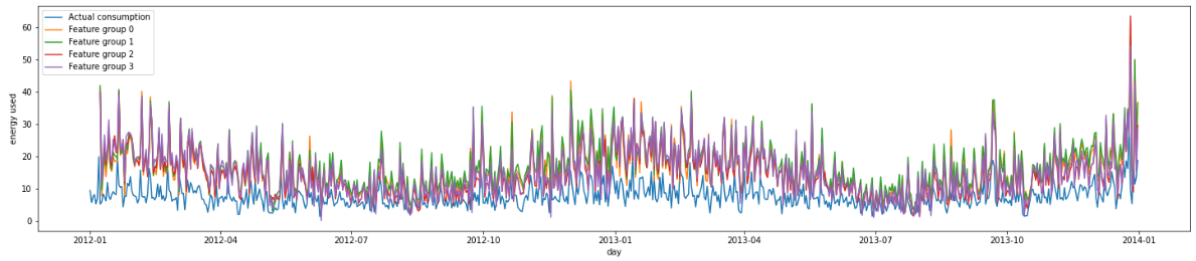
Household 1 - MAC004848 Acorn C



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	13.11054464909937	1.999730871173539	212.55088335324172	14.57912491726584	-69.42620553005216
1	13.73865725553678	2.100948650018881	228.6763569627549	15.122048702565236	-74.7691892936418
2	10.988495179760319	1.6586688390120004	164.05041086276876	12.808216537159604	-53.356151197499344
3	11.771788169777892	1.7803539228931657	184.9103180364293	13.598173334548626	-60.26783317582606

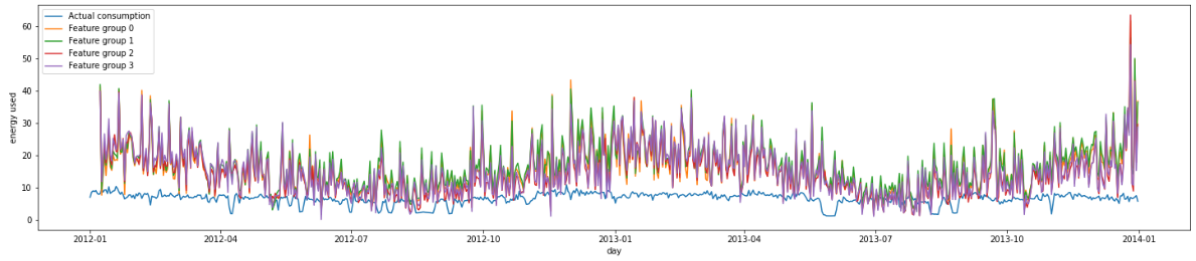
Household 2 - MAC004475 Acorn D



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	8.688141380290547	1.3894417805518575	118.69882284387772	10.894898936836345	-9.187098244479854
1	9.479249190329009	1.5103665012769967	132.31474888372895	11.50281482436925	-10.355658917065245
2	7.457243374979529	1.1765500958085346	94.39000433721544	9.715451833919792	-7.100840635503312
3	8.258757961375023	1.2991739428175983	109.9619465260287	10.486274196588067	-8.437272633177358

Household 3 - MAC004510 Acorn E



-- Household forecasting performance per feature group: --

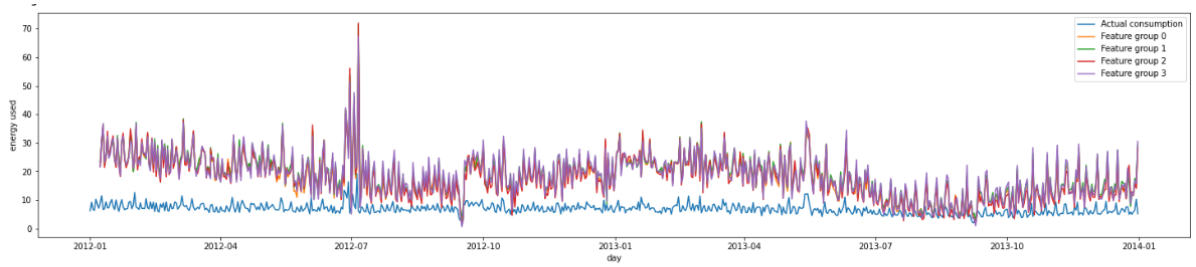
Feature group	MAE	MAPE	MSE	RMSE	R2
0	9.64721505351106	1.710410898022891	142.62106075824286	11.942405986996208	-48.58698762176406
1	10.441978018487452	1.845009840407124	157.89741578996336	12.565723846637859	-53.89832399688506
2	8.257576456427246	1.4379575787562	112.66888256730127	10.614559932814043	-38.17310988657653
3	9.039191631690693	1.5607127848333124	128.7419996687551	11.346453175717738	-43.76146727584025

LSTM – Affluent households dataset Mutual Information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	9.639639410780566	0.7975463606186969	141.12635817932025	11.879661534712184	-2.019126254926704
1	10.747497933648935	0.8958688170374355	172.8470959877975	13.147132614672962	-2.6977302632688764
2	10.15097377921356	0.8140994866948976	162.6822356480001	12.754694651303891	-2.4802726803944504
3	10.372113627695922	0.862860871005146	159.84231594007454	12.642876094468162	-2.4195180753532908

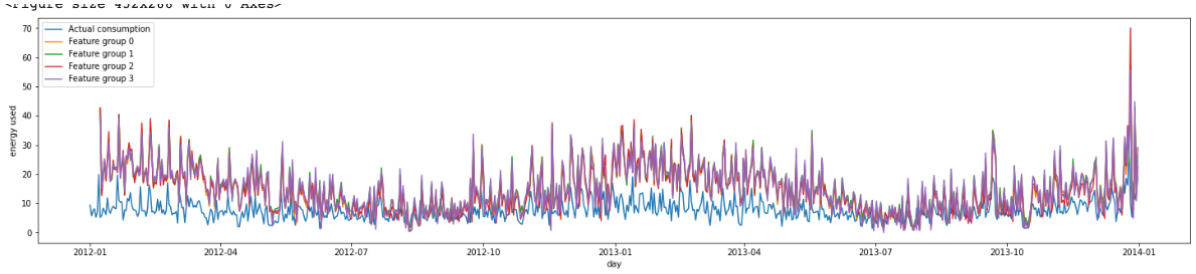
Household 1 - MAC004848 Acorn C



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	11.35628404442713	1.7087485983757016	172.81070123963707	13.145748409262863	-56.258769153496914
1	11.928591643200571	1.799971670571167	187.8374501391811	13.70538033544422	-61.23770356088306
2	10.989928544054438	1.6481557492859973	169.7246898934503	13.027842871843761	-55.23625602202511
3	11.824406146840753	1.784629818582002	186.91766817508312	13.671783650097858	-60.932944753841326

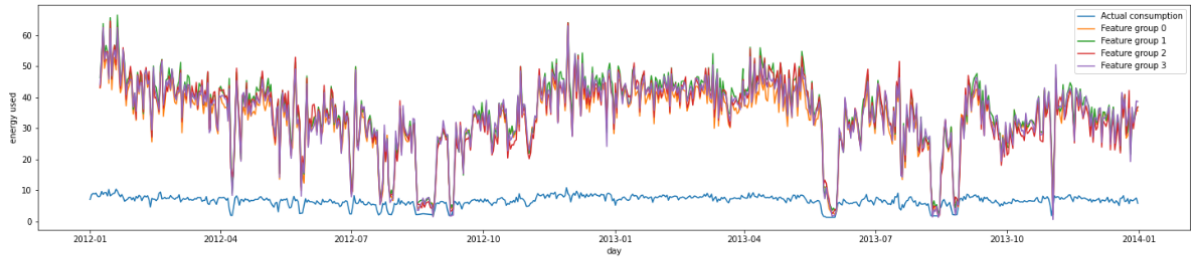
Household 2 - MAC004475 Acorn D



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	7.910601165910353	1.2345660283381696	103.8804460599857	10.192175727487518	-7.9153395487784035
1	8.286208755239265	1.296978260299881	111.34605008533303	10.552063783228997	-8.556060659893621
2	7.689596669962821	1.189155746314082	103.5494109685356	10.175923101543939	-7.886929098546661
3	8.373958568115619	1.3090583266721654	112.56236360570159	10.609541159055919	-8.660448429133597

Household 3 - MAC004510 Acorn E



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	26.63691713213855	4.051864928023457	791.6734711988531	28.136692613007185	-274.2517924639517
1	28.919274430692695	4.3927283401292705	938.9335716391847	30.64202296910543	-325.4516975753882
2	27.794105864612856	4.222818857355611	874.9343359080689	29.579288968940226	-303.2002201769649
3	27.89270400839207	4.214287809323266	873.07024369599	29.547762075933772	-302.5521060979169

Comfortable households results

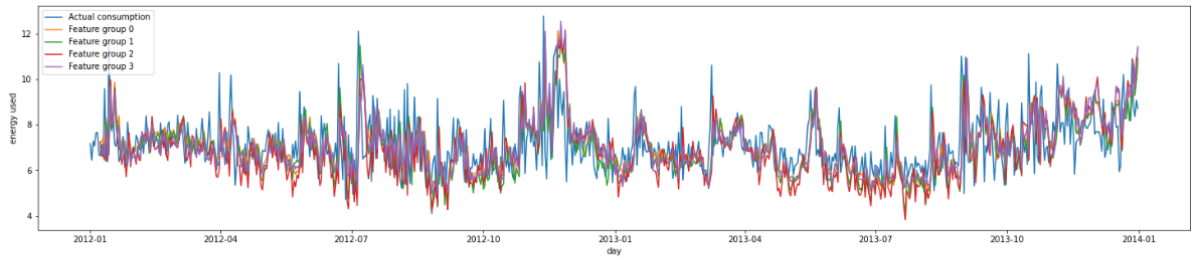
-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.1125540951712525	0.16756063401677515	2.280096670850565	1.5099988976322352	0.6793760555997126
1	1.0738187587558954	0.15831002654234863	2.195898725498129	1.481856513127411	0.6912158506813955
2	1.1362994898683063	0.1664217731423558	2.3453175690702035	1.5314429695781047	0.6702047858409135
3	1.14065198594057	0.1704832604120561	2.361496163582071	1.5367160321874926	0.667929774937401

LSTM - Comfortable households dataset Pearson's correlation results

Feature group	MAE	MAPE	MSE	RMSE	R2
0	0.9936971068773249	0.13969554239915524	2.1360192869093173	1.461512670800126	0.6996360119992301
1	1.0242466531171508	0.14756046791813088	2.1563939995731958	1.4684665469710898	0.6967709489412334
2	1.0475064699990417	0.1533789788207529	2.163495653528008	1.470882610383306	0.6957723244829519
3	1.0490535373013448	0.15319107959372258	2.177105066866463	1.4755016322818701	0.6938585881746042

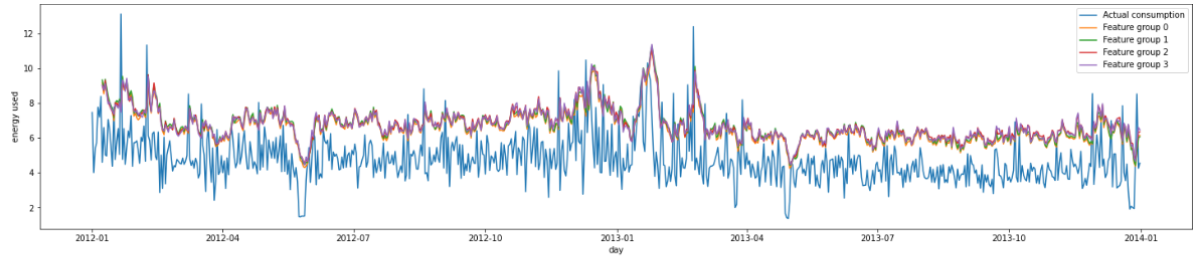
Household 1 - MAC000244 Acorn F



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	0.9467704712835054	0.1312643281581903	1.6844665256348195	1.2978699956601276	-0.22642098775994235
1	1.037858810145943	0.14359881555703685	1.9699117593087647	1.4035354499651103	-0.4342470383855763
2	1.1041756252489956	0.15309023883764963	2.148243594350337	1.4656887781348185	-0.5640863091293531
3	0.9568414647227306	0.13199432500759625	1.7609359962173465	1.327002636100376	-0.2820966347485099

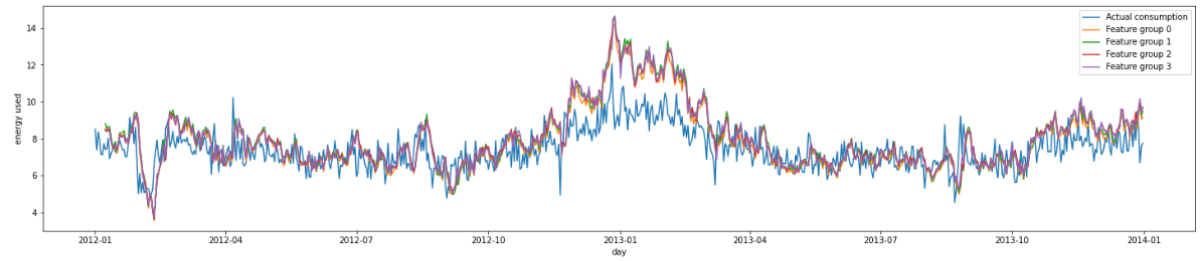
Household 2 - MAC004850 Acorn G



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.0675344828208226	0.5099212370443306	5.330215023969682	2.3087258442633853	-1.7664217707962546
1	2.170067310014238	0.534436367028424	5.8129644701154595	2.4110090149386543	-2.016972371785501
2	2.211106805540176	0.545872497892108	5.984726998971795	2.4463701680186904	-2.106118418820824
3	2.1861168754739806	0.539607097812619	5.8641899510837705	2.421608959159957	-2.0435587824899426

Household 3 - MAC000239 Acorn H



-- Household forecasting performance per feature group: --

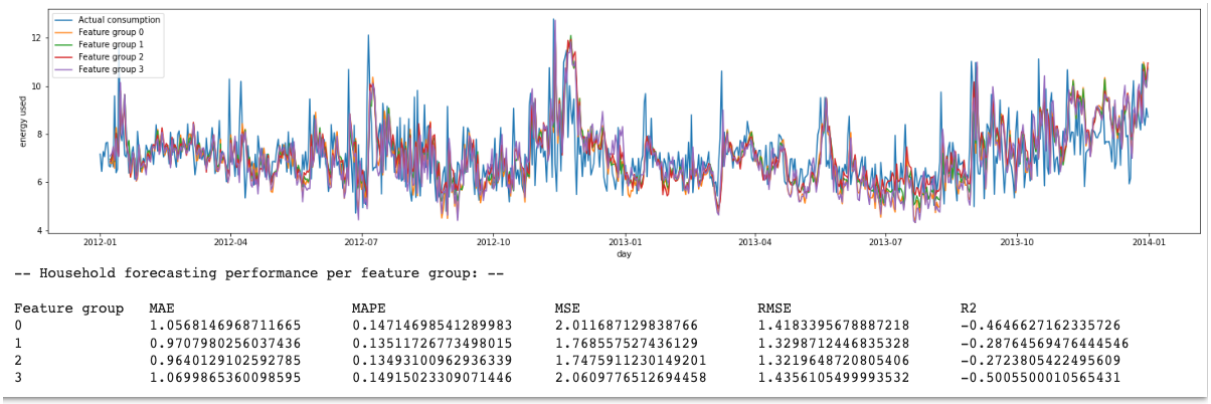
Feature group	MAE	MAPE	MSE	RMSE	R2
0	0.9045231682662165	0.1202456957399543	1.4541729980001958	1.2058909560985172	-0.28982112824117867
1	0.9942027004952042	0.1321501170915875	1.763287720121426	1.3278884441553915	-0.563999441406577
2	0.9803069430774992	0.13057311634492547	1.7049644814683476	1.3057428849005257	-0.5122679448201006
3	0.9663217299978166	0.1283302406111144	1.6823804506349225	1.297066093394983	-0.49223638037084916

LSTM – Comfortable households dataset Mutual information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.136724532065754	0.16569274130816072	2.409055491868927	1.5521132342290387	0.6612420499723575
1	1.1637752063074693	0.17424764414485133	2.4420838863644585	1.5627168285919424	0.6565976442084493
2	1.2445270571552516	0.1914376284721193	2.593314620438088	1.6103771671375895	0.6353317939078347
3	1.2103611650784691	0.17980329911270299	2.6220864767948764	1.619285792192001	0.6312859364708381

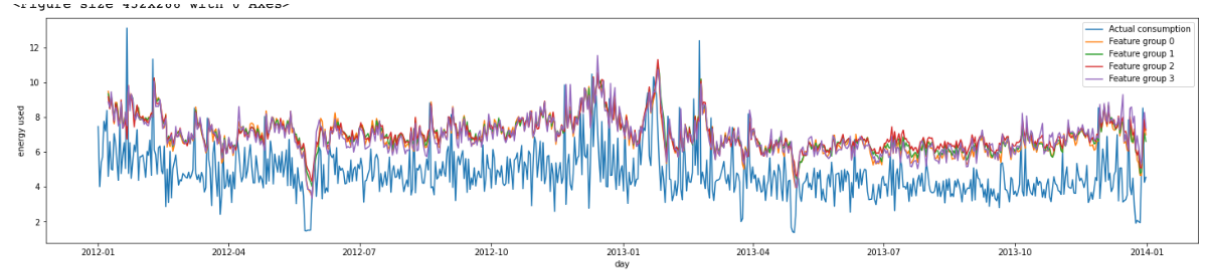
Household 1 - MAC000244 Acorn F



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.0568146968711665	0.14714698541289983	2.011687129838766	1.4183395678887218	-0.4646627162335726
1	0.9707980256037436	0.13511726773498015	1.768557527436129	1.3298712446835328	-0.28764569476444546
2	0.9640129102592785	0.13493100962936339	1.7475911230149201	1.3219648720805406	-0.2723805422495609
3	1.0699865360098595	0.14915023309071446	2.0609776512694458	1.4356105499993532	-0.5005500010565431

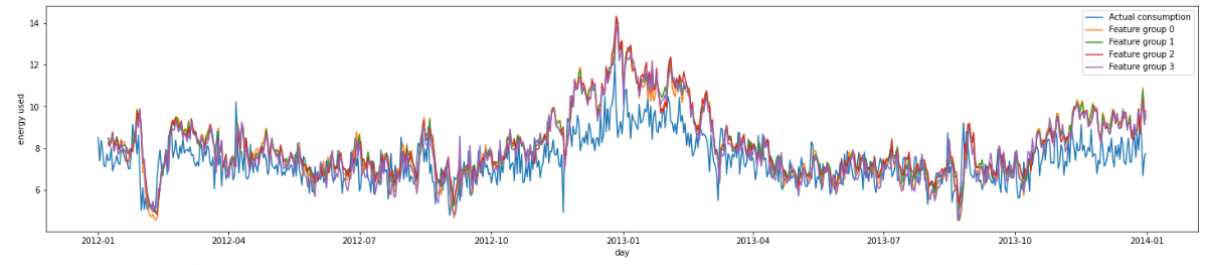
Household 2 - MAC004850 Acorn G



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.325058365864256	0.568229550906276	6.6329664029171305	2.575454601214537	-2.442559555191116
1	2.394065482715701	0.588601619923852	6.904457110348117	2.6276333668052163	-2.583465278549084
2	2.437312445712255	0.6016543989400972	7.1110087903488255	2.6666474814547247	-2.6906671572310707
3	2.3323512662233084	0.5746841982708308	6.799945224552574	2.607670459347303	-2.5292228221244204

Household 3 - MAC000239 Acorn H



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	1.024590985128515	0.13851024513529953	1.7068112112878944	1.306449850276655	-0.513905955663845
1	1.005625129201665	0.13603501407770882	1.6541788674709061	1.2861488512108177	-0.467222164136039
2	1.0005766078527865	0.1354411299408932	1.6780761247354212	1.295405776093121	-0.48841853304756144
3	0.9711544213949912	0.1308890942649201	1.546295297980314	1.24350122556446	-0.37153168747995813

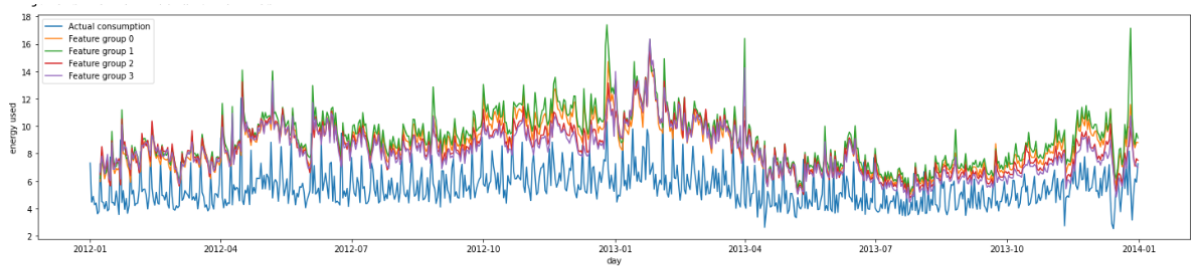
Adversity households results

LSTM – Adversity households dataset Pearson's correlation results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	9.144633698128782	1.6949359340645576	99.40082642283032	9.969996310071048	-31.710626746616853
1	9.282737041055551	1.7370389437313054	101.16583240627847	10.058122707855501	-32.2914514138538
2	8.197724919487447	1.5525462300274695	77.74585620845318	8.817361068281892	-24.584452112228664
3	8.469014683025119	1.6009673951866952	84.02534773063859	9.166534117682572	-26.650894723751808

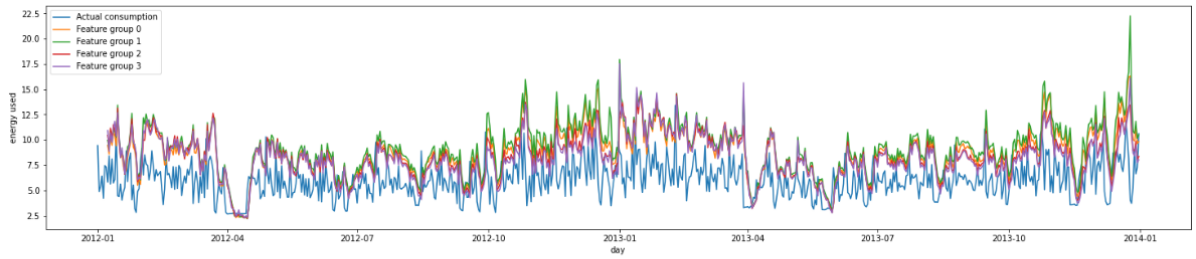
Household 1 - MAC004508 Acorn K



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.235884493809051	0.6411490087986887	13.175407845813368	3.629794463301382	-6.363735269501625
1	3.6824732330420544	0.7263755060579803	16.92091194508134	4.113503609464971	-8.457097460684695
2	2.9707944403482816	0.5922367169754551	11.422418482274596	3.3797068633647203	-5.38398953757322
3	2.795084932382437	0.5570996579403362	10.459735555117598	3.234151442823542	-4.8459460624100865

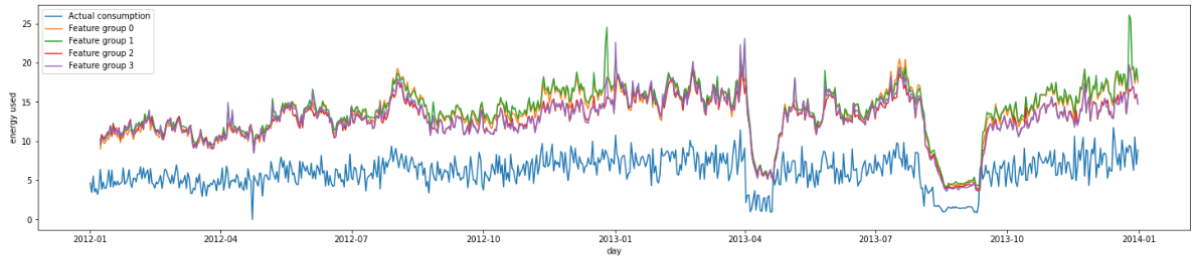
Household 2 - MAC004507 Acorn L



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	2.95074142875544	0.5671713285055133	12.52768501798867	3.5394469932446606	-2.8673634941813466
1	3.3712274991020093	0.6456919233998927	16.18726501850791	4.023340032672842	-3.9970954500632105
2	2.6981994914715806	0.5209544453914934	10.67388518693369	3.2670912425173695	-2.2950855528181133
3	2.597726803671512	0.5019464973526476	10.278858329052797	3.2060658647402733	-2.1731386450537444

Household 3 - MAC004513 Acorn Q



-- Household forecasting performance per feature group: --

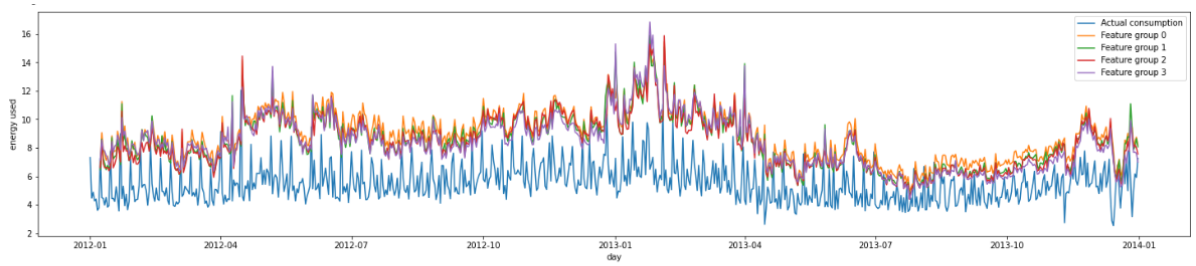
Feature group	MAE	MAPE	MSE	RMSE	R2
0	7.429805560411501	5.155357396208967	60.488233587330114	7.777418182618838	-14.688727829893919
1	7.775920514598037	5.3558283969849185	65.73979323852825	8.10800796981159	-16.050815713170405
2	6.655752097802386	4.945650245136917	48.27049860559934	6.947697359960301	-11.519835179897647
3	6.762788351111505	4.954934243948957	50.37813561435725	7.097755674461981	-12.06648932126673

LSTM - Adversity households dataset Mutual information results

-- Model performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	9.469152075962894	1.7826478779174506	103.9381387562299	10.195005579019067	-33.20375648720185
1	8.812850936386695	1.6630044085125082	90.164332326245	9.495490104583597	-28.67109959464401
2	9.491011949102003	1.761179206805896	106.9098456007228	10.339721737103122	-34.18167987968028
3	9.362529382941688	1.7640707653729266	102.31566047604376	10.115120388608519	-32.669834553777434

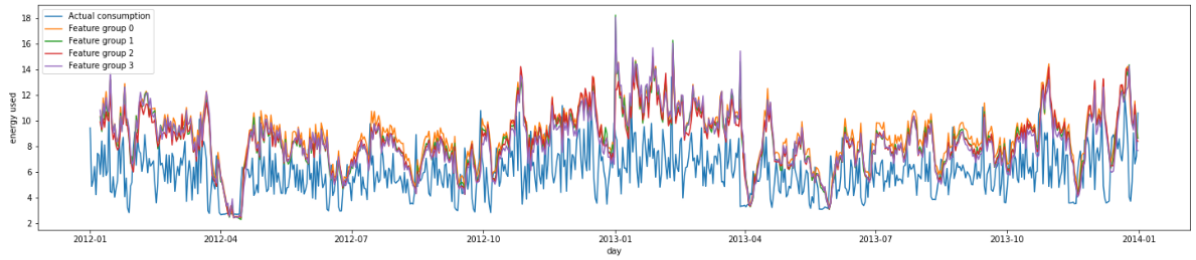
Household 1 - MAC004508 Acorn K



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.441786815977709	0.6848525271863115	14.662038691364431	3.8291041630340157	-7.1946132292751415
1	3.043170658269574	0.6036483093884518	12.127345016829802	3.48243377780968	-5.777972968345058
2	2.951132391811772	0.5851164082498519	11.454049040369702	3.384383110755888	-5.4016678560711915
3	2.966571357018076	0.5894399949245169	11.807380289444042	3.43618688220592	-5.5991446864717425

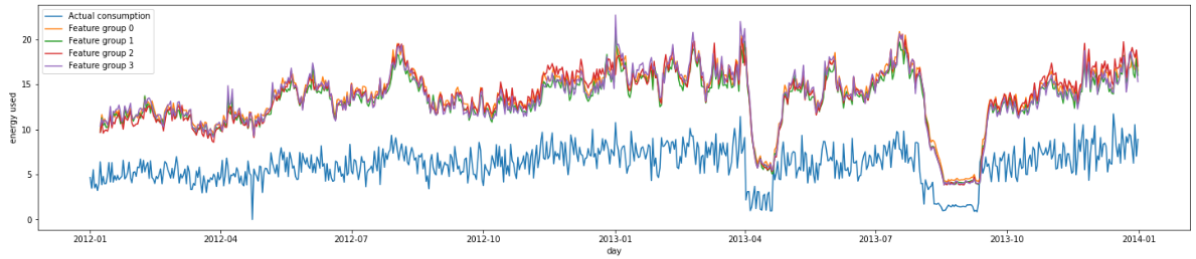
Household 2 - MAC004507 Acorn L



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	3.130379425945834	0.60585804222175	13.571325367708369	3.6839279807982632	-3.189540862447317
1	2.8065215056440787	0.5422363582394534	11.528764647296384	3.3954034586918214	-2.5589914230714617
2	2.7584533143811054	0.5343148644585727	11.126892793678612	3.3356997457323123	-2.4349314284444663
3	2.7859149478970946	0.5398415648263319	11.652702395599377	3.4136054832975904	-2.597251670088351

Household 3 - MAC004513 Acorn Q



-- Household forecasting performance per feature group: --

Feature group	MAE	MAPE	MSE	RMSE	R2
0	7.623818554034904	5.324843269560635	63.25559746743477	7.95333876227052	-15.40649418123824
1	7.0844354818648885	5.041641473570849	54.88165140645116	7.408215129601135	-13.234558371221429
2	7.495909491383998	4.996332831464999	61.8440969810234	7.864101791115334	-15.040395757631615
3	7.410168266359441	5.100730966578448	60.42938047684542	7.773633672668491	-14.673463201096252