University College Cork
School of Engineering

ME in Electrical and Electronic Engineering
Module EE6050 - ME Project
Dissertation

Model-Free Voltage Estimation of Low Voltage Electrical Power Distribution Systems using Smart Meter Data

Anthony O'Malley
119486196@umail.ucc.ie

Dr. Barry Hayes
barry.hayes@ucc.ie

April 2024

Acknowledgements

I would like to thank Dr. Barry Hayes for his help and guidance in this project. I would also like to thank Alisa Weber and my classmates for their support.

Declaration

This report was written entirely by the author, except where stated otherwise. The source of any material not created by the author has been clearly referenced. The work described in this report was conducted by the author, except where stated otherwise.

Name	**Signature**	**Date**
Anthony O'Malley	*Signed: A. O'Malley*	19 April 2024

Summary

Increasing penetration of low carbon technologies (LCT) in residential low voltage (LV) networks expands the need for modelling these networks to preempt voltage issues. LV electrical models are often simplified, incomplete or absent. A methodology for modelling LV electrical networks without an electrical model is proposed and tested across 127 real LV feeders with realistic smart meter data. Using machine learning methods, models could be generated at scale at low cost. This approach uses regression and historical active power and voltage data from smart meters to predict voltage at a node of interest. Finally, these models are tested and compared to electrical models at higher electric vehicle (EV) and solar photovoltaic (PV) penetration.

Modelling LV networks is difficult and expensive as the documentation of these networks is often incomplete, incorrect or out of date, furthermore, there has never been a need to model these networks. The rise of residential LCTs puts additional burden on the LV network, PV systems produce power during the day when the grid loading is low, leading to high voltages and EV charging typically occurs during the evening and night when there is no solar and there is an the grid loading is high. These LCTs can lead to voltages outside the statutory limits and could cause damage to infrastructure or cause PV inverters to disable reducing their effectiveness. Smart meters record active power consumption and the voltage for each customer. Their increasing adoption allows for unprecedented access to the workings of residential grids.

During planning, these voltage calculation models can help distribution system operators make more informed decisions and assist in future proofing the grid. These could be used to find the networks most prone to voltage violations.

The LV networks used are real LV networks in Manchester. The active power, PV and EV datasets used are generated from statistical models of Irish and UK customers [1]. Two machine learning methodology's were developed with the best performing linear and non-linear model. These were trained on 7 days of data across 127 different feeders at at 10% EV and PV penetration. Within a resampling loop the models were trained and the best set of hyperparameters was automatically chosen. The models were then trained on the entire set of training data for that feeder.

The linear model preformed best out of all models tested at 10% EV and PV penetration and, generalised better to higher penetration percentages. It achieved an average mean absolute error of 0.075V across 127 different feeders at 10% penetration. At 100% penetration the linear model works well within the statutory voltage limits but consistently under-predicts voltages outside of these ranges. The average mean absolute error was 0.86V. However, the worst performing models began to break down with a maximum mean absolute error of 6.2V. A sample LCT hosting capacity study is performed to showcase the application of this type of model.

Contents

Acknowledgements . . . i
Declaration . . . i
Summary . . . ii
Nomenclature . . . v

List of Figures **vi**

List of Tables **vii**

1 Introduction **1**

2 Literature Review **3**

3 Background **5**
3.1 Machine Learning Models . . . 5
3.1.1 Linear Regression . . . 6
3.1.2 Histogram Gradient Boosting Regressor . . . 7
3.1.3 Support Vector Regression . . . 8
3.1.4 Neural Network Regression . . . 9
3.2 Regularisation . . . 11
3.3 Repeated Cross-Fold Resampling . . . 12
3.4 Hyperparameters . . . 13

4 Methodology **14**
4.1 Exploratory Data Analysis . . . 14
4.1.1 Feature Distribution . . . 15
4.1.2 Correlation . . . 16
4.2 Hyperparameter Tuning . . . 18
4.3 Modelling Pipeline . . . 19

5 Case Study **21**
5.1 Electrical Models . . . 21
5.2 Model Training . . . 23
5.2.1 Preprocessing . . . 24
5.2.2 Hyperparameter Tuning . . . 24
5.3 Low Carbon Technology Hosting Capacity Study . . . 27

6 Results **29**

7 Discussion **32**
7.1 Modelling Performance . . . 32
7.2 Contribution to the Literature . . . 33
7.3 Assumptions . . . 34

8 Conclusion **37**

References **40**

A Source Code **43**

B Logbook **44**

Nomenclature

ϕ	Power Factor
P	Active Power
Q	Reactive Power
V	Voltage
W	Watts
AC	Alternating Current
DSO	Distribution System Operator
EDA	Exploratory Data Analysis
EPRI	Electrical Power Research Institute
LCT	Low Carbon Technology
LV	Low Voltage Distribution Network
MAE	Mean Absolute Error
MV	Medium Voltage Distribution Network
RBF	Radial Basis Function
RMSE	Root Mean Squared Error
SVR	Support Vector Regression

List of Figures

3.1 Demonstration of the bias-variance trade-off. The red lines are moving average filters with (a) 100, (b) 2 and (c) 20 point window size. 6
3.2 Neural network topology. 10

4.1 (a) Voltage distribution at the customer furthest from the feeder with (b) centring and scaling and (c) with a power transform, centring and scaling. 16
4.2 (a) The distribution of the active power loads. (b) The distribution of the active power loads after centring and scaling. (c) The distribution of the active power loads after transformation, centring and scaling. 17
4.3 Example active vs. reactive power for a one day for a customer. 18
4.4 Example hyperparameter tuning for a histogram gradient boosting regressor on network 1 feeder 1. 19
4.5 A flow chart of training of the linear regression and neural network models. The active power data is of shape NxT where N is the number of loads and T is the number of samples. 20

5.1 Topology of Network 1 Feeder 1 and Network 2 Feeder 1. The blue point is the transformer and the orange points are where loads are connected. 22
5.2 An example (a) active power load, (b) PV generation and EV load are shown over 24 hours. 23
5.3 Correlation matrix for active power loads over 7 days for network 1 feeder 1. . 25
5.4 Impact of preprocessing steps. The error bars represent 1 standard deviation of error. 25
5.5 Repeated resample RMSE for each model at 10% PV and EV penetration. . . . 26
5.6 Number of voltage violations at increasing PV and EV penetration percentages. Sorted by number of violations at 100% PV and EV. 28

6.1 Distribution of voltage at (a) 10% and (b) 100% EV and PV penetration. 29
6.2 Predicted voltage vs electrical model voltage at (a,b) 10%, (c,d) 30%, (e,f) 50%, (g,h) 70% and (i,j) 100% PV and EV penetration. Each black dot is one prediction over 10 days for all feeders. The left column (a,c,e,g,i) are linear regression predictions and the right column (b,d,f,h,j) are neural network regression predictions. A perfect fit is shown with the black dotted line, the red dotted lines are the statutory voltage limits ($1 \pm 0.1 P.U.$). 30
6.3 Distribution of absolute voltage error across all 127 feeders for the linear model (a) and the neural network (b) at 100% EV and PV penetration. 31

List of Tables

5.1 Neural Network Hyperparameters 27

6.1 Mean Absolute Error 31

1. Introduction

Power system modelling enables grid operators and planners to better understand the grid and perform voltage calculations to plan for the future. The introduction of LCTs such as residential PV and EV charging places an increased burden on LV grids. EVs and heat pumps place additional loads on the residential network and solar photovoltaic (PV) frequently generates when local loading is low. This can lead to voltage regulation issues where the voltage drops bellow the statutory limit or PV inverters trip off the grid reducing their usage rates and effectiveness. The integration of LCTs adds to the complexity of meeting required voltage restrictions in LV residential networks. This introduces the need for distribution system operators (DSO) to model these LV networks in greater depth.

To reduce the cost of grid upgrades, monitoring and planning for the future of LV grids is essential. Electrical models are expensive and time consuming to produce and LV models are generally not available to DSOs. There are almost 240,000 km of distribution network in the island of Ireland and, manually creating an electrical model of every LV network would be cost prohibitive. The Electricity Supply Board (ESB) has installed over 1.5 million smart meters in the island of Ireland and this provides an exciting opportunity to leverage this data to create machine learning models of the LV grid [2].

ESB smart meters record interval data of import and export active power (kW) and reactive power ($kVAR$), as well as voltage quality metrics and many other cumulative quantities. One of the analysis channels can collect single phase voltage information [3] [4]. These can be used to model the dynamics of a LV network using machine learning. Several real LV networks will be simulated using EPRI's OpenDSS [5] using this smart meter data. Machine learning models will be trained with the goal of predicting the voltage at the customer furthest from the transformer from the smart meter data.

Several questions have to be answered, can LV grids be modelled with no prior knowledge about the grid and a dataset of customer smart meter data? How accurate and generalisable is

a machine learning based LV network model? How does it performs in outlier cases such as during low and high loading scenarios? How does it compare to traditional electrical models? To what extent is the model generalisable to increasing levels of EV and PV penetration? If successful, a framework for that DSOs could use to model any LV grid from historical customer smart meter data and voltage measurements would be developed which would require no prior knowledge of the topology.

This has been successfully implemented by Bassi et al. [6] using a neural network, which provides no transparency. Future planners and decision makers require a verifiable and understandable model. Neural networks and other types of models are investigated.

The contribution of this dissertation is the development of a LV feeder voltage estimation method that solely uses smart meter data and no information about network topology. This framework was then tested on 127 different real LV feeders with realistic customer smart meter data. These methods are shown to be generalisable to higher LCT penetration levels and produce reliable predictions within statutory voltage limits.

2. Literature Review

A paper by Ferdowsi et al. [7] introduces a method of scalable state estimation using a combination of neural networks. The neural networks had 1 hidden layer for both the LV and MV state estimators, the layers had fewer than 5 neurons.The model estimates voltage magnitudes at all nodes of interest which are unmonitored. This approach requires AC power flows obtained from an electrical network model.

In [8] a wide variety of models are trained as surrogates for traditional LV electrical models with the goal of reducing computational complexity for large scale simulations. The models were trained on data at 15 minute intervals over one year of with integrated PV. These were validated using twelve fold cross validation and the best performing models were ordinary least squares regression and the neural network, the least squares regression had an average RMSE of 100×10^{-6} P.U. or 0.01%. The simulation speed up was dependant on the model but was between 4 and 20 times faster than the electrical model simulation. These models did not include an integrated MV feeder, which makes it unclear how the MV and LV models will interact. The surrogate models required a lot of training data with large variety. This method also requires a preexisting electrical model.

Liu et al. [9] apply an neural network based state estimation method for MV grid monitoring. This methodology assumes that the grid has an electrical model and only some of the busses are monitored. The neural network is trained using the grid model and a scenario generator. During deployment it is given the live measurements and it outputs voltage and line loading estimations. They test the strategy on the same network with a variety of PV, EV and customer loads. This approach is being applied practically to a field test on a suburban grid in Germany and the results from the field test still have to be seen. The single phase voltages at the transformer feeders, the single phase active powers and, the current per LV feeder were measured and a three hidden layer neural network was trained. The results were validated using the electrical model and the average deviation was 0.005 p.u. The deviation for line loading was up to 20%, the largest errors were during high grid loading, for loadings larger than 50%,the line

loading errors are less than 8%. This method requires an electrical model of the network to be developed to train the neural network prior to deployment. If there is a model, this method seems to be suitable for state estimation, but at scale, creating an electrical model of every LV grid is expensive. It is unclear how well this method will generalise to higher PV and EV loadings.

In [10] the authors propose an algorithm to calculate the PV hosting capacity using only the smart meter data at that location. This does not take into account the hosting capacity across the entire LV network and this does not produce a replacement for electrical models.

Bassi et al. [6] propose a methodology for training a neural network to predict voltage from customer smart meter data. The smart meter dataset is generated using an electrical model of a real LV network and realistic customer smart meter data. The LV network used is a town in Australia with 146 customers, it is connected to an integrated MV feeder so that the voltages include upstream effects, the simulation occurs over 3 weeks with 20% PV penetration. The smart meter data had a resolution of 30 minutes composed of active and reactive loads at each of the customers. Two networks were trained, one with active and reactive power as inputs, and one with active, reactive and aggregate active and reactive power as inputs. The first network performed better at lower PV penetration with an average error of 0.18 V but worse at 100% PV penetration with an average error of 1.61 V, but the second network generalised to 100% PV penetration better with an average deviation of 0.72V. At higher voltages introduced by PV, the first model tended to overestimate the voltage, while the second tended to underestimate it. This paper does not account for the effects of EV charging, or how this methodology functions on an unseen network.

3. Background

3.1 Machine Learning Models

The 'no free lunch theorem' [11] states that no one machine learning algorithm performs optimally across all datasets, this underscores the importance of algorithm selection tailored to the unique characteristics of each problem. Bias refers to models that make overly simplistic assumptions, while variance measures the model's sensitivity to fluctuations in the training data. The bias-variance trade-off highlights the delicate balance between model simplicity and complexity, emphasising the need to minimise both bias and variance to achieve optimal predictive performance.

The 'no free lunch theorem', asserts that there is no universally superior machine learning algorithm. No single algorithm outperforms all others across all possible datasets. This theorem emphasises the importance of understanding the characteristics of both the problem domain and the data when selecting an appropriate machine learning algorithm. Different algorithms have different strengths and weaknesses, and their performance depends on factors such as the nature of the data, the size of the dataset, and the complexity of the problem.

The bias-variance trade-off refers to the balance between bias and variance. Bias measures the difference between the expected prediction of the model and the true value being predicted. A high bias indicates that the model is overly simplistic and fails to capture the underlying patterns in the data. On the other hand, variance measures the variability of the model's predictions across different datasets. A high variance suggests that the model is overly sensitive to fluctuations in the training data and may not generalise well to unseen data. The bias-variance trade-off is demonstrated in Fig. 3.1, (a) has a high bias and is under-fit, while (b) has low bias but high variance and is over-fit, the model has captured some of the noise in its prediction. (c) is balanced and is the best fit.

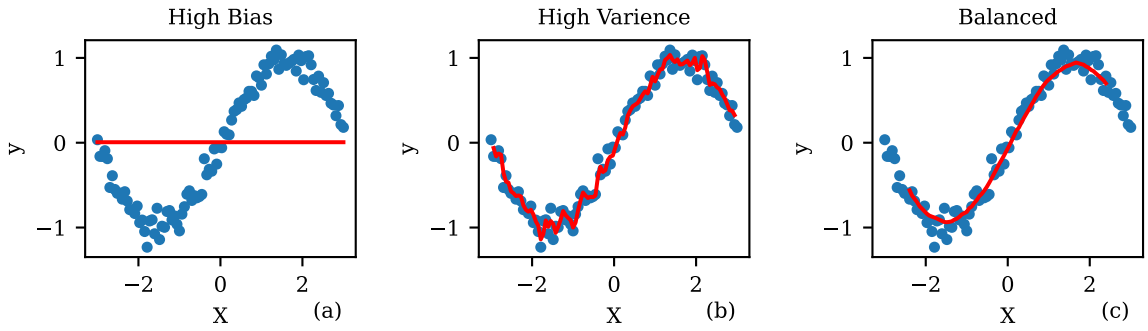


Figure 3.1: Demonstration of the bias-variance trade-off. The red lines are moving average filters with (a) 100, (b) 2 and (c) 20 point window size.

The bias-variance trade-off suggests that as bias is reduced in a model to make it more complex and expressive, it is prone to overfitting. Conversely, variance is reduced by simplifying the model, more bias may be introduced, resulting in under-fitting. The goal is to find the optimal balance between bias and variance that minimises the model's overall error on unseen data.

The 'no free lunch' theorem and the bias-variance trade-off highlight the importance of model selection, hyperparameter tuning, and validation techniques. To help reduce bias and variance, these decisions are made with the training data being split into repeated resampling to increase the probability that the models developed are not over-fit to produce robust and generalisable models.

3.1.1 Linear Regression

Linear models work well when the relationship between the predictors and target falls along a hyperplane. Assuming the relationship between active power and voltage is linear, for each time step i, the voltage can be written as:

$$y_i = \beta_0 + \beta_1 p_{i,1} + \beta_2 p_{i,2} + \ldots + \beta_N p_{i,N} + e_i \tag{3.1}$$

The voltage can be expressed as y_i which is a weighted sum of $p_{i,j}$, which is the active power load at each of the N loads. β_0 is the intercept and each β_j is the weight for each load. e_i is an error term and describes what cannot be captured by the model. Ordinary least square

(OLS) linear regression functions by minimising the sum of squared errors (SSE) between the predicted voltage and the observed voltage.

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \tag{3.2}$$

Where $\hat{y_i}$ is the predicted voltage. The coefficients $\hat{\beta}$ with minimised error can be shown to be found with:

$$\hat{\beta} = (P^T P)^{-1} y \tag{3.3}$$

Where X is a matrix of the predictor measurements over time and y is the voltage over time vector [12]. Linear regression's cousins are not particularly useful in this application, the active power loads are not highly correlated, eliminating the need for partial least squares (PLS), every load would effect voltage and therefore feature selection would worsen results. There is no noise or measurement error and robust regression methods that are robust against outliers show similar results as OLS linear regression.

3.1.2 Histogram Gradient Boosting Regressor

This model excels in handling nonlinear problems and is robust in the presence of missing values, outliers, and a large number of features. It operates by building multiple regression trees with a weak relationship to the output (voltage) and combining them to get improved performance. These trees are then amalgamated through a process called boosting, an ensemble technique aimed at enhancing performance by combining multiple weak learners into a robust single learner.

Boosting is an ensemble technique used to reduce bias and variance [13]. It combines a set of models that have their answer only weekly correlated to the output to produce a single model with its answer strongly correlated to the output. Initially, a basic tree model is fitted to the data, followed by successive iterations where subsequent trees are tailored to the errors of preceding ones. This iterative approach allows the model to integrate numerous weak regression trees [12].

The histogram gradient boosting regressor uses histograms of data to speed up gradient boosting regression. The SKLearn implementation used was inspired by Microsoft's XGBoost [14] and LightGBM [15].

Algorithm 1: Gradient Boosted Trees for Regression adapted from [13]

Initialise model $F_0(x) = \hat{y}$

for $m = 1$ **to** M **do**

- Compute the residuals between the observed value and the prediction;
- Train regression tree using the residuals as a response ;
- Predict each sample using the tree fit from the previous step;
- Update the model with the tree;

Where M is the number of regression trees fit.

3.1.3 Support Vector Regression

Although support vector machines were originally designed to predict discrete classes, they have since been expanded to regression to predict continuous variables [16]. They are effective with high numbers of features and are memory efficient. Support vector regression (SVR) represents data points as vectors in a high-dimensional space. It seeks to find a hyperplane that separates the data points while minimising error, thus effectively modelling the underlying relationships in the data. The margin of error, or epsilon, is a parameter that adds robustness to the model by making it less sensitive to changes within the margin of error. Any data points that fall within this zone are regarded as accurate and do not contribute to the error term.

The loss function used is shown in (3.4) if the absolute difference between the actual and predicted values is less than or equal to epsilon, the loss is zero. If the difference exceeds epsilon, the loss is proportional to the difference beyond epsilon.

$$\text{Loss} = \begin{cases} 0 & \text{if } |y_i - \hat{y}_i| \leq \epsilon \\ |y_i - \hat{y}_i| - \epsilon & \text{if } |y_i - \hat{y}_i| > \epsilon \end{cases} \tag{3.4}$$

The kernel trick allows SVR to implicitly map the input data from its original feature space into a higher-dimensional space. This allows SVR to find nonlinear relationships.The Radial Basis Function (RBF) kernel is one of the most commonly used kernels, it measures the similarity between two data points in the original feature space based on the Euclidean or straight line distance between them. The RBF kernel function is defined as:

$$K(x_i, x_j) = e^{\frac{-||x_i - x_j||^2}{2\sigma^2}} \tag{3.5}$$

$K(xi, xj)$ represents the similarity between data points x_i and x_j, $||x_i - x_j||$ denotes the Euclidean distance between the data points and σ is a hyperparameter that determines the spread or width of the kernel. It controls how rapidly the similarity decreases as the distance between data points increases. This hyper parameter helps control the flexibility of the model [17] [18]. It can be determined via a heuristic approach [12]:

$$\sigma = \frac{1}{N * \text{Var}(X)} \tag{3.6}$$

Where N is the number of features, X is the training dataset and $Var(x)$ is the variance of the features which is given by:

$$\text{var}(X) = \frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2 \tag{3.7}$$

SVR is a versatile regression technique capable of handling linear and non-linear data relationships, making it suitable for various real-world applications.

3.1.4 Neural Network Regression

A neural network or multilayer perceptron regression is an inherently non-linear model that can adapt to many problems. It has a predictor number of inputs and one or more hidden layers each with a number of hidden units. The output of each hidden unit h_k in each layer is:

$$h_k = f(\beta_{0,k} + \sum_{j=1}^{N} p_j \beta_{j,k}) \tag{3.8}$$

The output of each unit is a weighted sum of the inputs or outputs of the previous layer which is then transformed by an activation function $f(u)$. The hidden layers have a non-linear activation function, a rectified linear unit transformation (RELU) (3.9) was chosen. The output layer has one unit and has a linear activation function [12]. The RELU activation function is defined as:

$$f(u) = \begin{cases} u & \text{if } u > 0 \\ 0 & \text{if } u \leq 0 \end{cases} \tag{3.9}$$

Figure3.2 shows the topology of a single layer neural network model. There are N inputs corresponding to the number of loads. There are K hidden units, each of which outputs a weighted sum of the loads (3.8) with the RELU activation function. The output is a weighted sum of hidden units (3.8) with a linear activation function.

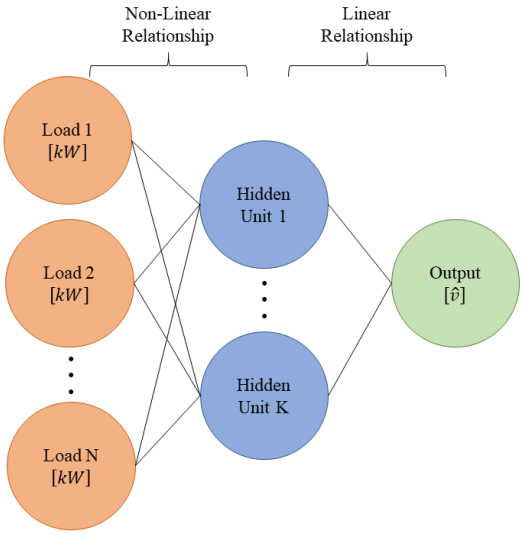


Figure 3.2: Neural network topology.

The parameters are optimised to minimise the sum of squared errors (3.2), the parameters are initialised at random values and often the solution found is local. The neural network is also highly prone to overfitting, this can be addressed with regularisation, which penalises the optimisation for making the parameters large. Hyperparameters, such as the strength of the regularisation, are set prior to the training process and affect the architecture and behaviour of the network.

3.2 Regularisation

Regularisation is a technique that is used to reduce the occurrence of overfitting, which occurs when a model learns to memorise the training data rather than generalise well to new, unseen data. This is achieved by introducing penalties to the size of the parameters in the model to the learning process to promote simpler models that generalise better. This helps balance bias and variance by reducing the complexity of the models produced. The strength of the regularisation is a tuneable hyperparameter.

Lasso or L1 regularisation adds a penalty term proportional to the absolute value of the model's coefficients. The regularisation term is calculated as the sum of the absolute values of the coefficients.

$$\text{L1 term} = \lambda \sum_{i=1}^{p} |w_i| \tag{3.10}$$

Where, λ is the regularisation hyperparameter which controls the strength of regularisation, w_i represents the p model parameters. Ridge or L2 regularisation adds a penalty term proportional to the square of the of the model's coefficients [17]. The regularisation term is calculated as the sum of the squares of the coefficients.

$$\text{L2 term} = \lambda \sum_{i=1}^{p} w_i^2 \tag{3.11}$$

Both L1 and L2 regularisation are added to the original loss function of the model and the combined loss function becomes:

$$\text{Total Loss} = \text{Original Loss} + \text{Regularisation Term} \tag{3.12}$$

Both L1 and L2 regularisation are used to prevent overfitting by adding penalty terms to the model's loss function, the models minimise the prediction error and the size of the parameters. L1 regularisation encourages sparsity in the model and can preform feature selection as it can set some parameters to exactly zero, while L2 regularisation encourages smaller coefficients but does not remove any features. L2 regularisation is used as the active power consumption and generation at every customer is expected to effect the voltage prediction.

3.3 Repeated Cross-Fold Resampling

Cross-fold resampling, also known as cross-validation, is used to assess the performance of a machine learning model by partitioning the available data into subsets, training the model on some of the subsets, and evaluating its performance on the remaining subset. This is to simulate the process of training on one set of data and testing on another, ensuring that the model's performance is evaluated on data it hasn't seen during training. By training and evaluating the model on multiple subsets of the data, cross-validation reduces the bias that may result from using a single train-test split, it provides reliable estimates of the model's performance by averaging performance metrics across multiple iterations, reducing the variability in the results.

In cross fold resampling the available dataset is divided into k equally sized randomly selected subsets, or folds. Each fold contains approximately the same number of samples. The model is then trained k times, with each iteration using a different fold as the validation set and the remaining $k - 1$ folds as the training set. For each iteration of training and validation, the model's performance is evaluated using Root Mean Squared Error (RMSE). [19].

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \tag{3.13}$$

RMSE is calculated on the validation set where y is the predicted voltage, $\hat{y}$ is the true voltage and n is the number of points in the test set. This process allows for more efficient use of the available data by using each data point for both training and validation.

Repeated cross-fold resampling is used to improve the reliability and robustness of the performance estimates obtained through cross-validation. While standard cross-validation provides a good estimate of the model's performance, repeating the process multiple times with different random data splits further reduces the variance in the performance metrics and increases the confidence in the evaluation [20].

The aggregated RMSE provides an estimate of how well the model is expected to perform on unseen data. This estimate is used to assess the model's generalisation ability and compare

different models or hyperparameters. All decisions including model selection, preprocessing decisions and hyperparameter tuning are preformed using this strategy to develop robust and generalisable models.

3.4 Hyperparameters

Hyperparameters determine the behaviour and performance of the machine learning models these are not learned from the data but rather specified by the user. Selecting close to optimal hyperparameters is essential for building models that perform well and reduce the likelihood of overfitting. The search space of hyperparameters was broken into discrete combinations of hyperparameter values. For each combination of hyperparameters models were trained using repeated cross-fold resampling. The RMSE (3.13) was calculated for each resample and the hyperparameter combination with the lowest average RMSE was selected.

Tuning hyperparameters can be a very high dimensionality problem with each hyperparameter added, the search space increases exponentially. This can be mitigated by first doing a course tuning and the hyperparameters that are most impactful can be finely tuned.

4. Methodology

This section addresses the function and development of the modelling pipeline covering preprocessing, model selection, hyperparameter tuning and validation. Using a data-driven approach, modelling decisions are made with repeated cross-validation. Which ensures robustness in model evaluation and reduces the risk of overfitting. Through repeated sampling from the dataset, multiple estimates of a decisions impact on model performance are obtained, which can be used to understand the change in predictive performance and potential trade-offs with increased variance. The resamples are evaluated using RMSE (3.13) to quantify the model error. Aggregating across resamples to helps mitigate the risk of overfitting.

Exploratory Data Analysis (EDA) helps understand the underlying structure of the data and informs modelling decisions. It is also used to check if the data needs to be cleaned or if any preprocessing may be needed. Using visualisations and statistical analysis, patterns, outliers, and distributions can be identified which inform preprocessing and feature engineering. Hyperparameter tuning aims to optimise model performance by systematically searching for a set of hyperparameters that minimise modelling error reliably over many resamples. This is accomplished with grid search with repeated cross resampling. Integrating systematic verification using resampling to inform modelling decisions produces more reliable, reproducible and generalisable models. This approach maximises the predictive ability of the models used and enhances their applicability across diverse datasets.

4.1 Exploratory Data Analysis

OLS linear, SVM and neural network regression, can be susceptible to numerical issues when dealing with correlated features [21], this can be mitigated by using partial least squares linear regression which is directly built to handle correlated features and to reduce dimensionality [12]. By identifying and removing features with little predictive value during EDA, model stability can be improved. This decreases the complexity of the learning task, models can learn

efficiently from this reduced dataset and the training data requirement can be reduced.

The target variable's and the features' distributions can be determined using EDA. These distributions are useful for identifying variability and central tendency. Some machine learning models function best when all of the data is on the same scale, which can be achieved by scaling and centring the data to have a standard deviation of 1 and a mean of 0. Zero variance features and outliers can also be found with the use of these distributions. Many models assume variables are normally distributed and transformations to skewed variables may improve model performance.

4.1.1 Feature Distribution

For linear regression models, support vector machines and neural networks, it is essential that predictors are on the same scale, this is achieved by centring and scaling these data about zero with a standard deviation of one. Normalising the distributions of the data may improve predictive utility [19].

The voltage distribution Fig. 4.1(a) has been centred and scaled in Fig. 4.1(b). The distribution is left skewed, which can be corrected using the Yeo-Johnson power transform [22] to increase symmetry and reduce skew as is seen in Fig. 4.1(c). These transformations may improve predictive performance. Similarly to the voltage transformations, Fig. 4.2 shows the distributions of each customer active power load (a) after centring and scaling (b), and power transformation, centring and scaling (c). The active power loads are right skewed, not centred at zero and do not have a standard deviation of 1. As each feature is predicting the same quantity the data across all customers can be aggregated to calculate the constants for centring, scaling and power transformation. It is important that the constants used by centring, scaling and power transformation are produced within the training data or resampling loop to prevent data leakage and overly optimistic results. Within the resampling process it is important to also recalculate these variables to understand the variance added and reduce that probability off overfitting. The preprocessing on the target variable inverse has to be applied to to the model output to scale it back to the original range.

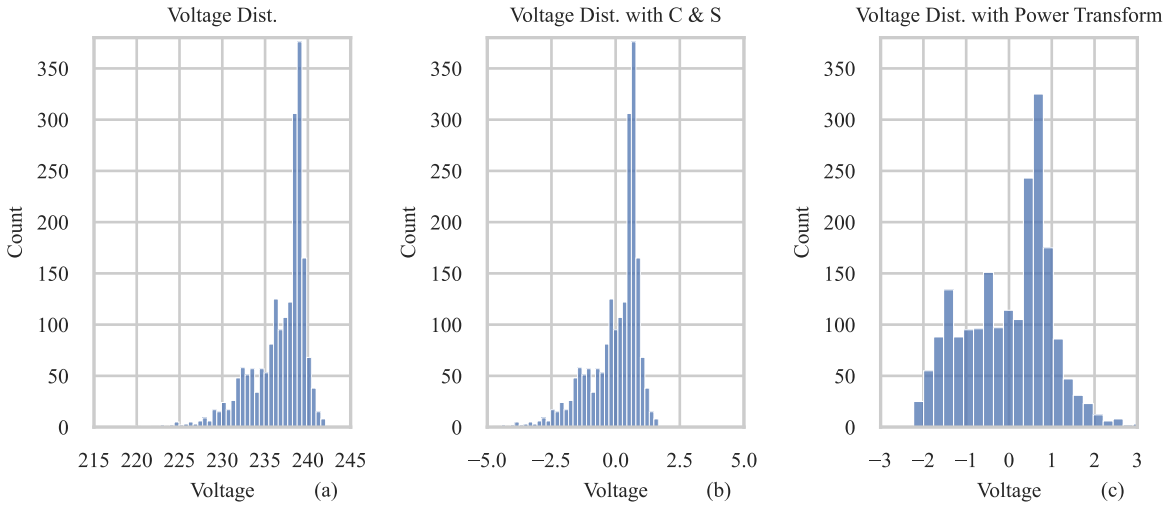


Figure 4.1: (a) Voltage distribution at the customer furthest from the feeder with (b) centring and scaling and (c) with a power transform, centring and scaling.

4.1.2 Correlation

If the predictors are correlated it can cause numerical issues in neural networks, linear regression and support vector machines. Some models such as partial least squares linear regression are purpose built to handle correlated predictors [19]. If the predictors are correlated or colinear, it suggests that redundant information is shared and the number of predictors or dimensions can be reduced. In Fig. 4.3 the active and reactive power for a customer over 7 days are highly correlated with each other. Due to the fact that the power factor (ϕ) is near unity in residential areas, the active and reactive powers at each load are highly correlated. The number of features can be halved without greatly reducing the information by excluding the reactive powers from the model inputs. Partial least squares or principal component analysis could help reduce the dimensionality without removing the reactive power data, however, it is unlikely that the reactive power data adds much information. If Q were to be introduced, it is expected that dimensionality reduction would be effective.

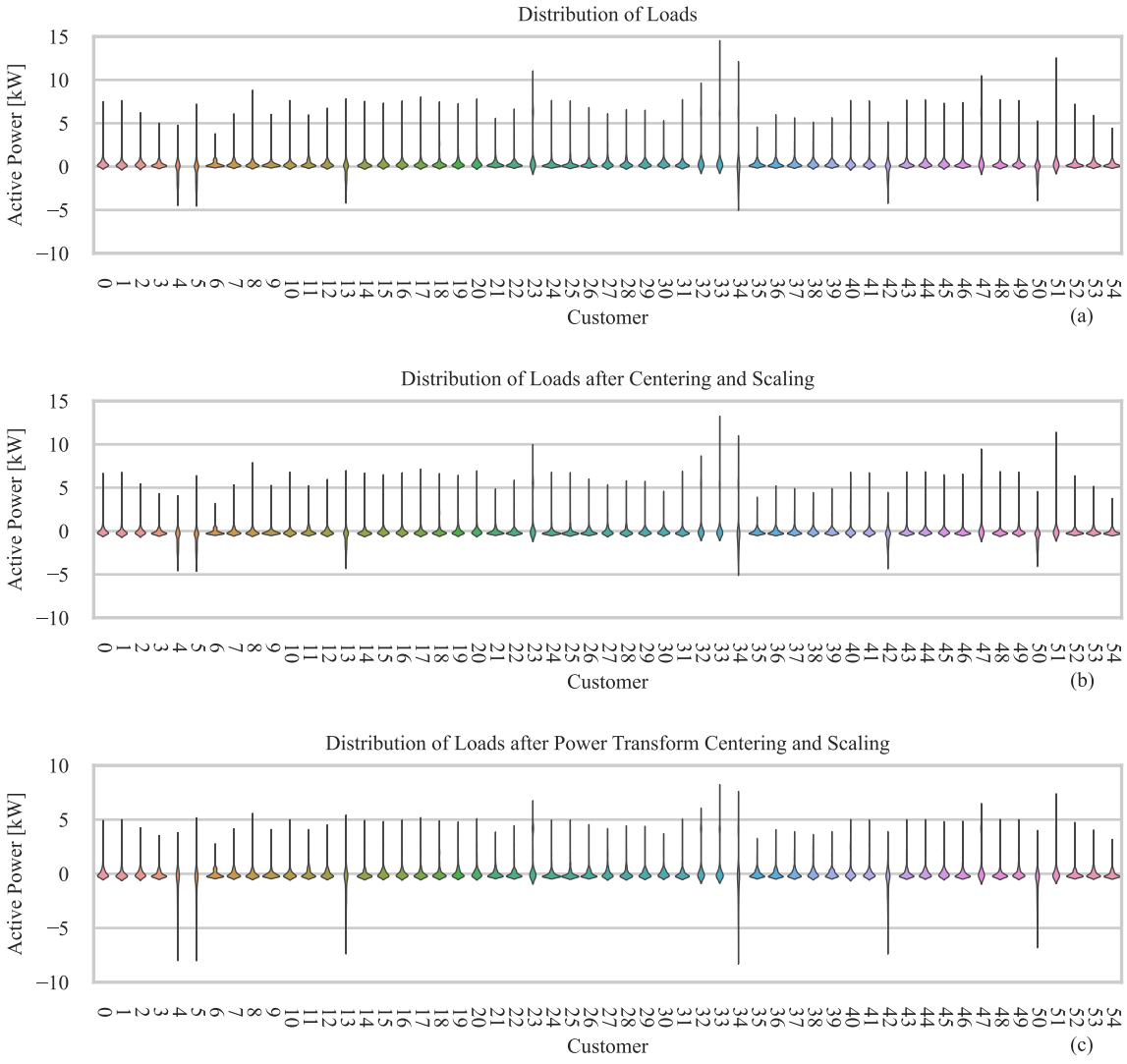


Figure 4.2: (a) The distribution of the active power loads. (b) The distribution of the active power loads after centring and scaling. (c) The distribution of the active power loads after transformation, centring and scaling.

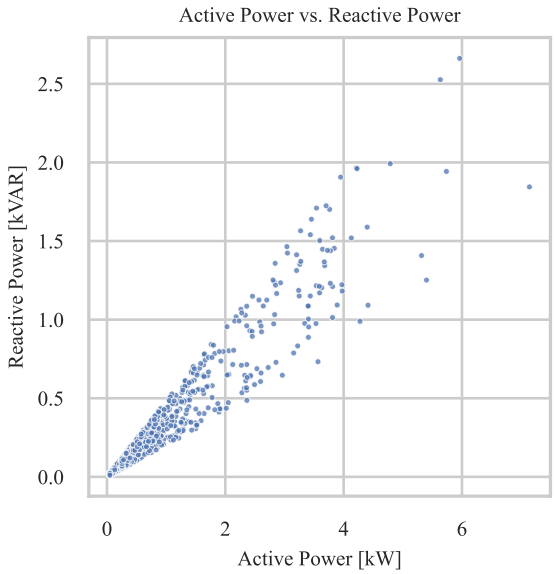


Figure 4.3: Example active vs. reactive power for a one day for a customer.

4.2 Hyperparameter Tuning

OLS linear regression has no hyperparameters, but many other machine learning models have at least one hyperparameter. The neural network algorithm used has 11 hyperparameters. The selection of these parameters is critical to performance and this can be a computationally intense process. In order to make a reliable guess at what the best hyperparameters are, the search space is split into a grid and the models performance is systematically evaluated at each combination of hyperparameters. To reduce bias and variance and to reduce overfitting and the probability to produce optimistic results each combination is trained with repeated cross fold resampling [12]. For each combination the average validation RMSE (3.13) is calculated across all resamples and the optimal combination has the lowest average RMSE. If w values of x hyperparameters are explored with y repeated z fold cross resampling the number of models trained is $w * x * y * z$. The final model is then trained on the the full training set with the best set of hyperparameters found.

In Fig. 4.4 a histogram gradient boosting regressor is tuned with with grid search, each hyperparameter combination is evaluated with 5 repeated, 10 fold resampling, the hyperparameters optimised are l2 regularisation and learning rate. The best combination of these hyperparameters is the combination with the lowest average RMSE over all resamples which is a learning

rate of 0.06 and l2 regularisation weight of 0, which corresponds to no regularisation.

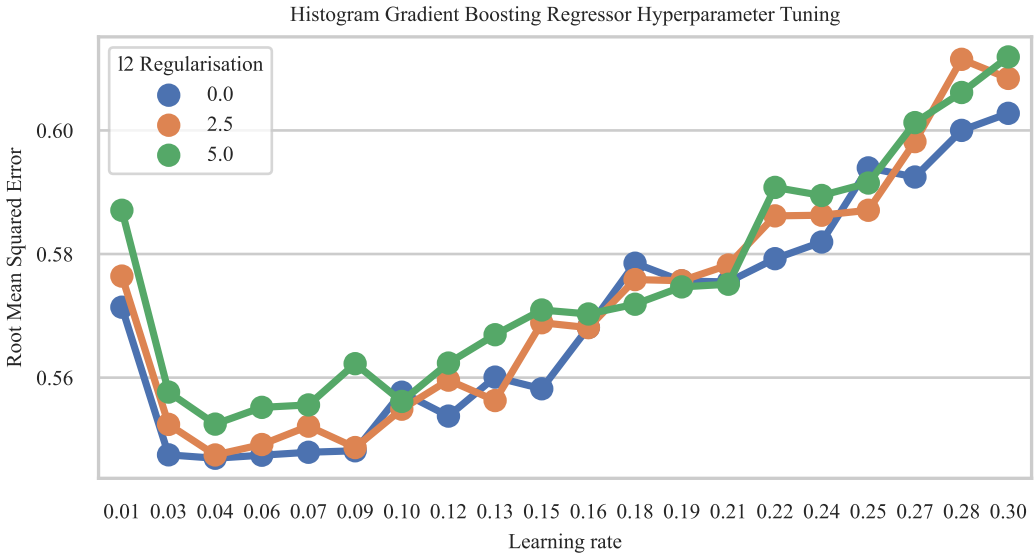


Figure 4.4: Example hyperparameter tuning for a histogram gradient boosting regressor on network 1 feeder 1.

4.3 Modelling Pipeline

The smart meter dataset was constructed from active power and voltage data. The active power data (4.1) is of shape NxT where N is the number of loads and T is the number of samples. In the voltage dataset (4.2), each element is the voltage at the customer furthest from the transformer, at every time step.

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,N} \\ P_{2,1} & P_{2,2} & \dots & P_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{T,1} & P_{T,2} & \dots & P_{T,N} \end{bmatrix} \tag{4.1}$$

$$V = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_T \end{bmatrix} \tag{4.2}$$

The algorithm for splitting the data and training the models is described in Fig. 4.5. The load data is split into a 70% training and 30% test sets. For the neural network the training set was split into repeated cross fold resampling. The preprocessing steps are applied to each fold and a grid search of hyperparameters is performed. The optimal hyperparameters are selected for the model that is trained on the entire training set. The linear model has no hyperparameters and can be directly trained on the preprocessed training set.

The grid search hyperparameter tuning is computationally expensive, however, this process can be parallelised. Each resampled fold can be executed simultaneously, leveraging multi-core processors or distributed computing systems thereby reducing the overall optimisation time.

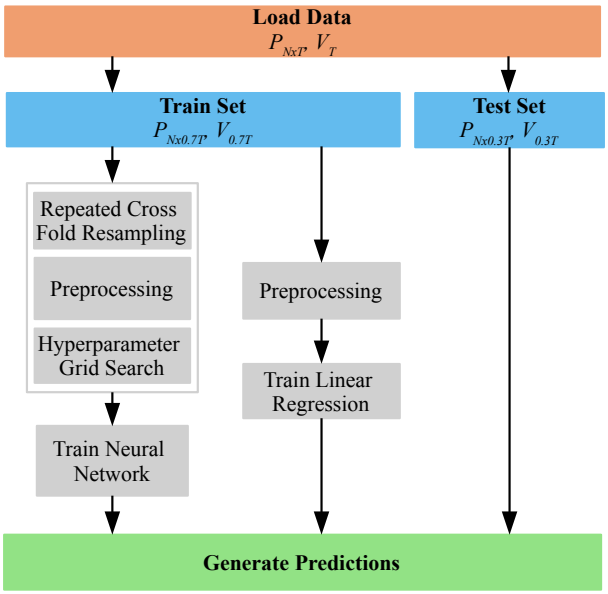


Figure 4.5: A flow chart of training of the linear regression and neural network models. The active power data is of shape NxT where N is the number of loads and T is the number of samples.

5. Case Study

In this section 127 real feeders are modelling using realistic smart meter data and machine learning models are produced to predict the voltage at the customer furthest from the feeder. This customer was chosen as it would exhibit the most extreme voltages and would be of most interest to a DSO, however, this methodology could be applied to any node of interest. The effects on the predictive performance of the model at increasing PV and EV penetrations far beyond what was in the training data are tested. This is to estimate the models generalisability and to quantify its degradation in performance as the residential energy landscape evolves.

The smart meter dataset used for training and testing was constructed from active power load data and voltage simulated using AC power flows. The feeders were simulated for a period of 10 days at each each desired penetration percentage and the voltages at the load furthest from the transformer along the cables were extracted.

An example use case of an EV and PV capacity study is demonstrated, the total number of voltage violations are calculated over a 10 day period at varying EV and PV penetration levels which could be used to rank which feeders are most in need of upgrades. The methodology used is LCT agnostic and could easily be applied to heat pumps or any other low carbon technology.

5.1 Electrical Models

The LV networks, active power, EV and PV load datasets are from [1]. These networks consist of 25 unbalanced real UK residential LV networks. These are split into 128 feeders with 3 - 303 customers, the furthest customer from the transformer was between 43m and 770m. Network 1 feeder 1 and network 2 feeder 2 are shown in Fig. 5.1. Network 13 Feeder 4 was excluded due to the power flow diverging with higher PV and EV penetrations.

The active power load datasets consist of 100 24 hour long datasets with 5 minute resolution,

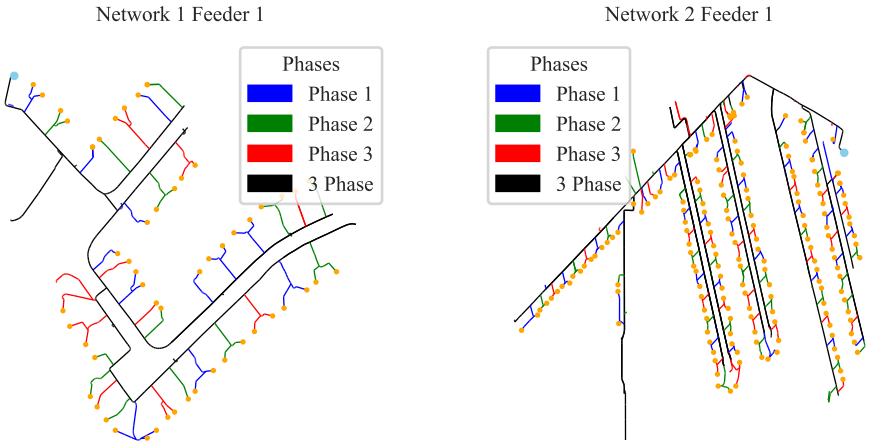


Figure 5.1: Topology of Network 1 Feeder 1 and Network 2 Feeder 1. The blue point is the transformer and the orange points are where loads are connected.

which were generated from UK statistical appliance usage data. This dataset assumes each customer has access to a gas network for heating. The PV dataset has a peak output of 4kW, and the EV charging is set to draw 6kW while charging. The reactive power was calculated from the active power using a uniform distribution of power factors (ϕ) between 0.9 and 0.98 (5.1). This assumption was made as the power factor in residential grids is near unity.

$$Q_{ni} = P_{ni} \tan(\cos^{-1}(\phi_{ni})) \tag{5.1}$$

Q_{ni} is the reactive power for a customer n at time i, similarly, P is the active power and ϕ is the power factor for customer n at time i. It was assumed that the LCTs operate at unity power factor.

A subset of customers was randomly selected and designated with EV loads, and the same process was repeated for PV generation. Each customer fell into one of the following categories: no LCT, only an EV load, only PV generation, or both PV and EV. Daily loads were generated by assigning each customer a random active power load and calculating their reactive power. The EV loads were added to and PV generation was subtracted from the active power of the selected customers. This process was repeated for 10 days. To accommodate networks with more than 100 customers and to diversify daily load patterns, the assignments were made with replacement, allowing for variability in load distribution from day to day. In Fig. 5.2 an exam-

ple load curve is shown over 24 hours with 5 minute sample rate. The voltage of interest was

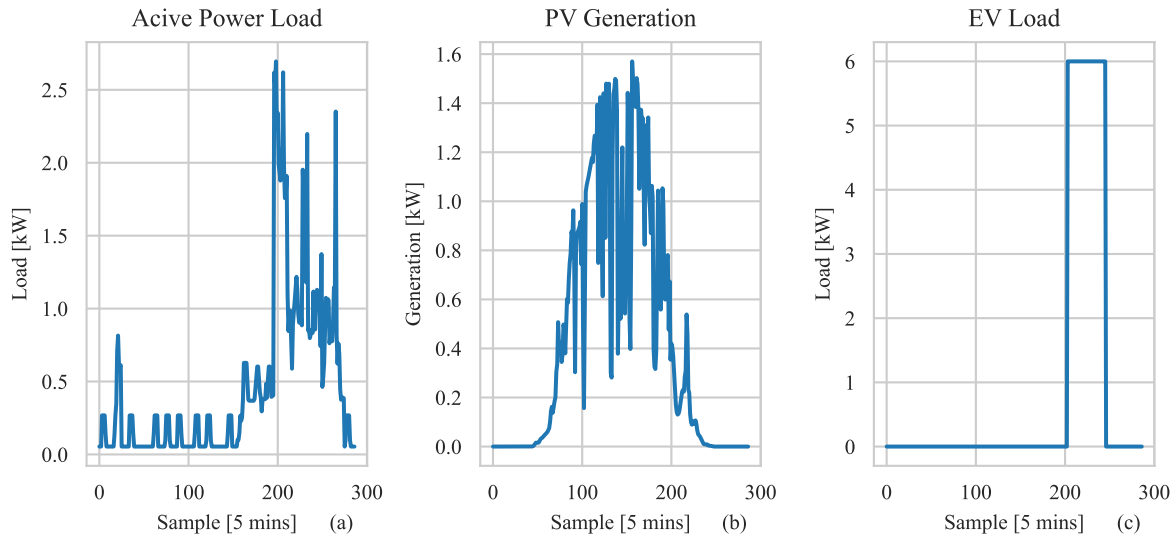


Figure 5.2: An example (a) active power load, (b) PV generation and EV load are shown over 24 hours.

computed using OpenDSS [5] which is an open electric power distribution system software from the Electric Power Research Institute (EPRI). The dss-python library [23] is an implementation of OpenDSS directly in Python [24]. The AC power flows were performed over 10 days at 10%, 30%, 50%, 70% and 100% EV and PV penetration. The smart meter dataset was constructed from the active power data (4.1) for every customer and the voltage at the customer furthest from the transformer (4.2).

5.2 Model Training

The modelling process was developed on Network 1 Feeder 1 and then applied to all feeders. All training was performed on a single machine with an i7-10700 CPU @ 2.90GHz with 8 cores and 64GB of RAM. With the training process fully parallelised, the neural network training including hyperparameter tuning took over 24 hours while the linear model training was performed in under 2 minutes.

5.2.1 Preprocessing

Numerical problems may arise in linear, SVR and neural network regression when features have a high linear correlation. Dimensionality reduction can be achieved with supervised techniques such as PLS or unsupervised techniques like Principal Component Analysis (PCA). In Figure 5.3 shows a sample correlation matrix for network 1 feeder 1 where the relationships between customers are shown, each row or column denotes the active power for each customer and the matrix represents the pairwise relationship between the variables. Each cell in the matrix represents the correlation coefficient between two variables, ranging from -1 to 1, with 1 indicating a perfect positive correlation, -1 indicating a perfect negative correlation, and 0 indicating no correlation. The colour intensity and shading of the cells highlight the strength and direction of the correlations, with darker shades representing stronger correlation. The active powers are not highly correlated with each other, therefore, it is unlikely that dimensionality reduction would improve performance.

In Fig. 5.4 the performance of 5 models with different preprocessing schemes are shown. The RMSE was calculated with 5 repeated 10 fold cross validation and the error bars are one standard deviation. Gradient boosting showed no change in predictive ability to each prepossessing change. Centring and scaling minimally improved performance, while power transformation, centring and scaling made most models worse. Moving forward, centring and scaling was chosen as the preprocessing scheme for all models.

5.2.2 Hyperparameter Tuning

Due to computational constraints each model tested was restricted to tuning with only three hyperparameters. Linear regression without regularisation has no hyperparameters to adjust. Linear and RBF SVR were optimised with respect to epsilon and l2 regularisation. σ was calculated for the RBF with a heuristic (3.6). The most impactful hyperparameters for histogram gradient boosting regression were learning rate and l2 regularisation with 500 iterations. The most important neural network hyperparameters were learning rate, l2 regularisation

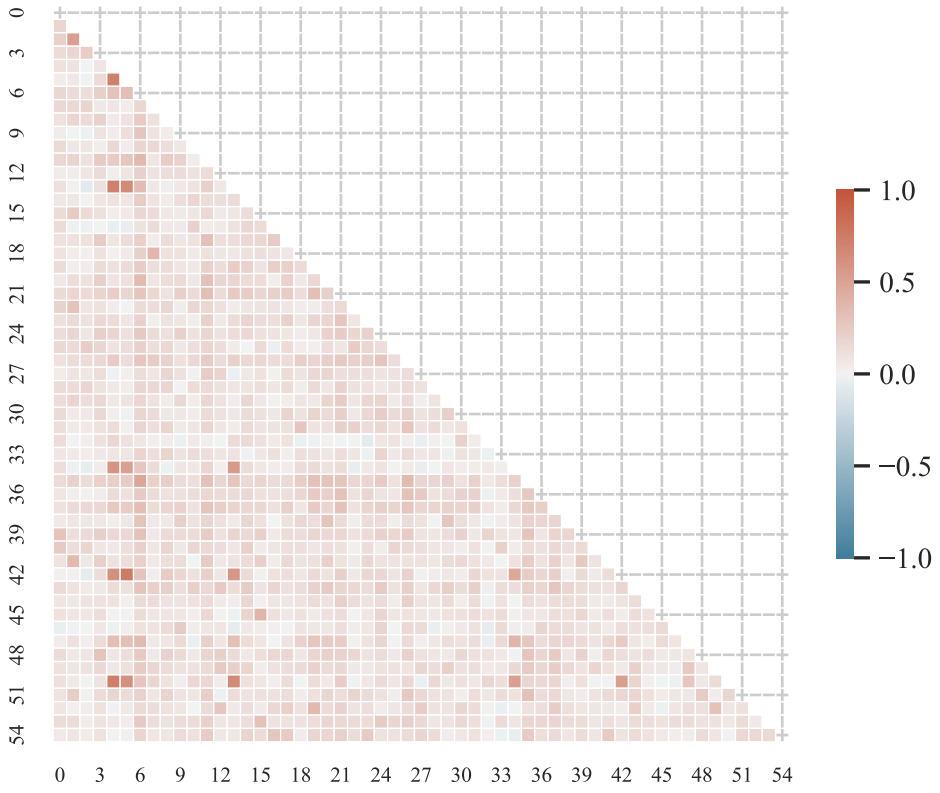


Figure 5.3: Correlation matrix for active power loads over 7 days for network 1 feeder 1.

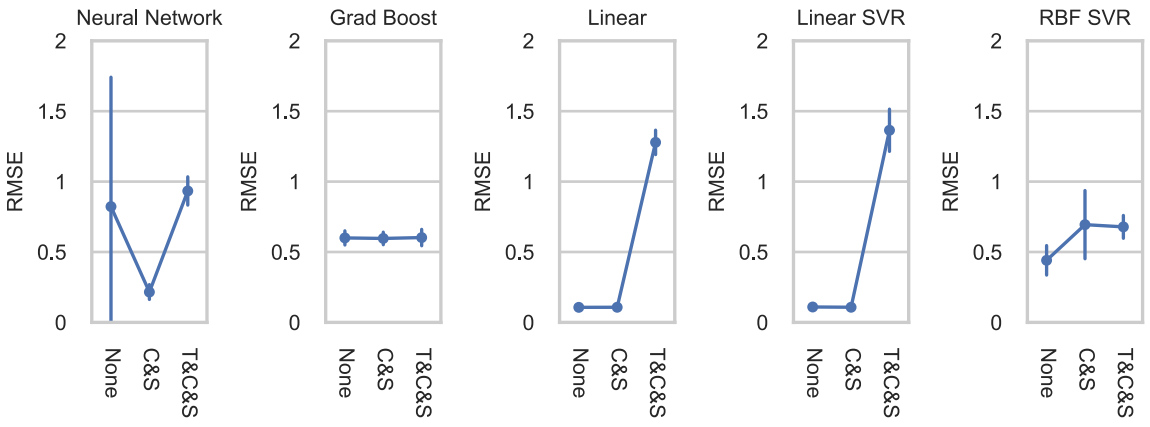


Figure 5.4: Impact of preprocessing steps. The error bars represent 1 standard deviation of error.

and number of hidden units. The hyperparameter ranges searched for every feeder are shown in Table 5.1. The hidden layers used the RELU (3.9) activation function and the output layer had a linear activation function, training stopped when an limit of 200 iterations was reached. Increasing the iteration limits further did not improve results significantly, but did increase computation time.

Boxplots of the RMSEs for each model tested at 10% PV and EV penetration on network 1 feeder 1 with five repeats of 10 fold cross validation are displayed in Fig. 5.5. A boxplot is a graphical representation of a dataset's distribution. It summarises the central tendency, dispersion, and skewness of the data. The line within the box represents the median value of the dataset. The box itself represents the interquartile range, which is the middle 50% of the data, the lines extending from the box represent the variability outside the upper and lower quartiles. They extend to the furthest data point within a distance of 1.5 times the interquartile range from the quartiles. Any data points beyond the whiskers are considered outliers and are plotted individually.

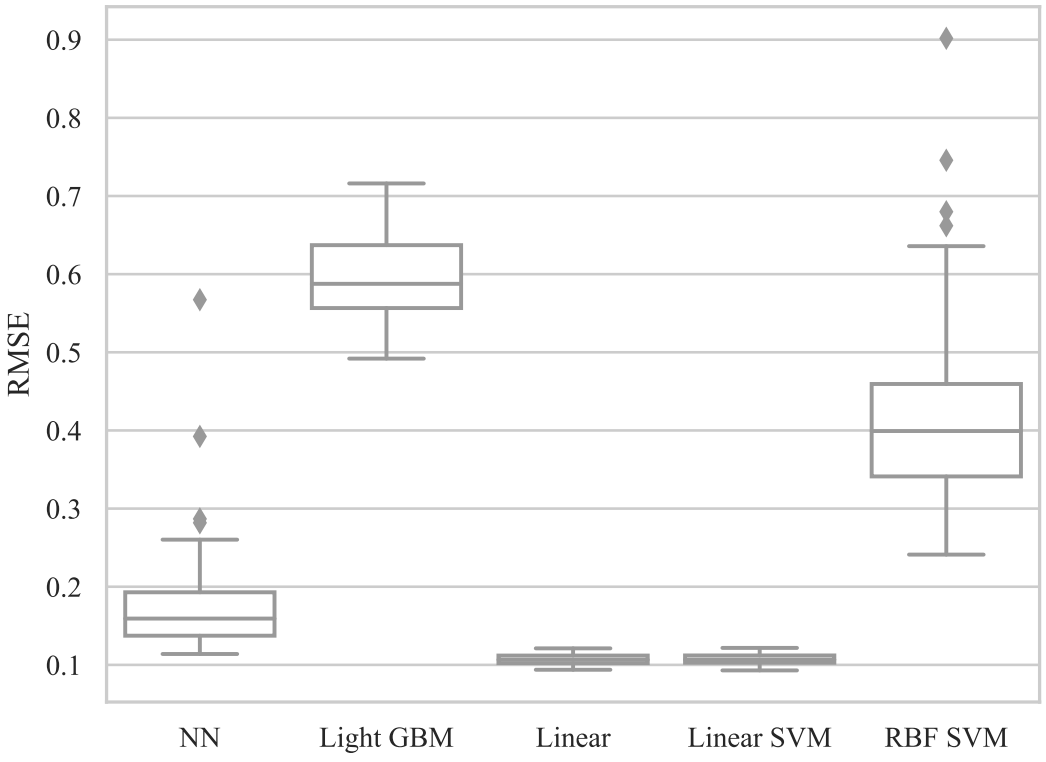


Figure 5.5: Repeated resample RMSE for each model at 10% PV and EV penetration.

Table 5.1: Neural Network Hyperparameters

Hyperparameter	**Range Tested**	**Number Points**
Learning Rate	$1 \times 10^{-3} - 0.1$	20
Alpha	$10 \times 10^{-6} - 1 \times 10^{-3}$	20
Hidden Layer Size	25 - 150	4
Total:		1,600

The best performing model was OLS linear regression, whereas the best performing non-linear model was the neural network. The linear and neural network regression were then applied to every feeder. When training the neural network on each feeder the 10% LCT penetration training set was resampled, preprocessed, the then hyperparameters were tuned. The entire training set was preprocessed and using the best set of hyperparameters the neural network was trained. The linear regression model, was trained on the test set after preprocessing as no hyperparameters had to be tuned. The models were then tested on the remaining 3 days of data with 10% EV and PV penetration as well as 10 days of 30%, 50%, 70% and 100% EV and PV penetration.

The neural network hyperparameters ranges are shown in Table 5.1. Many more hyperparameters could have been searched but due to exponentially scaling computational complexity, these hyperparameters were seen as a good compromise. The grid search was repeated twice with five fold cross validation, this was reduced from 5 repetitions due to computational constraints, this resampling was necessary to reduce the variance introduced by the randomly assigned model parameters. 1,600 neural network hyperparameter combinations were evaluated for each feeder. The hyperparameter combination with the lowest average RMSE (3.13) was chosen. The final model was trained on the whole training set with the optimal hyperparameters.

5.3 Low Carbon Technology Hosting Capacity Study

There are many ways to conduct a LCT hosting capacity study. The method chosen used is to provide a cursory example of what can be done. When using OLS linear regression the

weights at each customer directly correspond an active power injection or load to a change in voltage at the node of interest. A change in a customer load Δx_j would change the voltage by $\beta_j \Delta x_j$. This is desirable in LCT hosting capacity studies as the size of the parameters directly corresponds to the difference in voltage observed at the end of a feeder for a change in active power generation or consumption. OLS linear regression is highly interpretable and well understood, these properties could be used to conduct a hosting capacity study in the future.

The linear regression model was generated using a smart meter dataset with 10% EV and PV penetration. Additional LCT loads were added to the dataset after it has been collected to simulate higher penetrations. A voltage violation is defined as the voltage is above 1.1 P.U. (253 V) or below 0.9 P.U.(207 V). In Fig. 5.6 the total number of violations over a 10 day period with varying amounts of PV and EV penetration is shown. This can be used to rank which networks are most in need of upgrades in the tested scenarios. This method has to be validated against an electrical model and in the future more sophisticated LCT penetration studies can be conducted.

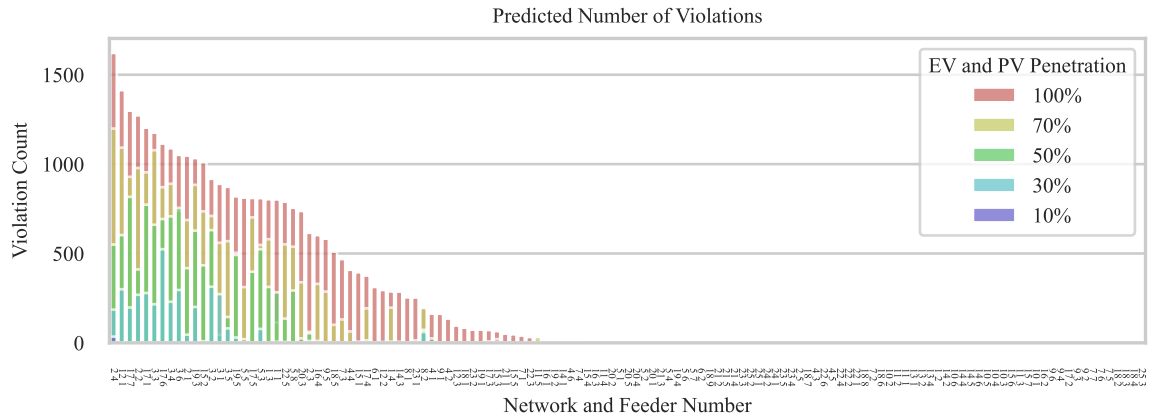


Figure 5.6: Number of voltage violations at increasing PV and EV penetration percentages. Sorted by number of violations at 100% PV and EV.

The number of violations would be dependant on not only the penetration of LCTs but also their exact location. If all EV chargers are near the target, far from the transformer, the voltage impact will be major. It is my recommendation that many combinations of each penetration level are tested and the voltage violations aggregated to rank the networks to find which are most in need of upgrades.

6. Results

Figure 6.1 shows the electrical model voltage distributions calculated using an AC power flow at the customer furthest from the transformer at (a) 10% and (b) 100% EV and PV penetration.

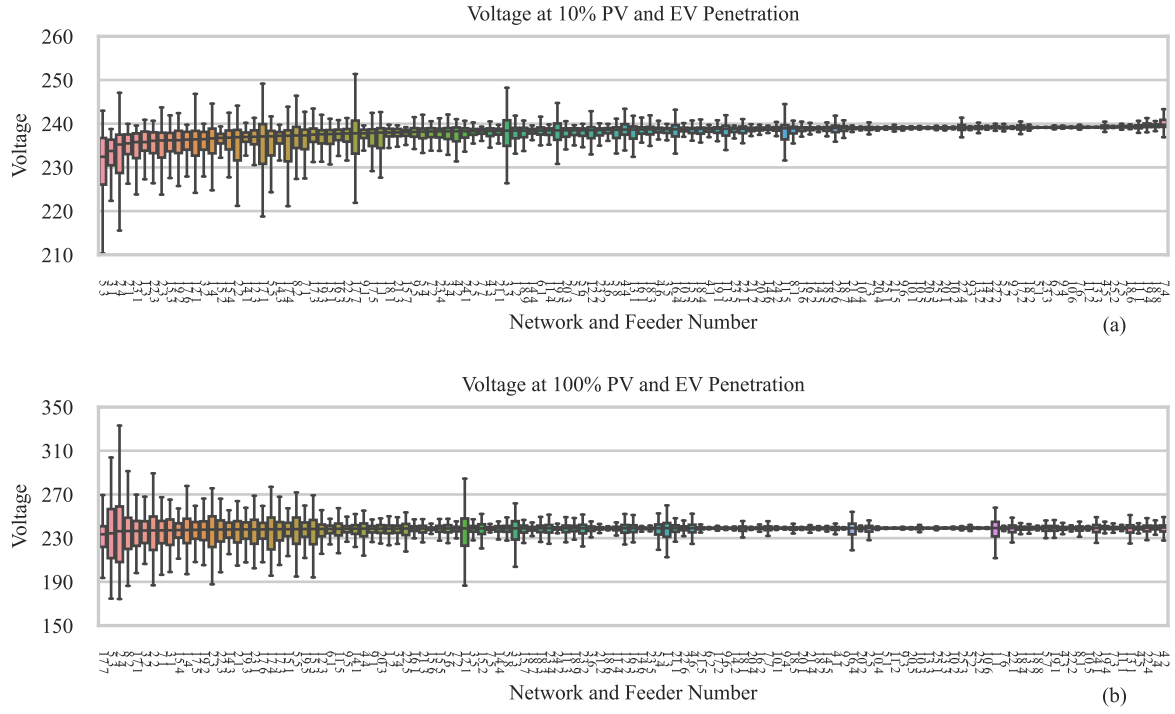


Figure 6.1: Distribution of voltage at (a) 10% and (b) 100% EV and PV penetration.

Figure 6.2 shows the results produced from linear and neural network regression models across 127 different feeders. The models were trained on 7 days of smart meter data for each feeder at 10% PV and EV penetration. All smart meter data used was at a 5 minute sample rate. The validation data at 10% PV and EV penetration was produced from 3 days of data that the model has not been trained on. The validation set size for each penetration percentage is 10 days. Each row in Fig. 6.2 shows the electrical model voltage vs the predicted voltage for 10%, 30%, 50%, 70% and 100% PV and EV penetration. The linear regression results are shown in the left column, while the results for neural network regression are shown in the right column.

In Fig. 6.3 the absolute prediction errors are shown across 127 different feeders at 100% PV and EV penetration, this is sorted from lowest to highest mean error. Table 6.1 presents the

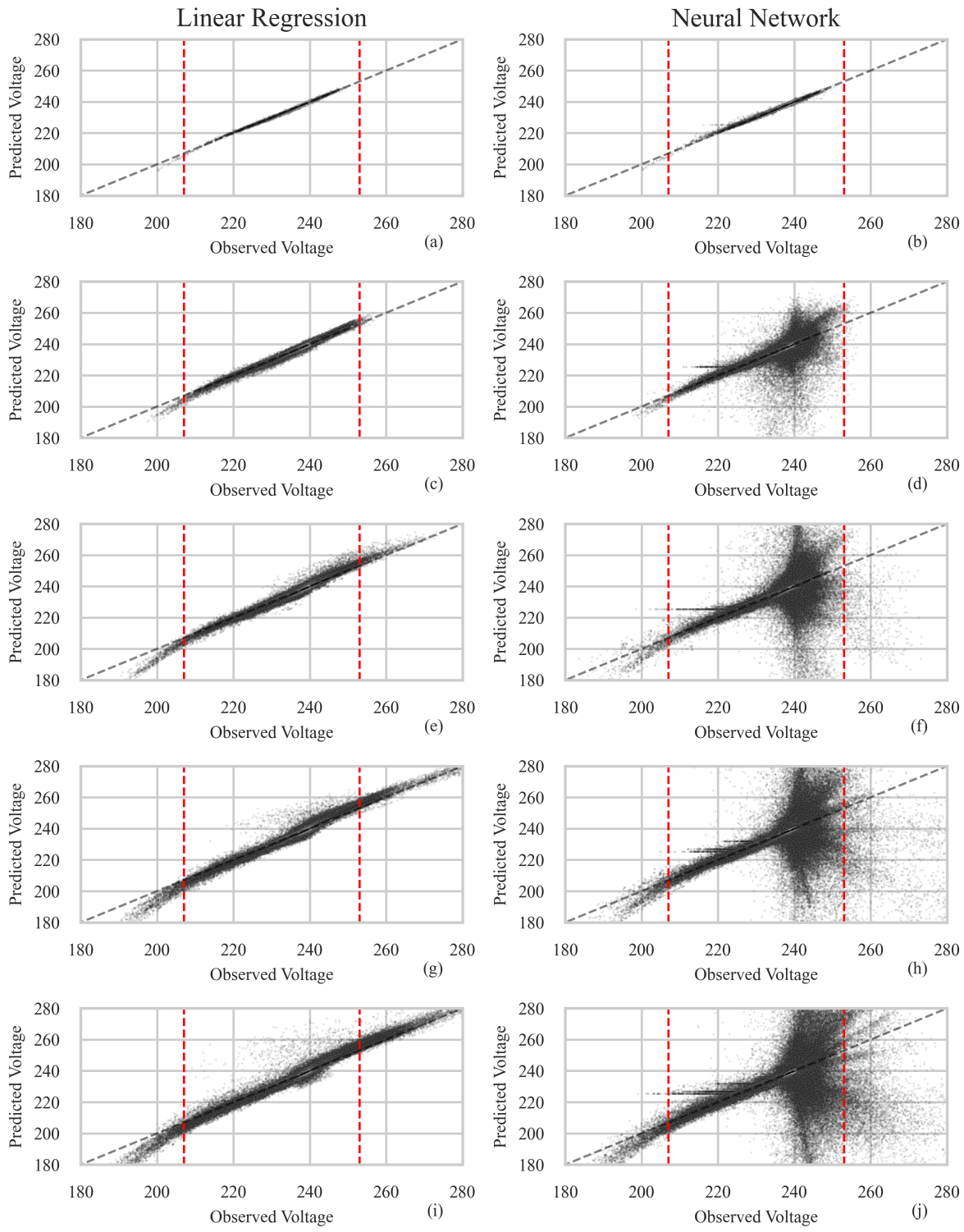


Figure 6.2: Predicted voltage vs electrical model voltage at (a,b) 10%, (c,d) 30%, (e,f) 50%, (g,h) 70% and (i,j) 100% PV and EV penetration. Each black dot is one prediction over 10 days for all feeders. The left column (a,c,e,g,i) are linear regression predictions and the right column (b,d,f,h,j) are neural network regression predictions. A perfect fit is shown with the black dotted line, the red dotted lines are the statutory voltage limits ($1 \pm 0.1 P.U.$).

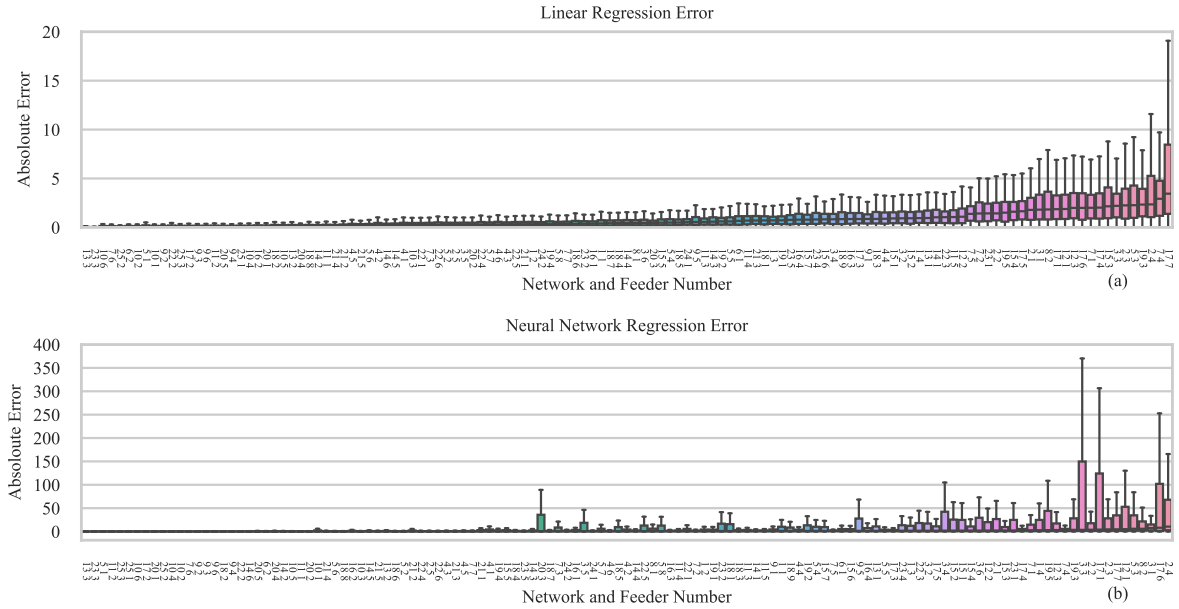


Figure 6.3: Distribution of absolute voltage error across all 127 feeders for the linear model (a) and the neural network (b) at 100% EV and PV penetration.

average mean absolute error (MAE), the average MAE plus three standard deviations and the maximum MAE metrics for both models aggregated across all 127 feeders.

Table 6.1: Mean Absolute Error

Model		**PV and EV Penetration Level**				
		10%	30%	50%	70%	100%
Linear Regression	Av. MAE	0.078V	0.345V	0.499V	0.662V	0.860V
	Av. MAE + 3σ	0.492V	1.868V	2.939V	4.027V	5.322V
	Max MAE	0.445V	1.756V	3.388V	4.571V	6.233V
Neural Network	Av. MAE	0.117V	1.129V	2.468V	3.887V	6.255V
	Av. MAE + 3σ	0.796V	11.990V	26.695V	43.733V	65.781V
	Max MAE	0.602V	11.605V	25.304V	52.093V	77.370V

7. Discussion

The discussion explores voltage prediction performance against electrical models grids with increasing EV and PV penetration. Linear models generally perform well but tend to under-predict voltages outside statutory limits. Despite this, they remain usable within constraints. Machine learning models offer predictive value, aiding in forecasting and identifying grid upgrade needs.

This project introduces an electrical model-free voltage calculation method tested across multiple feeders. Previous literature suggested regression models could replace electrical models but required electrical models and diverse training data. Bassi et al. demonstrated the adequacy of machine learning models trained on smart meter data. This dissertation further confirms the generalisability of machine learning methodologies across different feeders with acceptable performance.

7.1 Modelling Performance

Figure 6.1 shows the voltage for every feeder, over a 10 day period, at 10% and 100% EV and PV penetration. At 10% EV and PV penetration the average voltage is 237.8V and 238.1V at 100%, the voltage standard deviation increases from 2.8V to 8.05V. Many networks see voltages over the statutory voltage limit of 253V (1.1 p.u.) at 100% penetration. In reality the electrical model is not reliable outside of these ranges, for example, the PV inverters would detect this high voltage and stop injecting power into the grid, however, this generates worst case performance to test the models with.

Figure 6.2 compares the performance of the electrical models and the machine learning models as PV and EV adoption increase. A perfect fit is demonstrated by the black dashed line. When tested at 10% the linear and neural network regression models perform similarly, with the linear model performing marginally better, both models could be used interchangeably. At

higher penetration percentages and at voltages greater than 230V, the neural network models begin to diverge, feeder voltages are either severely underpredicted or overpredicted. This begins to happen even at 30% penetration. The difference between under and overprediction demonstrates the variance associated with training this type of model as it is very sensitive to the initial random parameters. The linear regression model performs preferably, it has a lower average MAE than the neural network at average penetration percentage tested. Even at higher penetration percentages, the model preforms well, however, it consistently underpredicts voltages outside of the statutory limits. The linear models can still be used with this restriction.

The linear model performs well with an average MAE of less than a volt across all 127 feeder, however, some feeder models begins to break down with the maximum MAE of 6.23V. Improved neural network performance could be found by searching more hyperparameters, however, a simpler model that performs similarly or better is often desirable and the linear model is a good candidate.

The machine learning models exhibited the best performance when evaluated with the same scenario on which they were trained, nevertheless, they still provide some predictive utility at higher PV and EV penetrations, the models may have long term utility. These can also be used to forecast voltages at higher LCT penetrations and can evaluate which feeders are most in need of upgrades.

7.2 Contribution to the Literature

The contribution of this project is that it is the first electrical model-free voltage calculation model that has been tested across multiple feeders. It has been shown that the approach used in this dissertation is robust and generalisable to much higher LCT penetrations. The neural network modelling pipeline used was insufficient and preformed poorly at PV and EV penetrations above what it was trained on. However, OLS linear regression successfully learned the dynamics of the feeders, it outperformed the neural network on average over all feeders at every LCT penetration percentage tested. It is possible that the neural network could be trained bet-

ter, with a better set of hyperparameters, trained for longer, with more layers or more neurons. It is computationally more expensive to train and linear regression models are well understood and in this case provides improved performance. A policymaker, distribution system operator or decision maker could understand, and trust a linear regression model.

In the literature it has been shown that regression models could replace electrical models, however, this has always needed an electrical model to generate training data that was diverse and numerous. Bassi et al. [6] successfully showed that an neural network based machine learning model was an adequate substitute for an electrical model when trained on smart meter data. This dissertation shows that a machine learning methodology is generalisable and can be blindly applied to many different feeders with acceptable performance.

The models and objectives differed between the methodology shown and In Bassi et al. every voltage across the network modelled using a neural network, with moth active and reactive power as inputs. It has been shown that a simplified version where the voltage at one node was modelled, the neural network did not constantly generalise to higher penetrations and the linear regression model preformed favourably. The underlying concept that smart meter data can be used to model a LV feeder has been verified.

Perhaps the neural network would work better if a greater range of hyperparameters were searched, or the data was preprocessed differently, but it is almost impossible to tell if the weights in a neural network are a global or local minimum.

7.3 Assumptions

This approach used is not without limitations, some assumption were made and these effect the results. In the electrical model, the reactive power profiles were generated from a random power factor between 0.9 and 0.98. Furthermore the LCTs were assumed to have a unity power factor. The low voltage feeders were connected to an infinite bus via a transformer, a more realistic approach would be to connect the feeders to an integrated MV feeder to model upstream effects.

This would add unmeasured voltage fluctuations to the end of the feeder which would result in reduced predictive accuracy. This can be mitigated if the transformer or the MV feeder was monitored.

The reactive power data was not included as an input to the machine learning models. It has an effect on the voltage, if reactive power varied more, perhaps the aggregate power factor, or reactive power could be introduced as a feature to improve predictive performance. If the reactive power was introduced it is likely still highly correlated with the active power and dominantly reduction techniques may prove beneficial to both linear regression and to neural network, neither of which properly handle highly correlated features.

The smart meter data used was simulated, if real smart meter data was obtained, a more realistic picture of what the models are capable of would emerge. However, much more data would be needed to verify the performance of these models as higher LCT penetrations cannot be tested.

The electrical model accurately predicts voltage within the statutory limits, outside of these limits, the voltage is undefined, for example the PV inverters tripping during high voltages is unmodelled.

The customer furthest from the transformer was chosen as the target voltage as it is likely to have voltage issues, however, it may not be the most extreme voltage, another phase may have twice the number of customers or a particular localised area may have a high density of PV in comparison to the rest of the feeder. To resolve this a separate model could be trained for every customer, a model that simultaneously predicts the voltage at every customer has been achieved in the literature.

The load data used lacked some variability there were only 100 daily load curves and some networks had more than 300 customers, the data is assigned randomly with replacement. Overfitting is not a concern as the training period was seven days and the neural network used was relatively small and trained with regularisation, it is unlikely that it memorised every load. Furthermore, more variability was added with the PV and EV loads. This can be tested with a summer load dataset.

The electrical model and machine learning model were time independent. In reality there are some time varying effects such as MV tap changers or control strategies. If these effects were present in the electrical model, a different modelling strategy would have to be employed. It is also possible that more effective machine learning techniques could have been used and a better combination of hyperparameters would have resulted in better models.

8. Conclusion

The deployment of LCTs in the home such as PV and EV charging puts an additional strain on LV grids. This can cause voltage regulation issues, as voltage drops below the statutory limit, or PV inverters tripping off the grid due to excessive voltage, affecting usage rates and efficiency. The ESB has installed more than 1.5 million smart meters in Ireland. ESB smart meters collect interval data on import and export active and reactive power, as well as voltage. A realistic smart meter dataset was created, and 127 real LV networks were simulated with electrical models. Machine learning models were trained on data with 10% EV and PV penetration, successfully replicating the electrical models with some error without prior knowledge of the grid or its topology. They were shown to be accurate and generalisable into a future with increasing LCT usage. They were demonstrated to be accurate and generalisable to a future with increased LCT adoption. The worst linear regression model had an MAE of 6.233V, although most models are generalisable to increased PV penetration, with the average MAE being less than a volt.

The modelling process was developed on network 1 feeder 1 and was then implemented and across 127 different LV Feeders. Realistic customer smart meter data was used in for the training and testing data, the machine learning models were then compared to an electrical model. Linear models and neural networks were trained on 127 different feeders with realistic smart meter data including 10% PV and EV penetration with 7 days of data at 5 minutes resolution. These models were then tested at higher PV and EV penetrations, and the neural networks diverged from each other while the linear models performed well and had an average error of less than 0.1V (0.043%) across all feeders. it also performed well at increasing PV and EV penetration levels, however it under-predicted extreme voltages outside the statutory limits. Every Feeder tested had a MAE of less than 6.24V (2.7%) at 100% EV and PV penetration.

A remote voltage calculation model was developed that could calculate the voltage at the customer furthest from the feeder from smart meter data. However, any node of interest could be modelled using this methodology this was shown to be reliable within the statutory voltage

limits, however it began to break down for some feeders at higher PV and EV penetrations. The most effective machine learning model used was linear regression, this performed best when tested on the penetration percentage that it was tested on. It was found that preprocessing had very little impact on model results. The linear model not only performs well, it also generalises well to higher LCT penetration levels, suggesting the model can be used for a long time, it can also be used for a LCT hosting capacity study and rank the feeders that are most susceptible to voltage violations that may be in need of upgrades in the future. The error rose as PV and EV penetration increased, however, they can still be used keeping this restriction in mind.

The linear regression model could be trained and used at scale to model any LV network that has smart meters, this would circumvent the expensive and time consuming process of producing electrical models and allow DSOs to model the grid at an unprecedented scale. Furthermore, no information on location or topology is needed allowing for the use of anonymised smart meter data.

These models could be used to asses which LV networks are most in need of upgrades and can conduct hosting capacity studies for any LCTS. This can also be used to test LCT control strategies.

This electrical models used made some assumptions and simplifications which could be addressed in future work. One could integrate an MV feeder and then train and test all the feeders at scale. Real customer smart meter data could be used to obtain a realistic view on what these data-driven models are capable of and to see how these models are applicable into the future. These models could be compared against electrical models for LCT hosting capacity studies. Many more LCTs could be easily integrated and their effects tested. One could also predict transformer current or any other quantity of interest.

There is a large scope for future work, the electrical model used could be improved to further test these models. This can be achieved by integrating an upstream MV network, modelling the LCT load more accurately and by adding more LCTs. The modelling process can be improved or extended a model could be generated for every customer on a network, or a model with a customer number of outputs could be produced. The machine learning framework can be

further optimised, more models can be tested and training can be improved. Linear regression worked best, other linear models and weak learners could be tested. Data can be extracted and interpreted from the linear regression weights which would allow for more detailed hosting capacity studies.

The ESB smart meters collect more information than just active and reactive power consumption and generation, this can be further leveraged. Each smart mater can also be linked to X-Y coordinates, perhaps an electrical model could be produced by generating connectivity information from the smart meter data. If DSO had access to 1.5 million smart meters how is that divided into discrete feeders and networks.

References

[1] A. Navarro-Espinosa and L. Ochoa, "Dissemination document "low voltage networks models and low carbon technology profiles"," *Ph.D. thesis The University of Manchester*, vol. 44, 2015.

[2] ESB Group, "ESB 2023 Annual Report and Financial Statements," 2023. Accessed on: April 19, 2024.

[3] ESB Networks, "Data protection for the national smart metering project," 2022. Accessed on: April 19, 2024.

[4] ESB Networks, "Smart metering data - data protection impact assessment," 2022. Accessed on: April 19, 2024.

[5] R. Dugan, "Reference guide: The open distribution system simulator," 2013.

[6] V. Bassi, L. F. Ochoa, T. Alpcan, and C. Leckie, "Electrical model-free voltage calculations using neural networks and smart meter data," *IEEE Transactions on Smart Grid*, vol. 14, 2023.

[7] M. Ferdowsi, A. Benigni, A. Lowen, B. Zargar, A. Monti, and F. Ponci, "A scalable data-driven monitoring approach for distribution systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, pp. 1292–1305, 5 2015.

[8] S. Balduin, T. Westermann, and E. Puiutta, "Evaluating different machine learning techniques as surrogate for low voltage grids," *Energy Informatics*, vol. 3, p. 24, 10 2020.

[9] Z. Liu, J. Ringelstein, M. Ernst, B. Requardt, E. Zauner, K. Baumbush, S. W. von Berg, and M. Braun, "Monitoring of low-voltage grids using artificial neural networks and its field test application based on the beedip-platform," 2023.

[10] J. A. Azzolini, M. J. Reno, J. Yusuf, S. Talkington, and S. Grijalva, "Calculating pv hosting capacity in low-voltage secondary networks using only smart meter data," in *2023 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, 2023.

[11] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[12] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2018.

[13] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[14] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[16] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, (Cambridge, MA, USA), p. 155–161, MIT Press, 1996.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[19] M. Kuhn and K. Johnson, *Feature engineering and selection: A practical approach for predictive models*. 2019.

[20] A. M. Molinaro, R. Simon, and R. M. Pfeiffer, "Prediction error estimation: a comparison of resampling methods," *Bioinformatics*, vol. 21, pp. 3301–3307, 05 2005.

[21] J. Kim, "Multicollinearity and misleading statistical results," *Korean Journal of Anesthesiology*, vol. 72, 07 2019.

[22] I. Yeo and R. A. Johnson, "A new family of power transformations to improve normality or symmetry," *Biometrika*, vol. 87, pp. 954–959, 12 2000.

[23] P. Meira, "Dss extensions, github repository," Acesssed: 01/12/2023.

[24] P. S. Foundation, "Python language reference," Acesssed: 01/12/2023.

[25] J.-H. Menke, N. Bornhorst, and M. Braun, "Distribution system monitoring for smart power grids with distributed generation using artificial neural networks," *International Journal of Electrical Power & Energy Systems*, vol. 113, pp. 472–480, 2019.

[26] I. Sengor, L. Mehigan, M. A. Zehir, J. J. Cuenca, C. Geaney, and B. P. Hayes, "Voltage constraint-oriented management of low carbon technologies in a large-scale distribution network," *Journal of Cleaner Production*, vol. 408, p. 137160, 2023.

[27] L. Ochoa, "OpenDSS training material," 2022. Accessed on: April 19, 2024.

[28] T. H. B. Huy, D. N. Vo, H. D. Nguyen, H. P. Truong, K. T. Dang, and K. H. Truong, "Enhanced power system state estimation using machine learning algorithms," in *2023 International Conference on System Science and Engineering (ICSSE)*, pp. 401–405, 2023.

[29] S. Pawar and B. F. Momin, "Smart electricity meter data analytics: A brief review," in *2017 IEEE Region 10 Symposium (TENSYMP)*, pp. 1–5, 2017.

A. Source Code

Train and test linear regression and neural network with hyperparameter tuning.

```
regressor = TransformedTargetRegressor(
    regressor = linear_model.LinearRegression(),
    transformer = StandardScaler(with_mean = True, with_std = True)
    )
pipeline = Pipeline(
    steps = [
        ("preprocess_cs",StandardScaler(with_mean = True, with_std = True)),
        ("target_transform", regressor)
    ])
# Split 10% data
X_train, X_test, y_train, y_test = train_test_split(Dataset_10, 230*Targets_10, test_siz

# Fit Lienar Model
pipeline.fit(X_train, y_train)

# Test Linear Model
preds_linear_10 = pipeline.predict(X_test)
preds_linear_30 = pipeline.predict(Dataset_30)
preds_linear_50 = pipeline.predict(Dataset_50)
preds_linear_70 = pipeline.predict(Dataset_70)
preds_linear_100 = pipeline.predict(Dataset_100)

# Edit model type
 pipeline.set_params(target_transform__regressor = MLPRegressor())
 # NN grid search
 param_grid = {
        'target_transform__regressor__learning_rate_init': np.linspace(0.001, 0.1, 20),
        'target_transform__regressor__alpha': np.linspace(0.00001, 0.001, 20),
        'target_transform__regressor__hidden_layer_sizes': [[25,], [50,],[100,],[150,]]
        }

grid_search_nn = GridSearchCV(pipeline, param_grid, cv=RepeatedKFold(n_splits=5, n_repea
grid_search_nn.fit(X_train, y_train)
NN_optimal_params = grid_search_nn.best_params_

# Set params to optimal found
pipeline.set_params(target_transform__regressor__alpha = NN_optimal_params["target_trans
pipeline.set_params(target_transform__regressor__hidden_layer_sizes = NN_optimal_params[
pipeline.set_params(target_transform__regressor__learning_rate_init = NN_optimal_params[

# Fit model
pipeline.fit(X_train, y_train)

# Test Model
preds_nn_10 = pipeline.predict(X_test)
preds_nn_30 = pipeline.predict(Dataset_30)
preds_nn_50 = pipeline.predict(Dataset_50)
preds_nn_70 = pipeline.predict(Dataset_70)
preds_nn_100 = pipeline.predict(Dataset_100)
```

B. Logbook

29September 2023

Reading

[9] This paper concerned predicting the unknown bus voltages and line transformer loading's on an under-defined LV network. This was tested on 2 real grids in Braunschweig, Germany where the grid topology is known. The model used was an artificial neural network (ANN) with 1 input, 1 output and 3 hidden layers. This model worked well ($<$8% error) during normal operation, however it performed poorly during low grid loading.

[25] An ANN was constructed to predict voltage magnitudes and line current magnitudes for a MV grid where the topology is known. The authors main contribution to the state of the art was to use a scenario generator to train the ANN, this used an AC power flow simulation with different switch states (which change topology) and varied load scenarios to provide a diverse data-set for the model. This was tested on a real network with an error of less than 10%.

[26] A method for managing voltage constraint LV/MV networks to expand their hosting capacity is proposed. This involves the curtailing of distributed generation(DG) and shifting of residential EV charging, BESS management with minimal impact to end users. This is simulated using OpenDSS and Matlab.

[6] An ANN was designed to predict voltage from a smart meter data-set generated from a realistic model of a LV network. This model uses 2 input layers, 2 hidden layers an and output layer. The first input consists of active (P) and reactive (Q) power for each customer in the data-set and the second input is the aggregated P and Q. The authors propose a grid search for hyper-parameters which could be used to train a model on a real data-set. The model generalises well to higher penetration of PV.

[19]This book details techniques and best practices that are useful before a machine learning model is trained, with particular emphasis on designing and finding the best representation of predictors. It also discusses the bias variance trade off in detail.

Achievements

- Used OpenDSS to simulate a test network and dump the node voltages in CSV files.
- Read chapter 1 and 2 of [19]

Goals for next week

- Follow an OpenDSS tutorial [27] with the goal of running an AC power flow at different time steps and exporting the node data as a CSV.These data will be read by Python and used to develop a signal processing and feature engineering pipeline.
- Continue to read [19]

06 October 2023

Reading

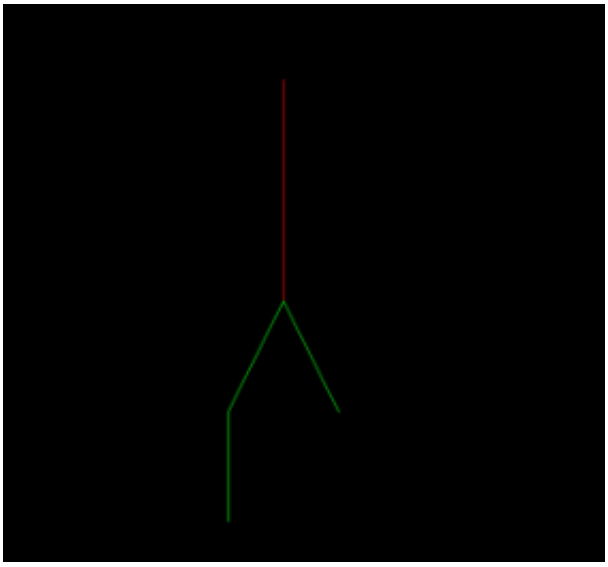
- Read chapters 3-4 in [19], A Review of the Predictive Modeling Process, and Exploratory Visualisations
- [28]This paper trains an ANN, a random forest and an XGBoost model and compares the results in a time series state estimation problem. The authors use simulation data generated from the IEEE 14 and 30 bus systems. The dataset contains 39444 samples of size 64 and 110 data points each. The states are represented by the voltage magnitudes and angles at all buses. All three machine learning algorithms show high accuracy at estimating the states of the system.
- [29] This paper is a review of smart meter technology and machine learning tools. It covers SVMs and clustering for classification. A Self Organising Map (SOM) uses unsupervised learning to reduce the dimensionality of the data and has been used for load profiling, load forecasting and customer segmentation.

Achievements

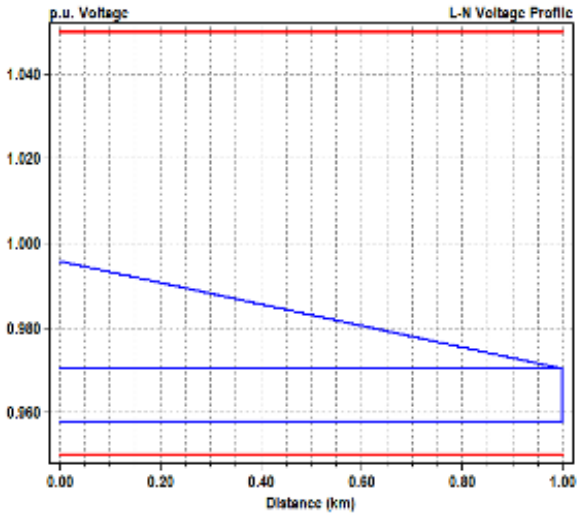
- Loaded a model into OpenDSS and ran an AC power Flow, graphs pictured below.
- Began to follow OpenDSS tutorial

Goals for next week

- Continue the OpenDSS tutorial with the goal of interfacing python with OpenDSS using the COM interface.
- Continue to read [19]



Voltage profile for the same network



Voltage profile for the same network

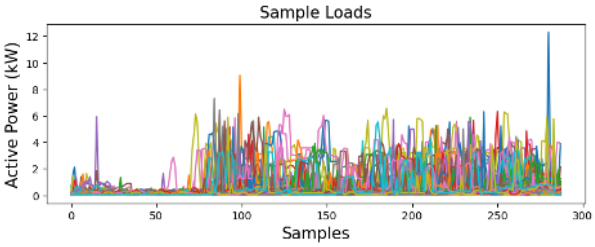
13 October 2023

Reading

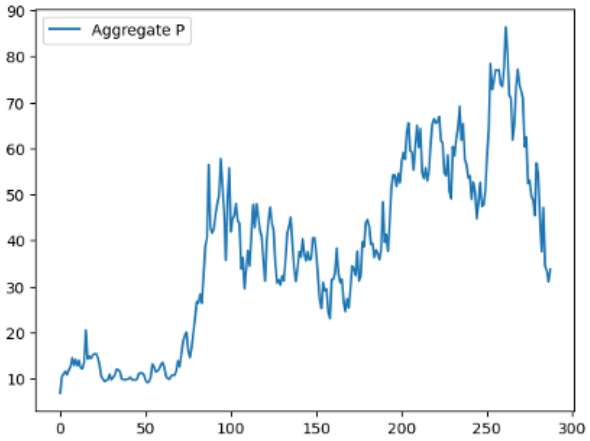
[19] Chapters 5 and 6 concerned encoding categorical predictors and engineering numerical predictors. [1] This paper is the dissemination document for 25 LV Networks Models each with several feeders each as well as 100 LCT and realistic customer loads. It details how each of the datasets were generated and how they might be used.

Achievements

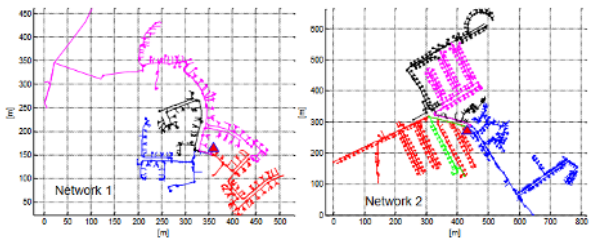
- Loaded LV network and load profiles from [1] into Python
- The load profiles include several low carbon technologies and have a 5 min resolution over 24 hours.
- Set up OpenDSS to through the Python environment
- Began randomly assigning load profiles to each customer
- Created set of reactive power from active power data and random power factors uniformly distributed between 0.9 and 0.98



100 customer loads in a 24 hour period



Aggregate customer load



Examples of the topology of 2 networks from [1]

Goals for next week

- Run full AC power flow for an LV Feeder
- Generate smart meter data from the OpenDSS AC power flow
- Begin feature engineering
- Continue to read [19]

20 October 2023

I was out sick

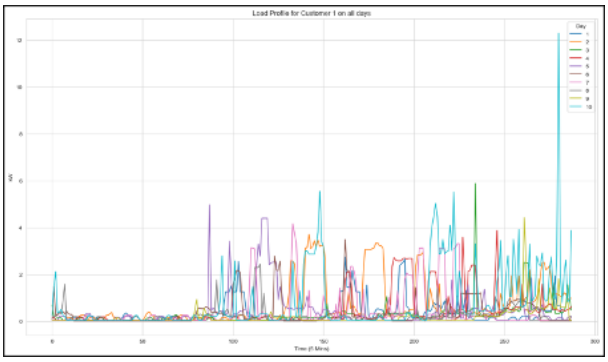
27 October 2023

I was out sick

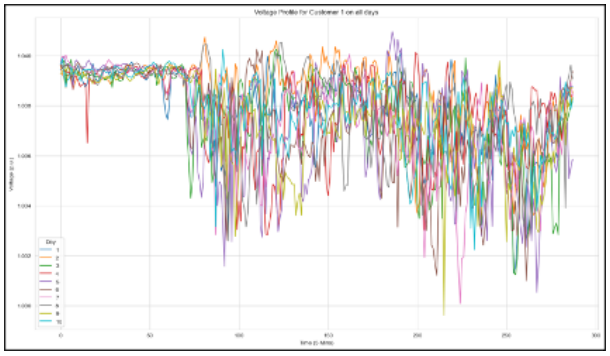
03 November 2023

Achievements

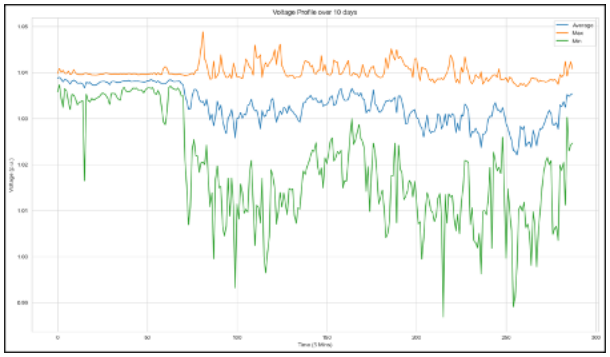
- Ran AC power flow on Network 1 Feeder 1 over 10 days with random load profiles and reactive load profiles for 55 customers.
- Plotted results



Load profiles assigned to Load1 over 10 days

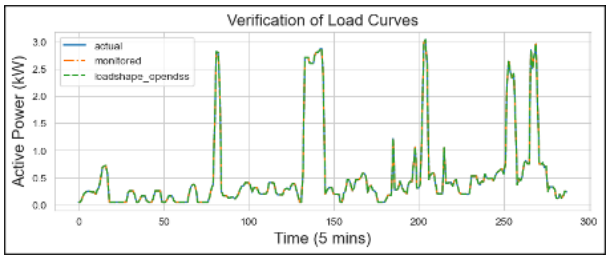


Voltage at Load 1 over 10 days

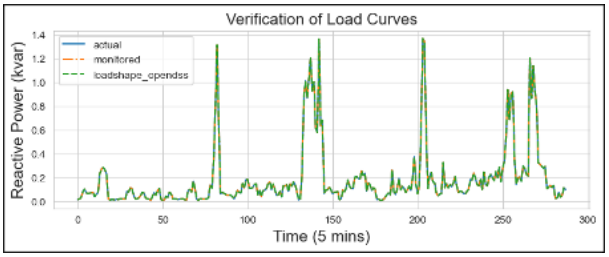


Average, max and min voltage achieved for all loads over 10 days

Voltage plot for customer 1, 16 and 55 on day 1.



Load profile pulled from OpenDSS (green), monitored load profile (orange) and load profile given to OpenDSS (blue).



Reactive power pulled from OpenDSS (green), monitored load profile (orange) and load profile given to OpenDSS (blue).

Goals for next week

- Arrange data into a matrix and split into test and train sets.
- Train a simple neural-network
- Begin feature engineering
- Possibly improve simulation by adding a MV network to the LV model

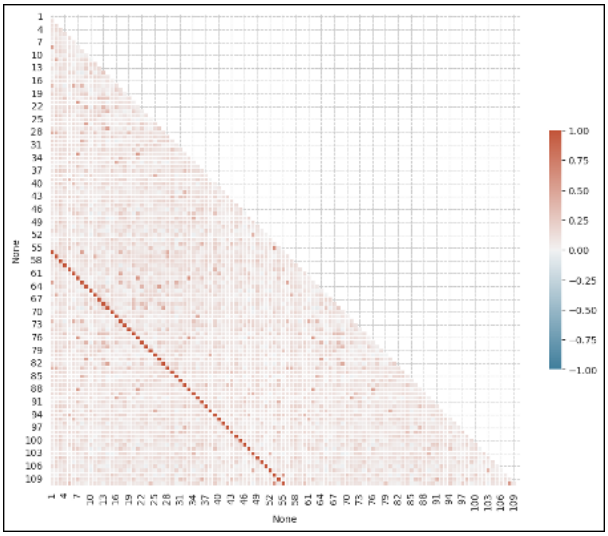
10 November 2023

Reading

Continued reading [19].

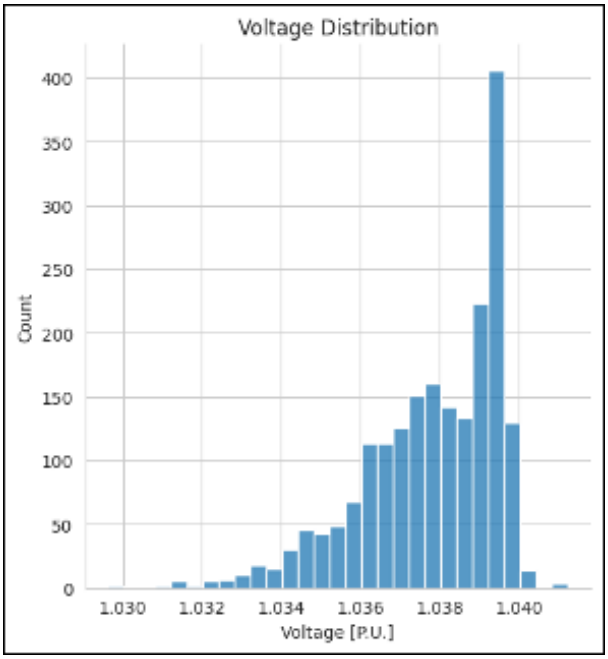
Achievements

- Ran AC power flow on Network 1 Feeder 1 over 10 days with random load profiles and reactive load profiles for 55 customers.
- Corrected error and replotted graphs.
- Began exploratory data analysis



Correlation matrix for P and Qs, the first 55 indexes are P the indexes 55-111 are Q. P is highly correlated to the Q at the same load.

Plot of P and Q for load 1. As is evident these are highly correlated.



Distribution of voltages in the training set

Goals for next week

- Continue exploratory data analysis.
- Set up resampling routine in test set for testing changes to models and predictors while avoiding overfitting.
- Investigate PCA for reducing the dimensionality of the data
- Transformation of predictors into more useful forms (Power Factor, etc.)
- Explore different models

17 November 2023

Reading

[19] Continued reading

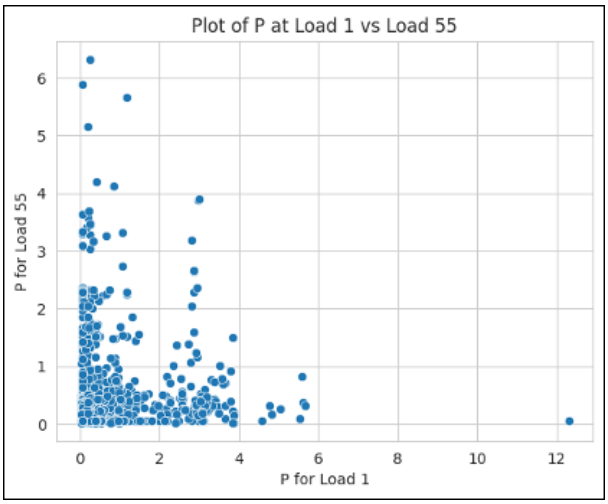
Achievements

- Investigated Principal Component Analysis[PCA] and kernel PCA for use in reducing the dimensionality of the data. As the data is weakly correlated, this method was suspected to be an improved method for

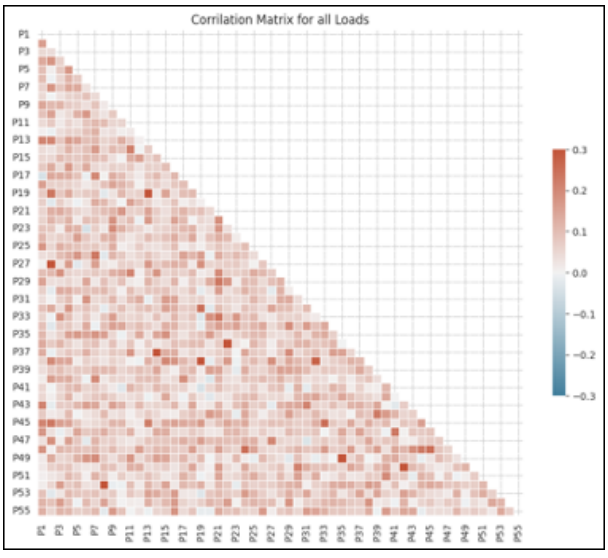
representing the data.

- The distributions were normalised using the Yeo-Johnson power transform, after which they were centered about 0 and scaled so that the standard deviation is 1. This is important for many models.
- After some consideration, Q was removed as a feature to simplify analysis. The power factor in residential grids is expected to be quite high and the relevance of Q may not be very important. It may be reconsidered in the future.

Correlation

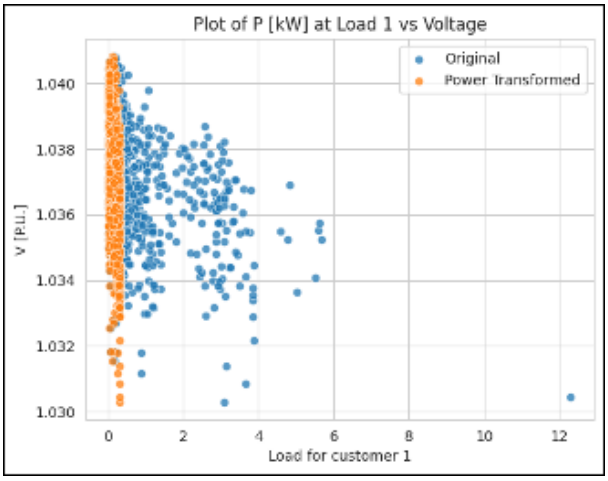


Plot of P at Load 1 against Load 55. They are weakly correlated: r = 0.12

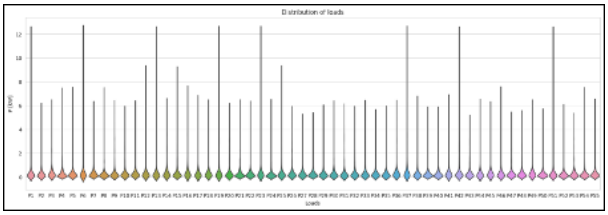


Correlation matrix of the loads, they are very weakly correlated, $|r| > 0.3$, 8 times

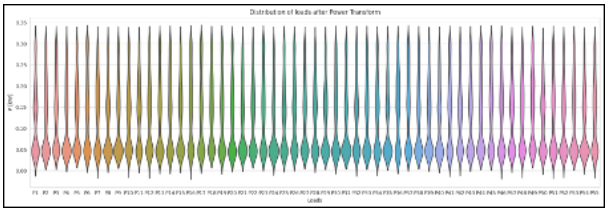
Distributions



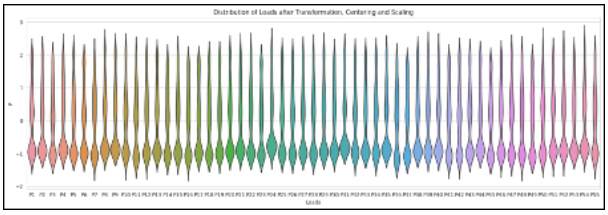
Scatter plot of Power at customer 1 before and after Yeo–Johnson power transform



Distribution of power at loads



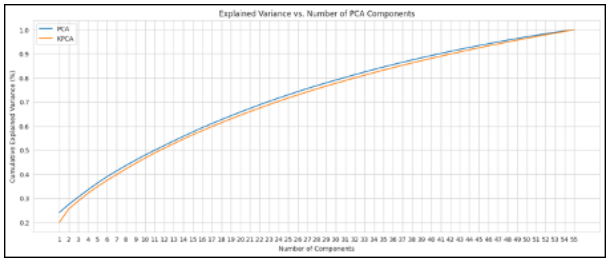
Distribution of power at loads after each has been power transformed



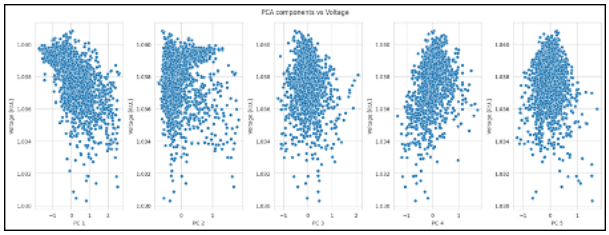
Distribution of power at loads after each has been power transformed centered at 0 and scaled to have a standard deviation of 0

Principal Component Analysis

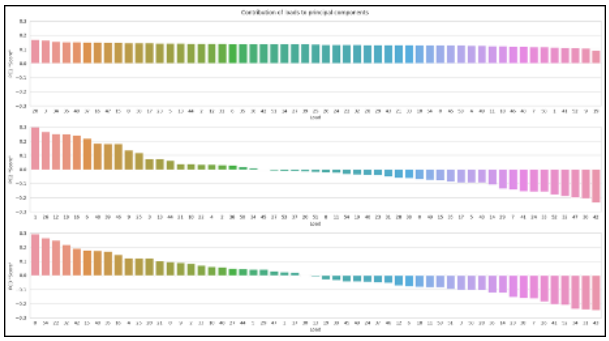
It is worth noting that PCA is unsupervised and there isn't a large amount of correlation between predictors. This method seems to have its limitations with regards to this application. This is also a distinctly different step from feature engineering.



PCA and polynomial kPCA were performed. Percent of variance explained with increasing numbers of components. 80% of the variance can be explained with 30 components.



Distribution of the first 5 Principal Components against voltage



Contribution of each load to the first 3 components of the PCA. The 1st component accounts for 20% of the variance and is a fairly evenly distributed weighted combination of all the features.

Goals for next week

- Continue exploratory data analysis.
- Set up K-Fold re-sampling routine in test set for Power Transform, Centering and Scaling. The current transformations use the entire training set.
- Investigate feature engineering.

- Investigate supervised formats for reducing the dimensionality of the data.
- Begin exploring different models.

21 November 2023

Worked on interim report

1 December 2023

Worked on interim report and presentation

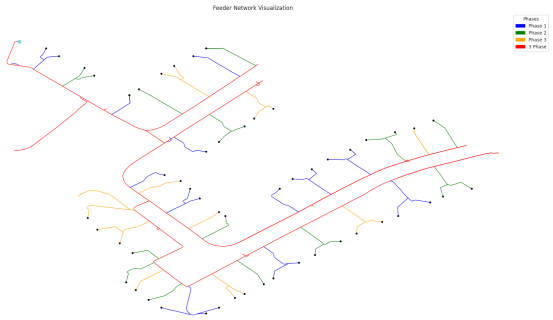
20 January 2024

Prepared and presented the interim presentation.

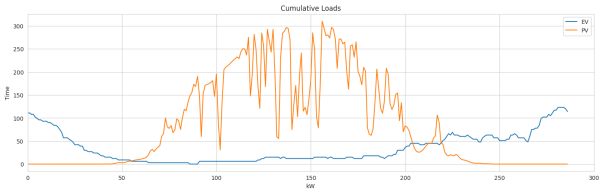
27 January 2024

Achievements

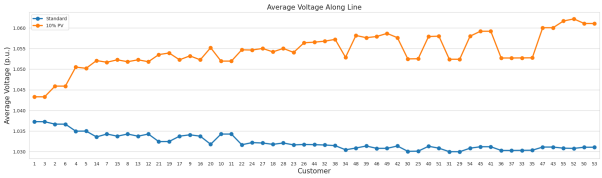
- Read the LV network files using Python and NetworkX to implement graph representations of the networks. This representation maintained phase information and encoded the length of each edge. These graphs were used to:
  - Plot the networks in xy space to aid visualisation.
  - Calculate the load that has the longest shortest electrical length to the transformer, these will be the voltages of interest for prediction as they are most susceptible to low (and high using PV) voltages.
- The electrical simulation was run and the voltages of interest were used as the target for the machine learning algorithms.
- The simulation was expanded to include random allocations of PV and EV loads. The penetration percentage can be changed from 0-100%. This currently has an issue where different customers are selected each day for EV and PV allocation. The EV loads are all 3kW loads.
- Added optionality to use Summer or Winter load datasets.



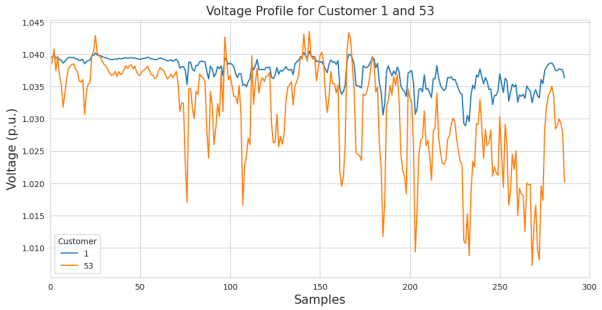
Plot of Network 1 Feeder 1. The red line represents a 3 phase line. Each load is represented with a black dot. The sky blue dot is the transformer.



Cumulative EV and PV loads.



Average voltage along line on day 1 sorted by distance from transformer of 10% PV penetration and 0% PV penetration.



Voltage at closest and furthest customer from transformer at 0% PV and EV penetration using winter dataset.

Goals for next week

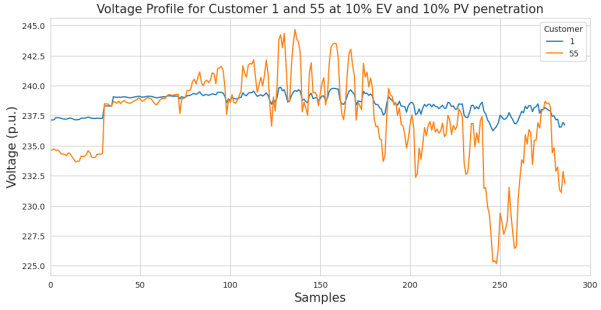
- Troubleshoot EV loads, they are not handled properly and do not introduce a load
- Include EV and PV loads in ML training data (usure if the PV loads should be subtracted)

- Introduce other LCTs
- Return to ML

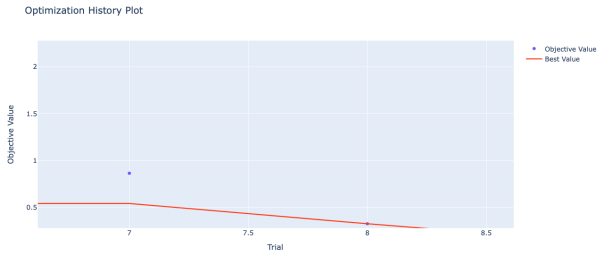
4 February 2024

Achievements

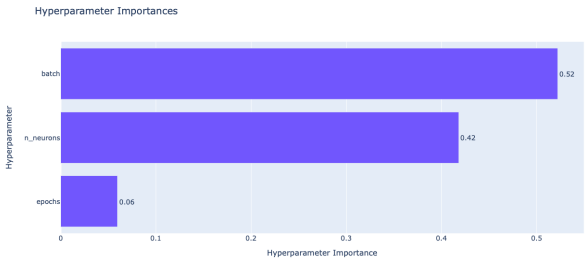
- Fully implemented the PV and EV loads.
- Replaced the EV loads with 6kW loads
- Exported all network load and voltage data
- Transformed load data (within resampling) to allow for faster training and hopefully less biased results
- Fixed issue with the neural network where there were more neurons than expected
- Set up greedy parameter search to optimise network parameters hopefully faster than grid search.
- RMSE on the furthest customer with 10% PV and EV penetration is about 1V



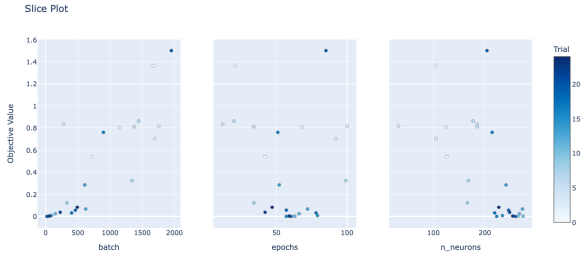
Plot of voltage at the customer 1 and 55 in Network 1 Feeder 1. There is a clear rise during the day due to PV loads and a clear drop off during the night due to EV loads.



Optimisation of training RSME over trials



Estimated importance of hyper parameters



Enter Caption

Goals for next week

- Improve regression.
- Implement other non linear regression techniques.
- Save best model found by parameter search
- Introduce Q?
- Recalculate RSME from voltage predictions. validation RMSE does not work anymore as it is during training and the voltages need to be uncentered and scaled and converted to 230V.

11 February 2024

Reading

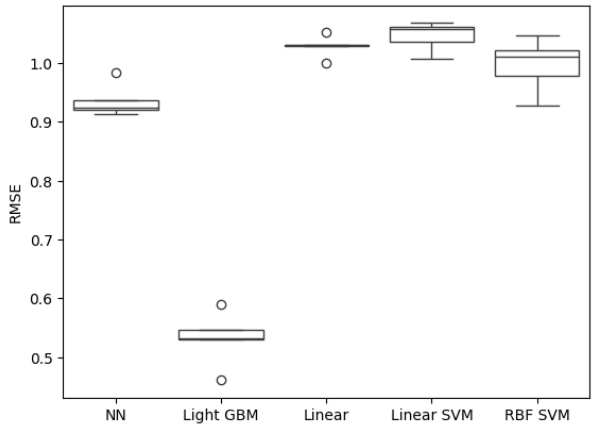
[12] Read chapters 1-7 of Applied Predictive Modelling. This book details the modelling process and it covers how linear, non-linear and tree based models function. The book also discusses the approach that should be taken to produce reliable models. The data flow is as follows: split into test and train set. Use 5 repetitions of 10 fold resampling to train models and optimise their hyperparameters. With the best set of hyperparameters the different models are trained using 5 repetitions of 10 fold resampling and the model with the lowest RMSE is chosen. This model is then trained on the whole training set using the best hyper parameters and tested on the test set. This resampling structure is vital with neural networks as results heavily depend on their initial weights.

Achievements

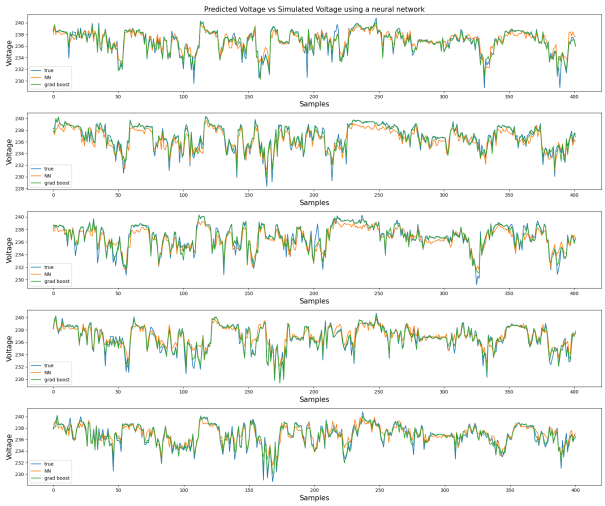
- Implemented repeated resampling.
- Trained Neural Network, Linear Regression, Light GBM, Linear SVM and RBF SVM.
- Plotted predicted against observed voltage.
- Plotted Model residuals (Voltage Error).

Note: It is worth mentioning that these models use the default hyperparameters for each of these models and optimising this will make a large difference to performance.

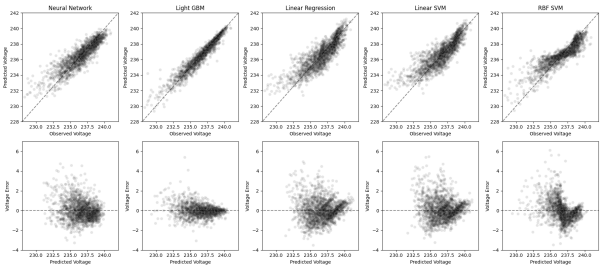
The plots only used 5 fold resampling with no repetition for performance reasons.



Box plot of RMSE for each model type. This was performed over 5 fold cross validation for performance reasons. All models have default hyperparameters except the NN which has 1 hidden layer and was only run for 50 epochs. The light gradient boosting machine performs the best by a wide margin. How much better it is and how statistically significant it is can be calculated.



Voltage for as predicted over 5 folds. Other results were excluded for clarity.



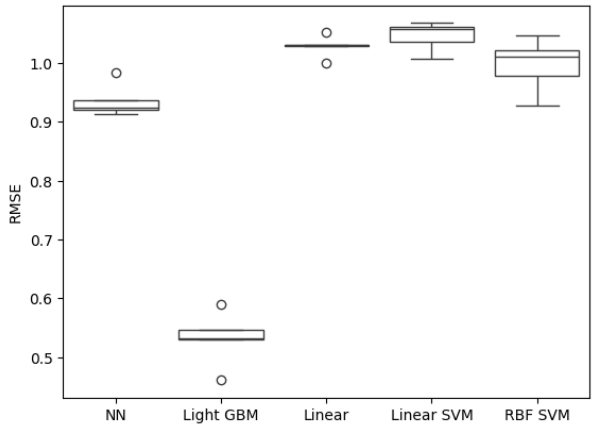
Predicted vs observed voltage and model residuals vs predicted voltage for each model type. This plot helps find systematic bias (e.g. Linear regression overestimates low voltages)

- Set up hyperparameter search for each model within the resampling structure.
- After running save models/hyperparameters.
- Automate choice of the best models and hyperparameters using the one-standard-error rule to find the simplest model is within one standard error of the model with smallest absolute error
- Set up the modelling file of the power lab computer
- Add more non-linear and tree based models. (As the other linear regression models (PLS, etc.) usually improve linear regression for cases with collinearity it is expected that they won't improve performance)

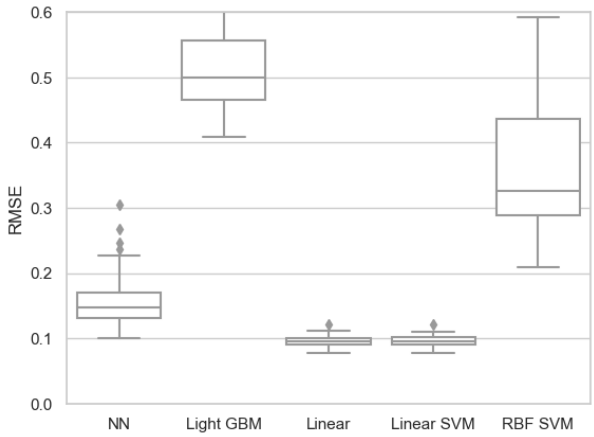
20 February 2024

Achievements

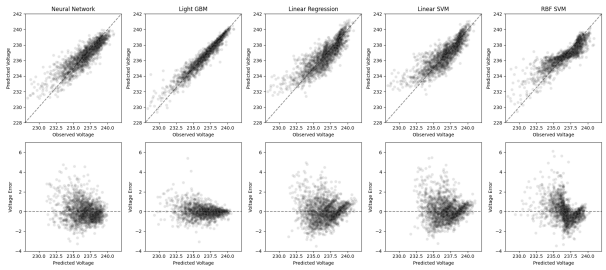
- Pipelined the modeling process
  - Allows for parallelisation
  - Significantly reduces code complexity
- Performed repeated refold hyper parameter tuning
- Some of the preprocessing steps made most models worse (except lightGBM)



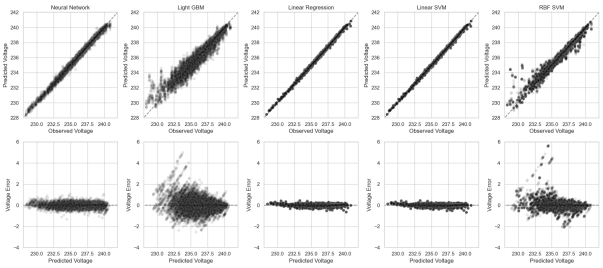
Box plot of RMSE for each model type with suboptimal parameters and power transform and centering and scaling the input data.



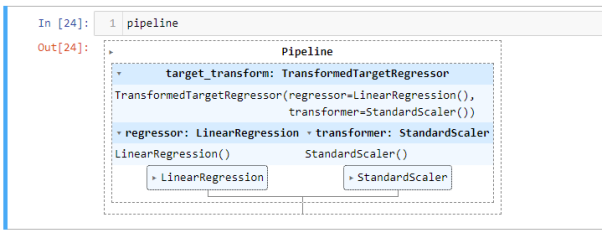
Performance with optimal parameters and centering and scaling WITHOUT power transform



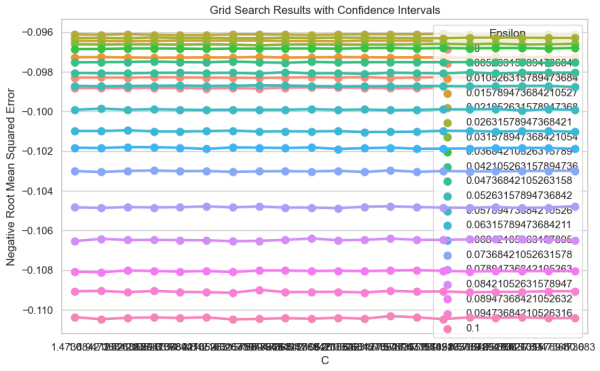
Predicted vs observed voltage and model residuals vs predicted voltage for each model type with suboptimal parameters and transformation with centering and scaling.



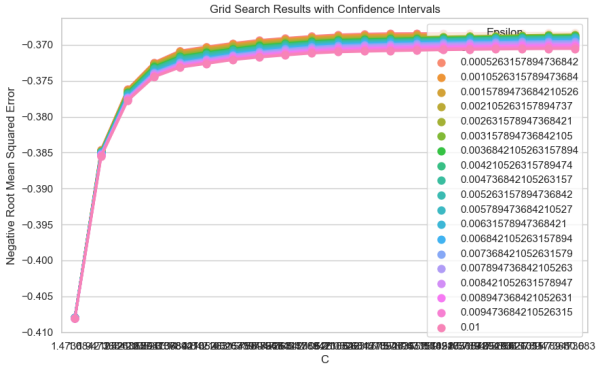
Predicted vs observed voltage and model residuals vs predicted voltage for each model type with optimal parameters and without power transformation with centering and scaling and 5 time repeated 10 fold resampling



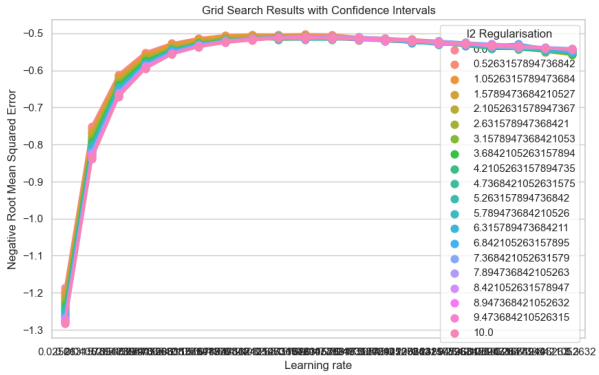
Pipeline structure



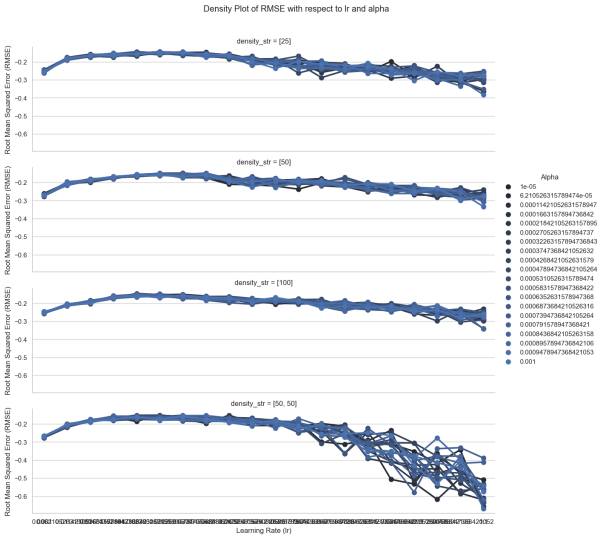
Linear SVM hyperparameter tuning (Higher is better)



RBF SVM hyperparameter tuning



Histogram gradient boosting hyper parameter tuning



Neural network hyperparameter tuning

- After running save models/hyperparameters.
- Automate choice of the best models and hyperparameters using the one-standard-error rule to find the simplest model is within one standard error of the model with smallest absolute error
- Investigate feature engineering's effect on performance
- Start training best model on all networks and feeders
- Generate test set for each feeder with higher PV/EV penetration or other interesting parameters

23 February 2024

Achievements

- Used OpenDSS to simulate and store the voltages and load information for every feeder in every network (total 128).

- Trained linear regression on 7 days worth of data and tested on 3 days for every feeder and stored the results.
- The worst performing model was that of Network 13 Feeder 4 (Fig. 1). At 100% EV and PV penetration the voltage from OpenDSS is in excess of 10,000,000 V, which is not remotely the case for the other networks. Upon further investigation this electrical model has 4 unique load positions and 173 loads.

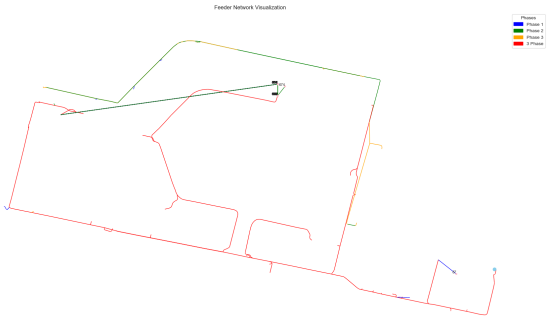
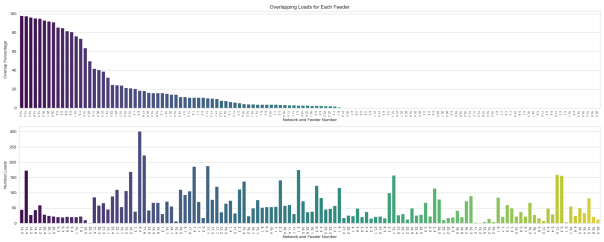
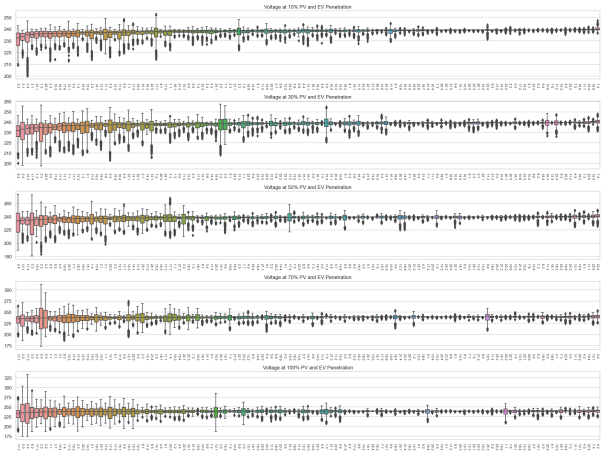


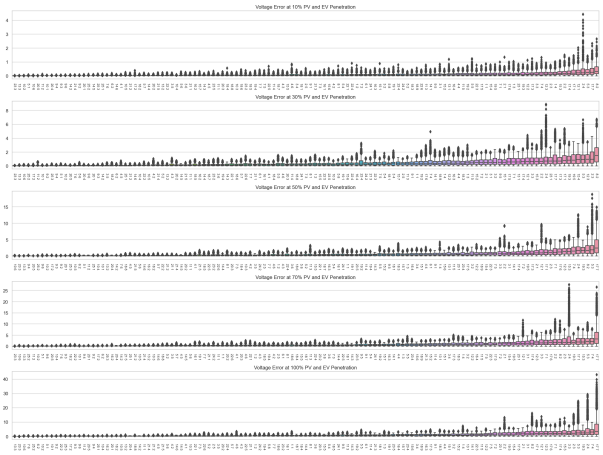
Image of Network 13 Feeder 4



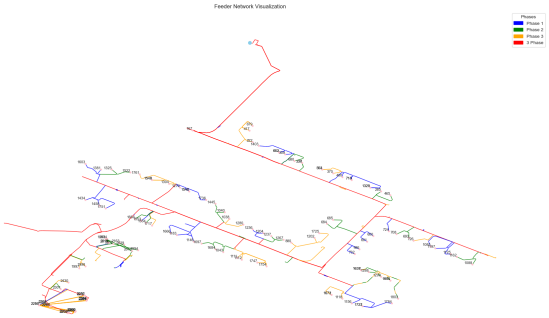
Bar chart of overlap percentages per feeder and load count per feeder. 13, 4 is the second most overlapped feeder by percentage, however, the feeder with the most overlap performs well.



Voltage from electrical simulation for every feeder once feeder 13,4 has been removed.



Voltage error using linear regression across different feeders with 13, 4 removed.



Topology of 17,7 which is the second worst performing linear model after 13,4.

Goals for next week

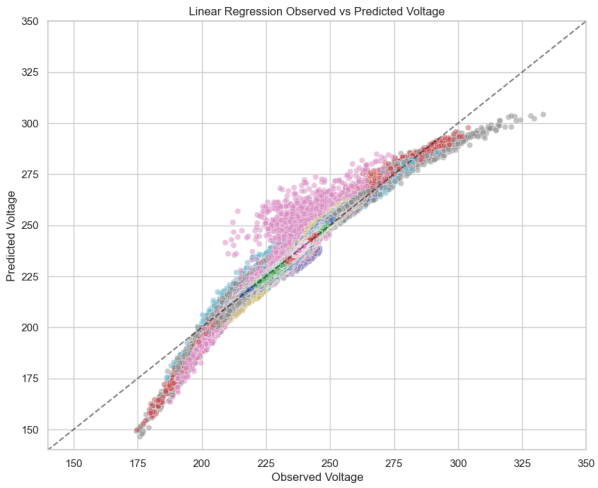
- Investigate performance against different network parameters
  - Number of loads
  - Distance of furthest load
  - Connectivity
  - Spread
- Implement other modelling techniques
- Investigate effect of preprocessing on each model type

01 March 2024

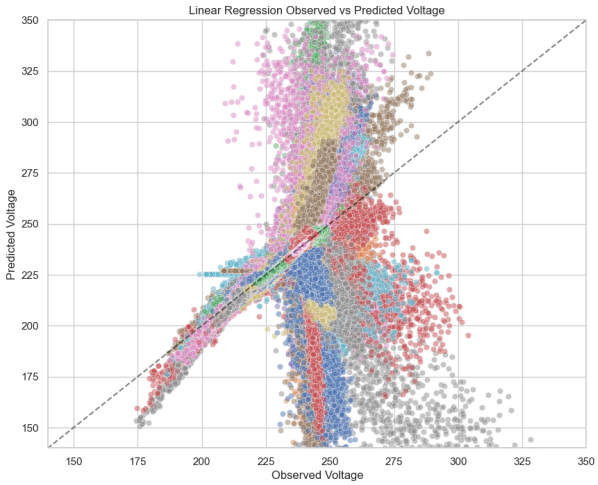
Achievements

- Set up cross-fold validation for hyperparameter tuning and trained NNs on all 128 feeders to significantly worse results than the linear model. Linear Regression takes seconds to run on all feeders, while the neutral network had to train for a day and a half.

Box plot of neural network errors.



Linear Regression observed voltage vs predicted voltage. There is a systematic bias toward underpredicting voltage at the high and low voltage levels.



Observed voltage vs predicted voltage for the neural network. At lower voltages the predictions are alright, but at higher voltages the predictions diverge. This shows the variance in neural network training.

Goals for next week

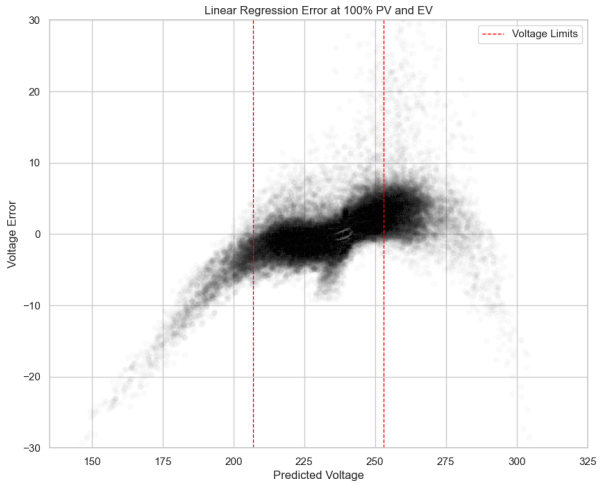
- Investigate performance against different network parameters
- Investigate effect of preprocessing on each model type
- Begin to write conference paper

08 March 2024

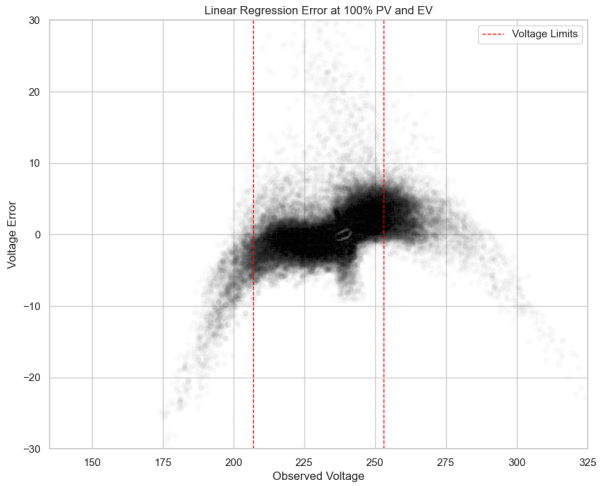
Achievements

- Began to write conference paper
- Looked at plotting in more detail

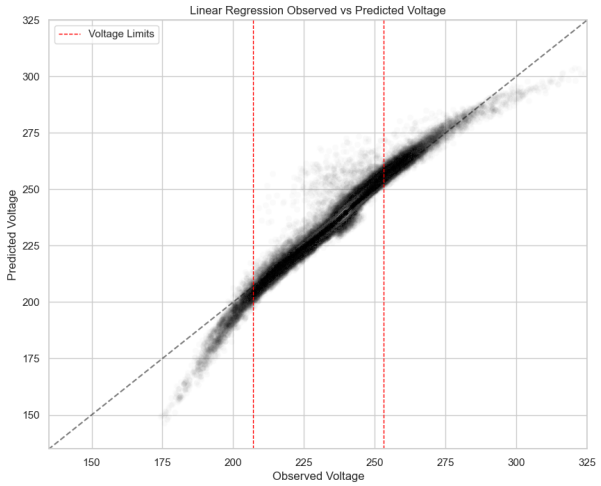
We have options for plots that show the same thing slightly differently, I'm only including the linear model for simplicity. These are the results across all 127 feeders. I like Fig.3 the most. The voltage limits are set to 230v ± 10%.



This is the predicted error vs the voltage error. In essence: “If the model predicts X volts what is the error like?”



This is the “true” voltage vs the voltage error. in essence “If the voltage is X volts what is the model’s error?”



The “True” voltage vs the predicted voltage

Goals for next week

- Investigate performance against different network parameters
- Continue conference paper.

15 March 2024

Completed conference paper.

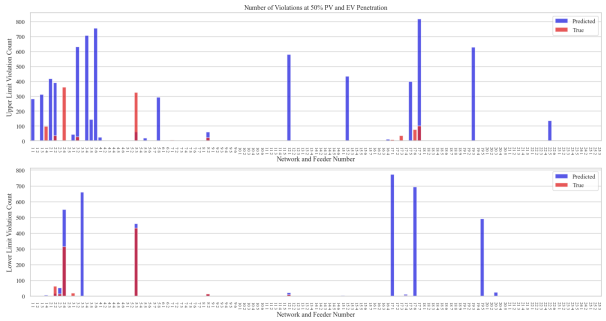
24 March 2024

Achievements

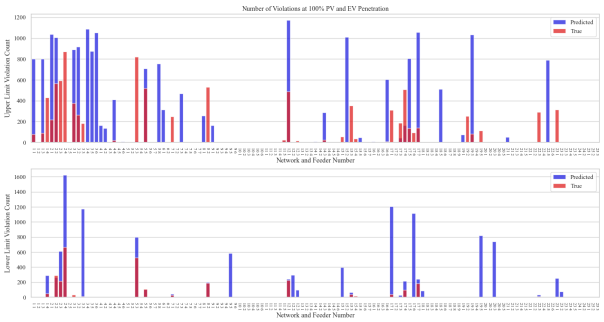
- Began Dissertation
- Looked at number of voltage violations

Penetration Rates

Penetration %	10%	30%	50%	70%	100%
Observed Upper	0	6	12	26	35
Predicted Upper	0	16	22	31	41
Observed Lower	1	6	10	16	22
Predicted Lower	1	11	17	25	33



Number of violations at 50% EV and PV penetration with linear regression and the electrical model.



Number of violations at 100% EV and PV penetration with linear regression and the electrical model.

Goals for next week

- Continue Dissertation

12 April 2024

Worked on dissertation and and oral examination presentation.

19 April 2024

Completed on dissertation and and oral examination presentation.