School of Computing
FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

Final Report

Machine Learning based Workload Prediction for Edge Computing

Toby McCarthy

Submitted in accordance with the requirements for the degree of
BSc Computer Science

2022/2023

COMP3931 Individual Project

The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Final Report	PDF file	Uploaded to Minerva (02/05/23)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) Toby McCarthy

Summary

Edge computing is an emerging area within the technology industry. Due to dynamic demands placed upon edge computing centres, a workload prediction mechanism would allow centres to anticipate future workload and provide the necessary resources to ensure good user experience and performance.

The aim of this report is to examine machine learning approaches to workload prediction to identify which machine learning models perform best in this regard. Additional findings will be identified such as trends in machine learning model performance and sliding window size, as well as what type of data each machine learning model performs best on.

Finally, ideas for future work will be identified to help guide further research into this subject area.

Acknowledgements

I would like to thank my supervisor for this project, Professor Karim Djemame, for being an invaluable resource in helping guide me in the process of this final year project.

I would also like to thank my assessor, Professor Netta Cohen, for giving me helpful feedback in my assessor meeting that allowed me to improve my final report.

Finally, I would like to thank my family and friends for supporting me throughout my university experience. In particular, I would like to thank Caroline and Jason; Ryan and Kirstyn; Anne and Ian; Elizabeth and Peter; Terry and Lin, and Doreen.

Table of Contents

Chapter 1
Introduction and Background Research

1.1 Introduction

My project is focused on the application of machine learning models for workload prediction in edge computing environments. Edge Computing (EC) is a growing area in technology, edge computing centres are local computation centres that sit between the cloud and the user. These centres can provide users which much faster service and reduce the workload on cloud centres by taking on computation that would have otherwise been done by the cloud.

Machine learning prediction models are algorithms that can be trained on the typical workload patterns, and then be used to predict the edge computing systems' workload ahead of time. This would allow time for additional computation resources to be allocated to the edge computing system so it can accommodate for all the changes in workload demands.

1.2 Literature Review

1.2.1 Introduction

In recent years there has been a surge in the number of mobile devices, and the demands placed on cloud platforms. Cloud platforms are typically placed in geographically remote locations, this allows for large centres to be built for much lower prices than if constructed in built-up areas like city centres. This geographical distance from built up areas, where most end users are located, creates an issue with latency as the data travels from the end user to the cloud, and back again. This high latency means lower overall performance and certain technologies - like driverless cars - cannot use cloud resources as they require low response times (1). Additionally, as demand increases, more load is placed upon cloud services requiring an increase in resources to cope with end user demand. Demand in recent times has increased greatly as shown in Al-Dhuraibi et al. (2) and it appears that this will continue, therefore if workload on cloud platforms can be reduced, this would mitigate the issue of rapidly increasing demand.

All these factors have led to the creation of EC centres. These EC centres address the two key issues of latency and load. This is achieved by being located geographically close to end

users, meaning latency is massively reduced, and by performing computations which would have otherwise been performed at the cloud, it reduces the load on the cloud platforms.

However, another problem has been created. Due to the nature of internet traffic, demands change dynamically. This can be in the form of data spikes and dips, or consistent patterns related to how users typically use their devices. This dynamic changes in demand means that an effective and efficient EC centres requires an elastic approach to resource provision.

1.2.2 The Cloud

The cloud “refers to servers that are accessed over the Internet” (3). These servers are in data centres and allows users and companies to offload the computation of software applications from their own machine to the cloud (3), meaning computational processes beyond the means of the local device can be sent to the cloud, processed, then sent back to the device. Furthermore, it enables users and companies to access the same files across multiple devices, since instead of the file stored locally on one device, it is stored in the cloud which any device with an Internet connection can access (3).

1.2.3 Elasticity

Elastic resource provision enables an EC centre to respond to increases and decreases in device demands (2). These resources can be in the form of back-end servers (4). When demand is high, additional resources can be provided to the EC system to ensure Quality-of-Service (QoS) does not drop and prevent Service Level Agreement (SLA) violations (4), violations against the level of service agreed between the user and a service provider (5). Whereas, when demand is low, resources can be reduced to avoid energy wastage and unnecessary costs (4), this ensures the EC system is energy efficient and can help maximise profits. Due to the concerns around climate change and the competitiveness of the technology industry, improving energy efficiency and lowering costs are areas of key importance.

With this elasticity of resources, a prediction mechanism can be used to forecast the workload required in the future (2). This would allow EC centres to request more resources before demand increases (2). This would allow for the necessary time to provide the required resources to the EC centre, before the demand increases.

There is no clear consensus on the amount of time required to allocate additional resources to an EC centre, this is an area of research that could be expanded upon.

1.2.4 Prediction Techniques

Machine Learning (ML) prediction models have shown promise in workload forecasting (6). These predictions can be achieved via the use of a supervised learning algorithm, an algorithm that uses past data and labels to build a model to then make predictions on future data (7). These algorithms can then be broken down into two categories: regression and classification (7). Regression is when the “output variable to be predicted is continuous in nature” (7), this will be the focus of this report. Classification is when the “output variable to be predicted is categorical in nature” (7) so can be used to group results into categories, for example the analysis of tumours sizes and categorising the results into benign, malignant, and intermediate (8).

Aljulayfi and Djemame (6) examines three ML algorithms: Linear Regression (LR), Support Vector Regression (SVR) and Neural Networks (NN). In this paper SVR performed best for data of a decreasing nature, whereas NN algorithms performed best for data of an increasing or fluctuating nature (6). Aljulayfi and Djemame (6) also examines different training splits and the effect of the sliding window size on the performance of the prediction algorithms.

Amiri and Mohammad-Khanli (9) lists all the possible prediction techniques available, including LR, SVM, ARMA, NNs, and KNNs. Amiri and Mohammad-Khanli (9) examines the strengths and weaknesses/challenges of each method when used for predictions. Amiri and Mohammad-Khanli (9) supports the usage of Support Vector Machines (SVMs) over NNs or Auto-Regressive Moving Average (ARMA) models.

Liu et al. (10) proposes a prediction model which is a combination of an ARMA model with an Elman Neural Network (ENN). The ENN model is used to “correct the forecasting results of the ARMA model” (10, p.11).

Moreno-Vozmediano et al. (4) compares the performance of a SVM forecasting model against simpler methods, such as last value, MA and AR. Performance is measured using MAE and RMSE (4). Moreno-Vozmediano et al. (4) showed that SVM outperforms the simpler methods.

1.2.4.1 Linear Regression

LR is the simplest of all ML regression algorithms (7), it is a supervised learning algorithm that finds the “linear correlation between variables” (7). LR aims to find a straight line of best fit from the data, this is done by finding the line that minimises the cost function - this is typically Mean-Squared Error (MSE) (7). MSE is described later in the report.

Amiri and Mohammad-Khanli (9) states the strengths of LR is its simplicity and interpretability, “the degree to which an observer can understand the cause of a decision” (11). The weaknesses/challenges identified by Amiri and Mohammad-Khanli (9) are the assumptions that the data is independent, and the behaviour of the application is linear, as well as the difficulty of selecting the order of the model. This means that LR cannot model non-linear data, which can be a major weakness if the behaviour of an application is non-linear.

1.2.4.2 Support Vector Regression

SVRs are a type of Support Vector Machines (SVMs). SVM “tries to find a line/hyperplane that separates” (13) two classes, SVR works in a similar way to SVM. SVR attempts to find a “hyperplane that best fits the data points” (13). Like LR, SVM is a supervised learning algorithm (14). The difference to LR is its ability to model “non-linear relationships between variables” (14) instead of only linear relationships in LR. In addition, they also have additional flexibility as the model can be adjusted via the tuning of hyperparameters (14). The values of the hyperparameters “control the learning process” (15) of ML models and be used to optimise the model to improve the prediction performance.

Two of the key hyperparameters for SVR are epsilon, the maximum error, and C, the tolerance for points beyond epsilon (16). The hyperparameters epsilon and C are used to define the accuracy and tolerance of the SVR model respectively (16). Another hyperparameter is the kernel, the kernel is responsible for finding a “hyperplane in the higher dimensional space without increasing the computational cost” (13). The kernel can be linear or non-linear, non-linear is used to “capture more intricate patterns in the data” (17). The best kernel to use “depends in the data’s characteristics and the task’s complexity” (17).

Amiri and Mohammad-Khanli (9) identifies the strengths of SVM/SVR as the ability to model nonlinear behaviour and its use of multistep-ahead prediction, which is when data is predicted “multiple time steps into the future” (18). The weakness/challenges of SVM/SVR are the high algorithmic complexity, inability to adapt to workload changes, and the speed of training and testing (9).

1.2.4.3 K-Nearest Neighbour

K-Nearest neighbour (KNN) is another type of supervised ML algorithm, it “assumes that similar things exist in close proximity” (19). “K is the number of nearby points that the model will look at” (20). KNN, like SVR, has hyperparameters that can be tuned to adjust the performance of the model.

The first of these is the distance metric. The standard choice is Euclidean distance which is the distance given by an equation like Pythagoras theorem (20). Another option is Manhattan distance which is distance in the cardinal directions (North, East, South, West) (20). This is controlled by the p value, where when p is 1, the Manhattan distance is used, and when p is 2, the Euclidean distance is used (20).

Another hyperparameter is the addition of weights. When the algorithm looks at the k-nearest neighbours and is using weighting, the points closer to the chosen point have a stronger influence on the decision than points further away (20). The other option is uniform, this is where the distance between points is not factored into the calculations and so are all treated equally.

The strengths of KNN are no training phase, no data assumptions, and its simplicity (9). Meanwhile, its weaknesses/challenges are its speed and high computation cost and the difficulty of selecting the K value and the distance metric (9).

1.2.4.4 ARMA

ARMA is the combination of an autoregression (AR) model with a moving average model (MA). AR models use past data to predict future values (21). MA models calculate the errors of past data to predict future values (21).

Amiri and Mohammad-Khanli (9) states that, like LR, the strengths of ARMA are its simplicity and interpretability. Similarly, the weaknesses/challenges identified by Amiri and Mohammad-Khanli (9) are the assumptions that the data is independent, and the behaviour of the application is linear, as well as the difficulty of selecting the order of the model.

1.2.4.5 Neural Networks

NNs are a type of supervised learning (22). They are a method of machine learning that is inspired by the human brain (22). It is typically referred to as deep learning, and it uses “interconnected nodes or neurons in a layered structure that resembles the human brain” (22). The basic neural network is broken into node layers: the input layer, one or more hidden layers, and the output layer (23). Each node in the layer connects to other nodes and has an “associated weight and threshold” (23). NNs use the training data to improve their accuracy over time (23) and can be used very effectively in ML prediction tasks.

The strengths of NNs, as outlined by Amiri and Mohammad-Khanli (9), are its simple implementation and the ability to model nonlinear behaviour. However, its weaknesses/challenges are needing a lot of time and data and its inability to adapt to workload changes (9).

1.2.4.6 Overfitting and Underfitting

Overfitting occurs when a ML model becomes specialised to the training data to the point where its prediction performance suffers. This is often due to noise in the training data, which the ML model identifies patterns in, and then uses to predict future data, where the same noise patterns will not be present. Noise is “deviation from the true dataset” (24) and comes from inaccuracies in data measurement or collection (25) this means that noise in a dataset can obscure the underlying patterns in the rest of the data.

Mali (7) states the ways to prevent overfitting are cross-validation, adding more relevant data if the dataset is too small, doing feature selection if dataset is too large (reducing the size of the dataset), and regularisation.

Cross-validation is the process of removing a portion of the training data, then using this removed portion to test the model (26). This portion of code is called the validation set (26). This ensures that your model generalises to the data well and prevents it from suffer from overfitting when measured using the final testing data (26). One way to do this is K-Fold cross validation, which is when “data is divided into k subsets” (26), where one set is used for validation and the rest for training, this is then repeated for each of the k subsets (26). The average of all k trials is an error estimation to evaluate the total effectiveness of the model (26).

Underfitting is the reverse of overfitting. This occurs when the ML model does not learn the training data sufficiently, meaning it does not identify the “underlying patterns from the data” (7) and cannot generalise to predict future data. Mali (7) says underfitting can be prevented by increasing model complexity, increasing the number of features in the training data (increasing the size of the dataset), and removing noise from the data.

Model complexity is the “number of predictor or independent variables or features” (27) that a model uses to make predictions. A high degree of model complexity can lead to overfitting (27) as the model becomes too specialised to the training data. However, a low degree of model complexity can lead to underfitting as the model cannot “capture all the relevant information in the data” (27).

1.2.5 Sliding Window Technique

The sliding window technique is a technique that can be applied to ML algorithms. It uses the previous n samples of a dataset to forecast the next value (6). The size of the window effects the accuracy of a model. A small window size might be unrepresentative of the rest of the data, whereas a large window size can cause overfitting (6). Aljulayfi and Djemame (6)

determined that the best performing window size depends on the trend of the data within the period examined.

In Aljulayfi and Djemame (6), only for fluctuating data does increasing window size have a "positive impact on all ML algorithms overall evaluation metrics" (6, p.6). For decreasing and increasing data, increasing window size had no "significant impact" (6, p.5) on performance (6). For LR and SVR, changing the window size did not have a significant effect on overall performance, however for NN, changing window size had a significant effect (6).

1.2.6 Evaluation Methods

To evaluate the performance of each ML model, evaluation metrics must be used. The metrics measure the error between the predictions made by the ML models, against the actual values from the testing data. The four metrics looked at will be Mean-Absolute-Error (MAE), Mean-Absolute-Percentage-Error (MAPE), Root-Mean-Squared-Error (RMSE), and Mean-Squared-Error (MSE).

MAE is the "sum of absolute errors divided by the sample size" (28). MAPE is the "average of the absolute percentage errors" (29). RMSE is the square root of the MSE. MSE is the "average of the square of the errors" (30).

MAE: $\frac{1}{N}\sum_{j=1}^{N}\left|y_j - \widehat{y_j}\right|$ (31). Where y_j is actual value, $\widehat{y_j}$ is the predicted value, and N is the sample size.

MSE: $\frac{1}{N}\sum_{j=1}^{N}\left(y_j - \widehat{y_j}\right)^2$ (31). Where y_j is actual value, $\widehat{y_j}$ is the predicted value, and N is the sample size.

RMSE: $\sqrt{MSE}$.

MAPE: $\frac{1}{N}\sum_{t=1}^{N}\left|\frac{A_t - F_t}{A_t}\right|$ (32). Where A_t is the actual value, F_t is the predicted value, and N is the sample size.

Aljulayfi and Djemame (6) used Mean-Absolute-Error (MAE), Mean-Absolute-Percentage-Error (MAPE), and Root-Mean-Squared-Error (RMSE). Moreno-Vozmediano et al. (4) uses MAE, Mean-Squared-Error (MSE), and RMSE. Amiri and Mohammad-Khanli (9) lists the evaluation metrics used across the literature, including MAE, MSE, MAPE, and RMSE.

1.2.7 Focus of Report

The focus of this report will be expanding on the existing knowledge of the use of ML algorithms for workload prediction for EC centres. Aljulayfi and Djemame (6) covers LR, SVR, and NN. From this an additional ML algorithm will be investigated, KNN, and will be compared to two of the algorithms in Aljulayfi and Djemame (6), LR and SVR.

In addition, the sliding window technique used in Aljulayfi and Djemame (6) will be used to see how it affects the performance of KNN. MAE, MAPE, and RMSE will be used to evaluate performance as they feature in (6, 4, 9) and these evaluation methods will give sufficient coverage of the performance of the ML algorithms to give accurate insights into their utility.

Chapter 2
Methods

2.1 Dataset

The dataset used is the Shanghai Telecom dataset, which is reported in (33-34, 12). This dataset has over 7.2 million records of accessing the internet through 3,233 base stations from 9,481 mobile phones over a period of six months (35). The dataset has six parameters Month, Date, Start Time, End Time, Base Station Location, and User ID (35). For this report, we will use the Month, Date, and Start Time parameters.

This dataset was used in Aljulayfi and Djemame (6) and Guo et al. (12). This dataset is useful as it has a large number of records and is taken over a lengthy period of time. This allows patterns to be found in the data, to define what typical usage is, and enables the ML models to be trained on these patterns to then predict future values.

2.2 Dataset Analysis

Starting by examining the dataset, the data for each day can be found to follow a daily pattern (6). The data falls into three categories: decreasing, increasing, and fluctuating (6). These categories reflect the trend of the data and indicate periods of decreasing, increasing, and fluctuating number of users on the base stations - reflecting changes in the demands that would be required by an EC centre (6).

These categories are shown in figure 1. This graph is created by calculating the number of total connections for each minute in the day. This data is then manipulated by removing the outliers to allow the pattern of data to become clear, like what was done in Aljulayfi and Djemame (6). From this graph, the decreasing period can be seen is from the later hours of the previous day to the early hours of the next day (approximately 9pm - 5am), the increasing period follows the decreasing period (approximately 6am - 12pm), then finally the fluctuating period (approximately 1pm - 8pm).

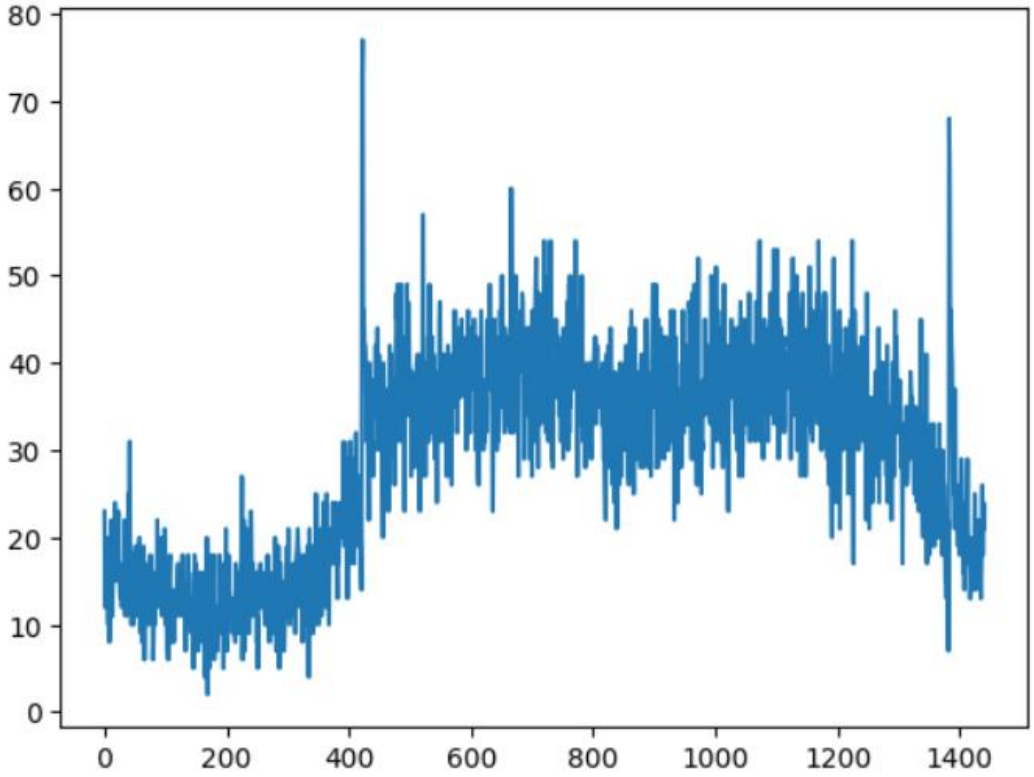

Figure 1 - Graph of number of connections per minute on a typical data, with outliers removed to illustrate the trend more clearly

With the three categories defined, the ML models can be trained for each category to find which ML models perform best at predicting in each category. This is done to highlight the strengths and weaknesses of each ML model for predicting the different types of data.

From there, due to the large dataset size and the regularity of the daily patterns, a single day from the dataset is used for training and testing the ML models, similarly to Aljulayfi and Djemame (6). For this, the month and day with the least Not a Number (Nan) values as a percentage of the total number of records for that day was identified. This came out to be the 13$^{th\ of}$ June. Therefore, the 13$^{th\ of}$ June would be used as the date for the ML models.

The data for the 13$^{th\ of}$ June has a total of 300,648 records, which is large number of records for training the ML models. To reduce training and testing times, the amount of data for each ML model has been reduced. To reduce the amount of data, a single hour for each category was chosen to train and test the models. An hour in the middle of the category was chosen to provide a balance between the total number of records and the strength of the trend. For the decreasing category 1am was chosen, for the increasing category 9am was chosen, and for the fluctuating category 4pm was chosen.

Figures 2, 3, and 4 show the data for the chosen hours for each category, a line of best fit is applied to each graph to show the trend of data to confirm it matches the trend of the category the hour is chosen from. Furthermore, it shows that the trends in data categories are not that pronounced, as the line of best fit has only a small incline or decline, for the increasing and decreasing data. In addition, the graphs show that the data varies by large amounts illustrating the data has a high standard deviation.

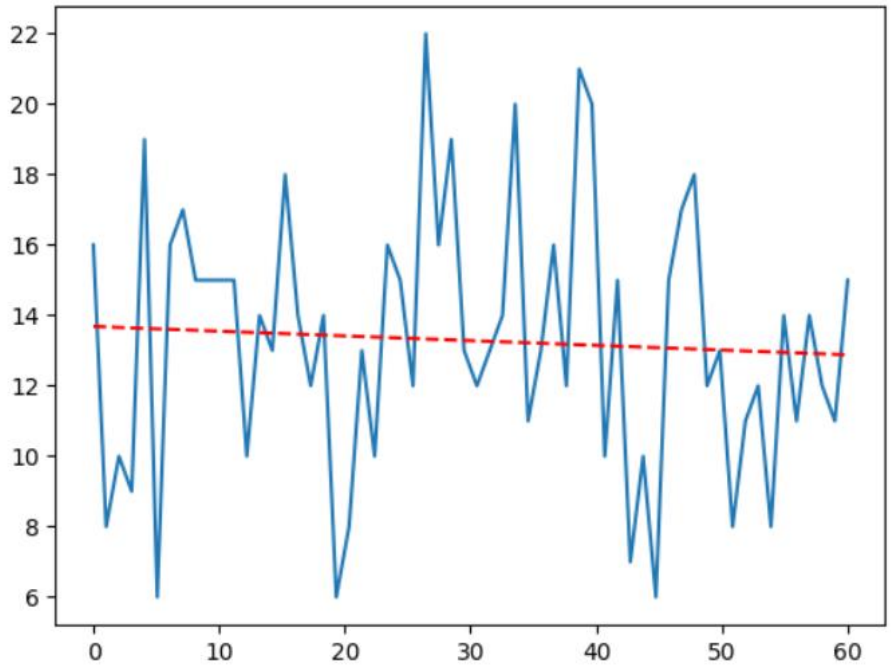

Figure 2 - graph of connections per minute for decreasing data category

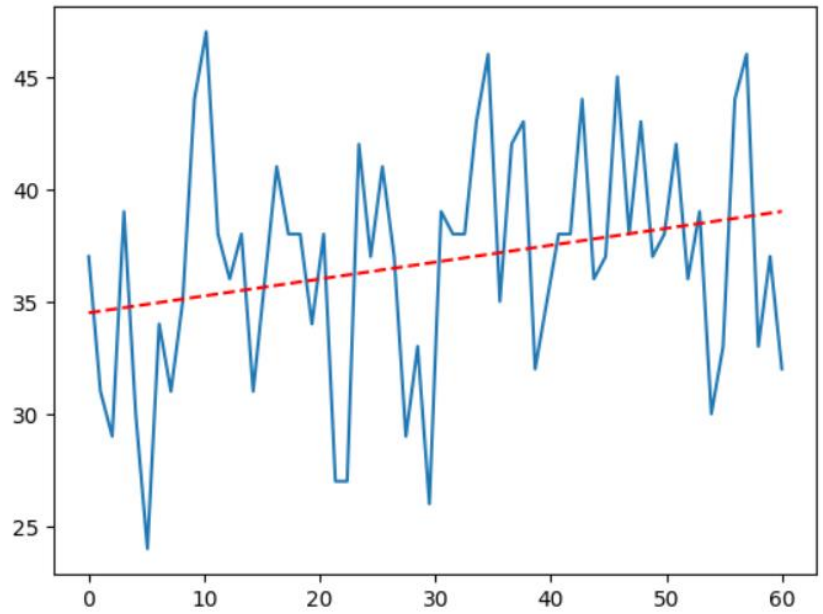

Figure 3 - graph of connections per minute for increasing data category

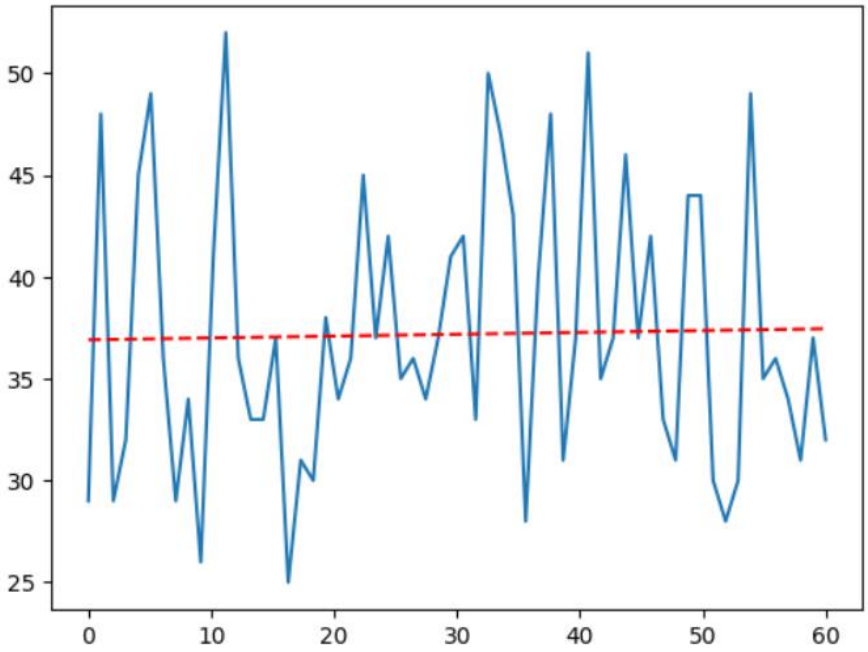

Figure 4 - graph of connections per minute for fluctuating data category

2.3 Dataset Preparation

As mentioned previously, the totals for each minute for the chosen hours are calculated. This data is then used for the training and testing of the ML models. The decreasing data category has 797 records, the increasing data category has 2205 records, and the fluctuating data category has 2231 records.

2.4 Software

The environment used was the Python Jupyter Notebook. This environment was used as it is easy-to-use, and I had previous experience using it for ML-based projects. This environment allows the importing of libraries into the notebook, this will be used throughout to implement the ML models, evaluation metrics, and train-test splits.

The data is stored in an Excel file and is imported via the pandas' libraries. The graphs are plotted via the Matplotlib libraries. Numpy libraries are used for any mathematical operations.

2.5 Sliding Window Technique

To test the effect of the sliding window on each of the ML models, each model will be trained and tested on every window size between 2 and 9 (inclusive). This mirrors the approach in Aljulayfi and Djemame (6) and will be used to compare against the results in this paper.

When using a window size of 2, the ML models cannot make a prediction/ calculate a result for the first or second value, this is because the first and second values have insufficient previous data to allow for a window size of 2. For each increase in window size, an additional value is needed before a prediction/ result can be calculated. Due to this constraint, the training and testing datasets will be adjusted to allow for sliding window technique to work correctly. This will ensure that the values produced for each window size is accurate and demonstrates the effect of increasing the sliding window in workload prediction for each of the ML models.

2.6 ML Models

2.6.1 Linear Regression

The LR model was imported from sklearn.linear_model.LinearRegression. LR has no hyperparameters, therefore no hyperparameter tuning was required.

2.6.2 Support Vector Regression

The SVR model was imported from sklearn.svm.SVR. For the SVR model, the same hyperparameters from Aljulayfi and Djemame (6) were used as these hyperparameters wre found to perform the best on this dataset.

Parameter	Value
C	1.0
Kernel	RBF

Table 1 - SVR Configuration

2.6.3 K-Nearest Neighbours

The KNN model was imported from sklearn.neighbors.KNeighborsRegressor. The scikit learn grid search was used from sklearn.model_selection.GridSearchCV. Grid search was used to final the optimal parameters for the KNN model.

GridSearchCV is used to tune the hyperparameters to optimise performance for a given dataset (36). The functions loops through a predefined set of hyperparameters, trains the model on the training data, then evaluates the performance via K-fold cross-validation (mentioned previously) (36). The grid search was done on the decreasing data and tested for all window sizes, the hyperparameter values that featured most frequently across all

window sizes was chosen. The predefined set of hyperparameters to be tested are written in table 2.

Hyperparameters	**Possible Values**
Number of neighbours	5, 10, 15, 20, 25
Algorithm	Ball Tree, KD Tree, Brute Forces
Leaf Size	20, 25, 30, 35, 40
Weights	Uniform, Distance

Table 2 - Predefined Hyperparameters for GridSearchCV for KNN model

From this, the best hyperparameters were found as written in table 3.

Parameter	**Value**
Number of neighbours	25
Algorithm	Ball Tree
Leaf Size	20
Weights	Uniform

Table 3 - Final KNN Configuration

LR and SVR were used in Aljulayfi and Djemame (6) while KNN was not. This will be one of the areas that the report expands upon current understanding. The performance of this model against SVR and LR will determine whether this ML model should be considered for EC workload prediction systems.

2.7 Training and Testing

Aljulayfi and Djemame (6) found that an 80/20 training-testing split performed best, therefore that will be the split used for the ML models. The algorithm is from the scikit learn libraries and imported from sklearn.model_selection.train_test_split. The splits for all data categories use a random state of 42. The use of a constant random state ensures that the results are repeatable externally and do not change between each execution of the code.

Each of the three ML models has its own separately trained model for each data category, this meaning there was 9 models trained for each window size. This results in a total of 72 separately trained ML models covering 8 different window sizes, 3 ML algorithms, and 3 data categories.

The models were trained on the training data, 80% of the total data, using each window size in the training. From there, it makes predictions on the test data, 20% of the total data. These predictions and the test data are then passed into the evaluation stage to obtain the results of each ML model.

2.8 Evaluation

Three performance metrics will be used, these will be MAE, MAPE, and RMSE. Algorithms for calculating these metrics will be implemented from the scikit learn libraries. The algorithms will be imported from sklearn.metrics.mean_absolute_error, sklearn.metrics.mean_absolute_percentage_error, and sklearn.metrics.mean_squared_error respectively, where the root of MSE will be then be calculated to get the RMSE value.

The evaluations take the predictions made by each ML model and compares it to the actual test data. Each of the three-performance metrics are calculated for each ML model and stored in an array. The array is then printed out to two decimal places as pictured in figures 5, 9 and 13. This allows for the comparisons of all the model and allows for identifying the best performing ML model and window size for each performance metric, as well as trends in the data.

The use of these metrics ensures that the results from this report are thorough and conclusive in expanding the current understanding of the performance of the chosen ML models in EC workload prediction.

Chapter 3
Results

3.1 Decreasing

Decreasing

Window Size	MAE LR	MAE SVR	MAE K	MAPE LR	MAPE SVR	MAPE K	RMSE LR	RMSE SVR	RMSE K
2	4.26	3.95	3.93	0.34	0.33	0.34	4.72	4.43	4.43
3	2.92	3.10	2.95	0.24	0.26	0.24	3.40	3.52	3.44
4	3.25	2.42	2.30	0.25	0.18	0.17	4.24	2.97	2.96
5	2.21	1.67	1.62	0.16	0.12	0.12	2.75	2.05	1.89
6	3.02	3.02	3.11	0.30	0.30	0.31	3.91	3.95	4.08
7	4.44	4.11	4.05	0.41	0.39	0.39	4.90	4.76	4.73
8	2.52	2.27	2.19	0.23	0.21	0.21	3.25	2.64	2.63
9	3.68	3.39	3.37	0.26	0.23	0.22	4.61	4.31	4.36

Figure 5 - results for the decreasing data category

From figure 5, the best performing model and window size for MAE was KNN with a window size of 5, the best performing model and window size for MAPE was KNN and SVR with a window size of 5, and the best performing model and window size for RMSE was KNN with a window size of 5.

From this, the best algorithm and window size pairing for the decreasing data category is a KNN model with a window size of 5.

3.1.1 Window Size

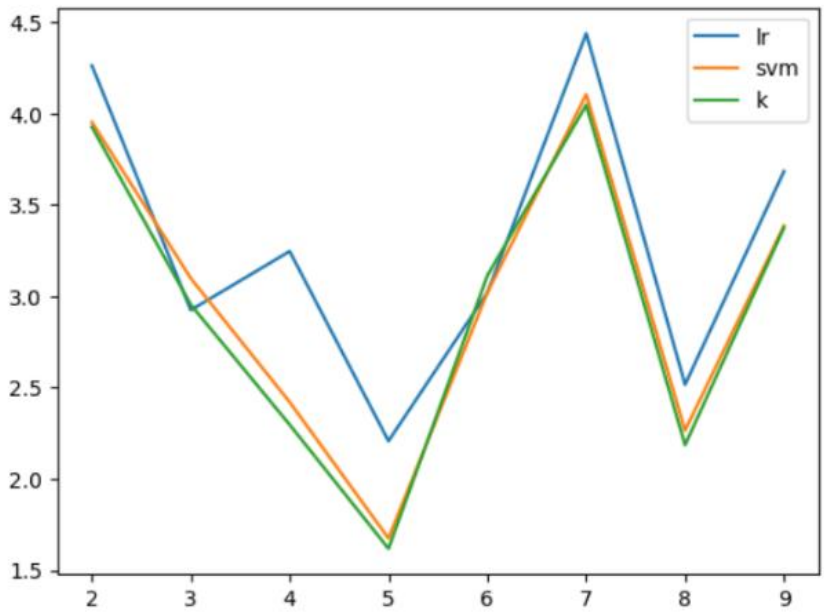

Figure 6 - graph of MAE results for each ML model across each window size for the decreasing data category

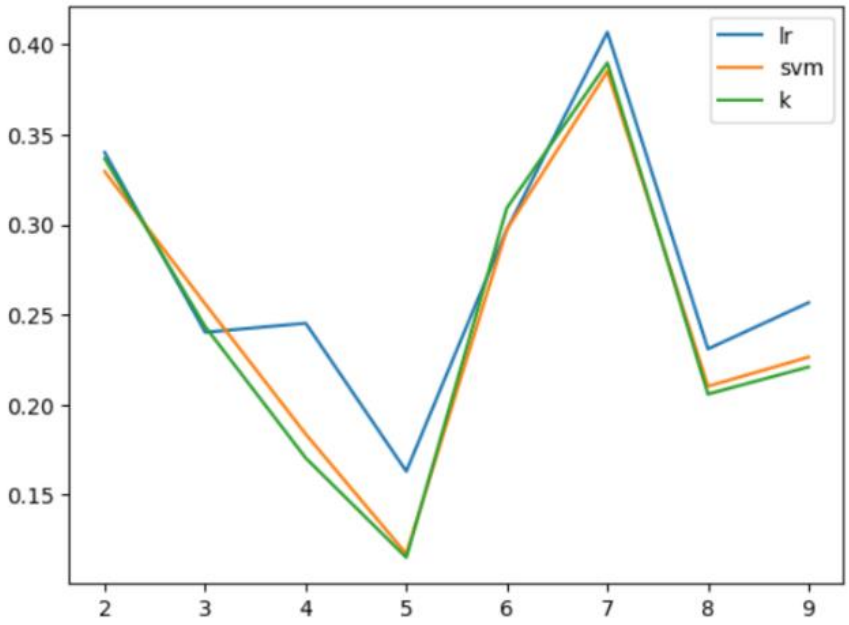

Figure 7 - graph of MAPE results for each ML model across each window size for the decreasing data category

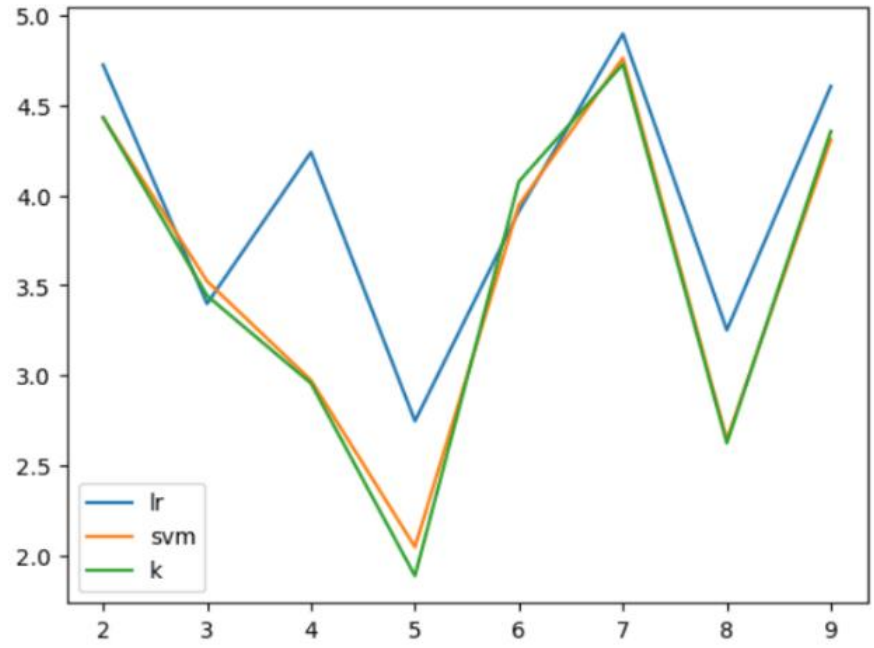

Figure 8 - graph of RMSE results for each ML model across each window size for the decreasing data category

Figures 6, 7, and 8 show that increasing the window size for the decreasing data category has no clear effect on the performance of any of the ML models, as the values vary by large amounts. The best performing window size for all the models, and across all the performance metrics, was a window size of 5. The worst performing window size for all models, and across all the performance metrics, was a window size of 7 (a window size of 2 also performed very poorly).

In Aljulayfi and Djemame (6), LR, SVR, and NN had no clear improvement in accuracy as window size increased. This is consistent with the findings in figures 6, 7, and 8, which similarly showed no improvement in accuracy as window size increased for LR, SVR, or KNN. In Aljulayfi and Djemame (6), the best window size for SVR was 3, whereas in this report the best performing window size for all ML models was 5.

3.2 Increasing

Increasing Window Size	MAE LR	MAE SVR	MAE K	MAPE LR	MAPE SVR	MAPE K	RMSE LR	RMSE SVR	RMSE K
2	4.90	5.04	5.16	0.16	0.17	0.17	5.99	6.23	6.42
3	3.86	3.64	3.75	0.09	0.09	0.09	4.57	4.60	4.66
4	4.03	4.03	3.87	0.11	0.11	0.10	4.76	4.75	4.67
5	3.88	2.97	3.28	0.10	0.07	0.08	4.97	4.21	4.53
6	4.08	4.05	3.77	0.10	0.10	0.10	4.95	4.63	4.52
7	4.54	3.20	3.36	0.12	0.08	0.09	5.50	4.07	4.25
8	3.47	3.17	3.08	0.10	0.10	0.10	4.63	4.52	4.64
9	4.47	3.76	3.69	0.14	0.12	0.12	5.39	5.31	5.35

Figure 9 - results for the increasing data category

From figure 9, the best performing model and window size for MAE was SVR with a window size of 5, the best performing model and window size for MAPE was SVR with a window size of 5, and the best performing model and window size for RMSE was SVR with a window size of 7.

From this, the best algorithm and window size pairing for the increasing data category is a SVR model with a window size of 5.

3.2.1 Window Size

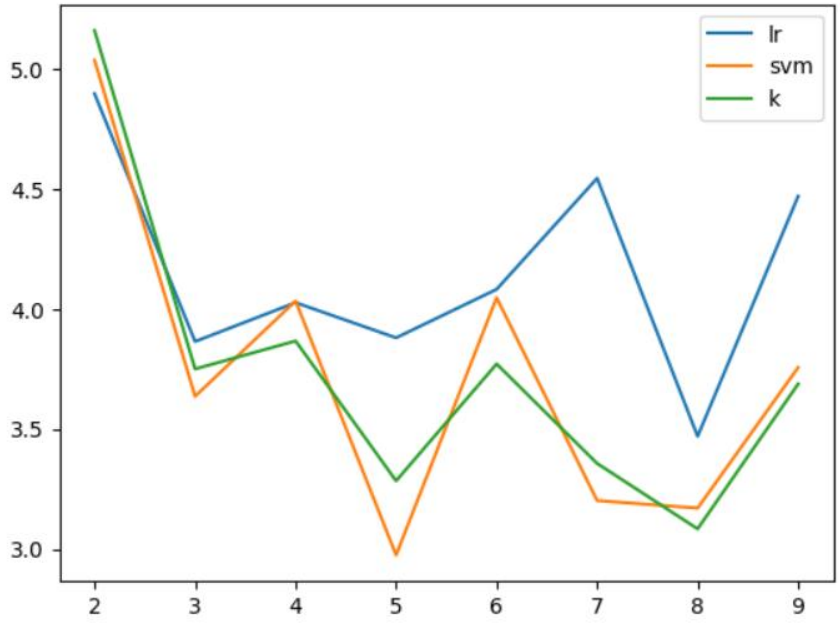

Figure 10 - graph of MAE results for each ML model across each window size for the increasing data category

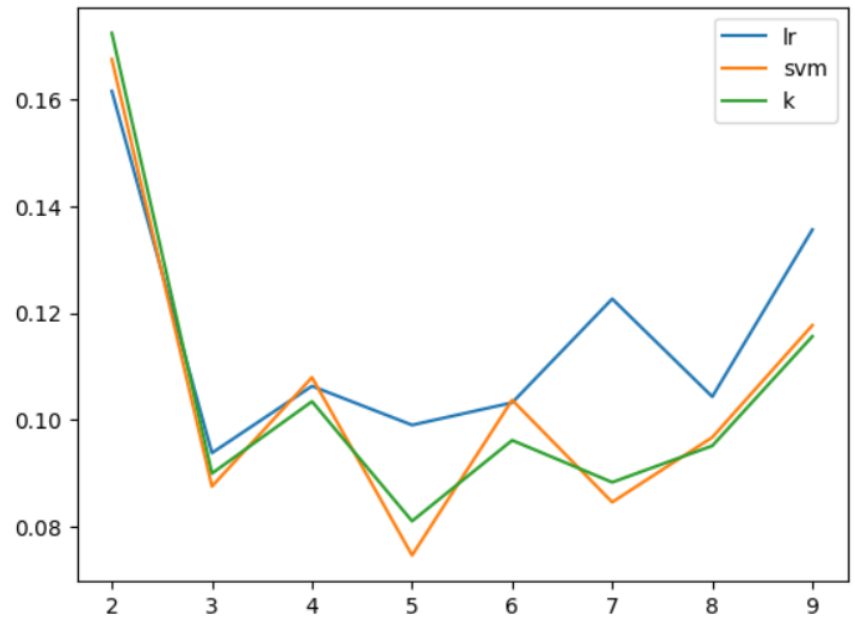

Figure 11 - graph of MAPE results for each ML model across each window size for the increasing data category

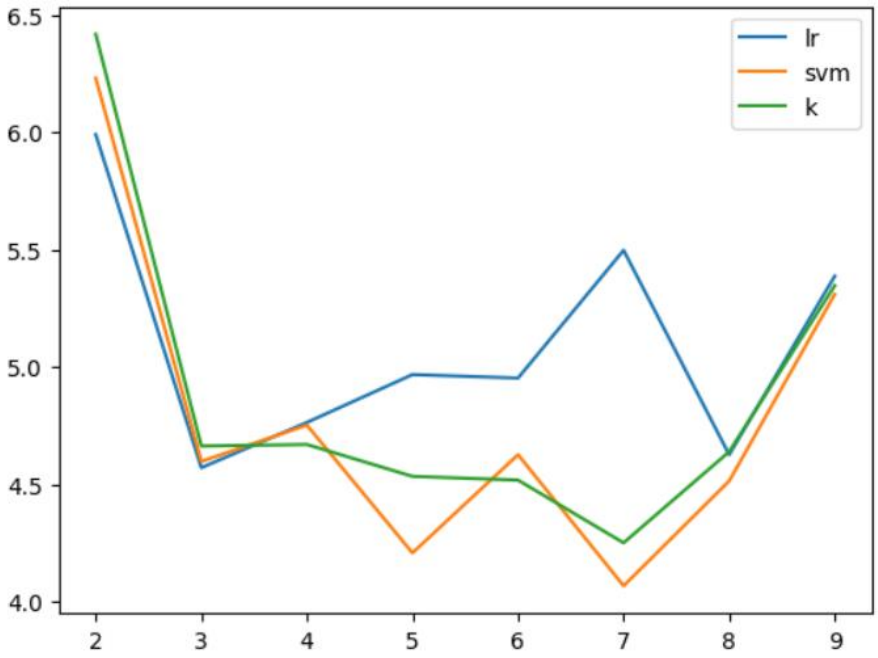

Figure 12 - graph of RMSE results for each ML model across each window size for the increasing data category

Figures 10, 11 and 12 show that increasing the window size for the increasing data category appears to improve performance across the performance metrics. The best performing window sizes for SVR was 5 and 7. The worst performing window size was 2, and this was by a large margin in all the evaluation metrics for all the ML models.

The results from figures 10, 11, and 12 differ from those in (6). Aljulayfi and Djemame (6) found that there was no clear improvement in any of the ML models when increasing window size, however, there appears to be an increase in performance in this report. In Aljulayfi and Djemame (6), the best accuracy was for NN with a window size of 9, whereas in this report the best accuracy was for SVR with a window size of 5 or 7.

3.3 Fluctuating

Fluctuating

Window Size	MAE LR	MAE SVR	MAE K	MAPE LR	MAPE SVR	MAPE K	RMSE LR	RMSE SVR	RMSE K
2	5.80	5.13	5.26	0.17	0.15	0.15	6.45	5.90	6.11
3	5.25	5.94	5.70	0.15	0.16	0.16	6.48	7.25	6.84
4	7.27	7.60	7.28	0.21	0.21	0.21	8.41	9.04	8.47
5	6.90	5.42	5.71	0.19	0.14	0.16	7.16	6.10	6.04
6	6.18	4.69	5.28	0.19	0.14	0.16	7.60	6.87	6.94
7	5.14	3.45	3.78	0.14	0.09	0.11	5.75	4.28	4.46
8	6.77	5.47	5.81	0.17	0.13	0.14	8.65	7.82	7.52
9	5.24	5.02	4.57	0.13	0.12	0.11	6.75	7.18	6.41

Figure 13 - results for the fluctuating data category

From figure 13, the best performing model and window size for MAE was SVR with a window size of 7, the best performing model and window size for MAPE was SVR with a window size of 7, and the best performing model and window size for RMSE was SVR with a window size of 7.

From this, the best algorithm and window size pairing for the fluctuating data category is a SVR model with a window size of 7.

3.3.1 Window Size

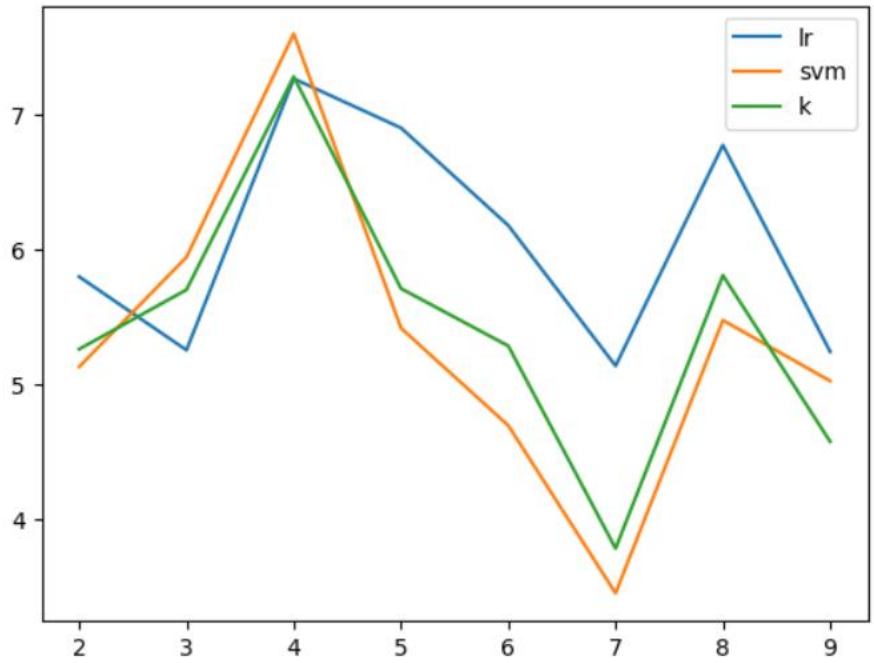

Figure 14 - graph of MAE results for each ML model across each window size for the fluctuating data category

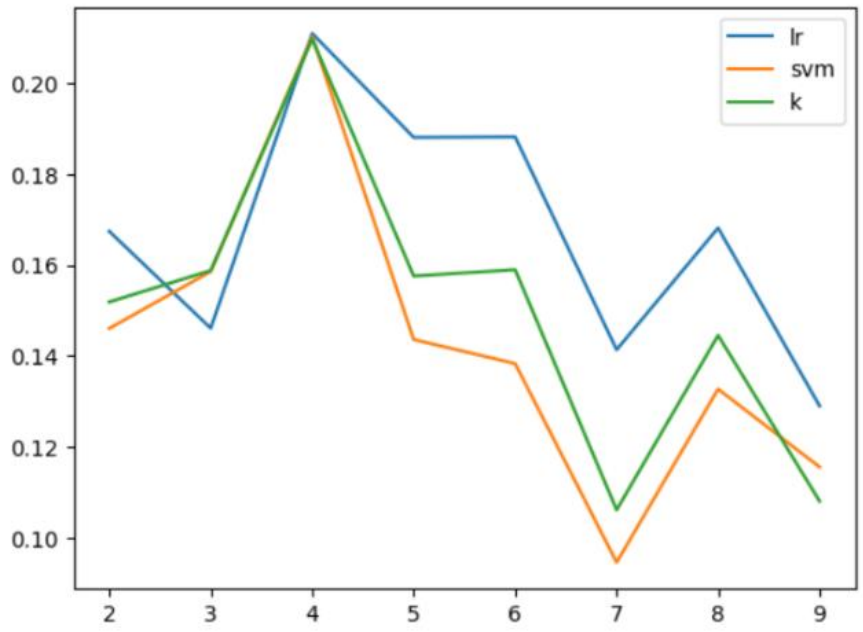

Figure 15 - graph of MAPE results for each ML model across each window size for the fluctuating data category

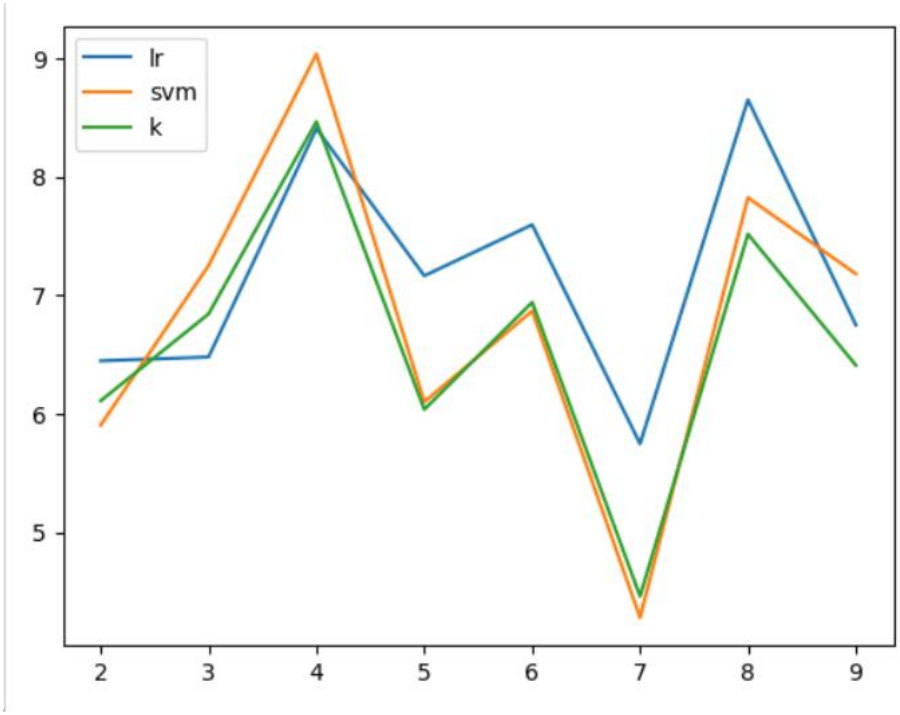

Figure 16 - graph of RMSE results for each ML model across each window size for the fluctuating data category

Figures 14, 15, and 16 show that increasing the window size for fluctuating data had a positive impact on the performance of the ML models. Figures 14 and 15 clearly show a downwards trend in the error metrics as window size increases. However, figure 16 is not as clear, though a downwards trend does appear to be present. For SVR and KNN, the best performing window size was 7. The worst performing window size for all performance metrics and all ML models was 4.

This finding is consistent with the results in Aljulayfi and Djemame (6) where LR, SVR, and NN had clear improvements in accuracy as window size increased. Aljulayfi and Djemame (6) found that the best performance was for NN with a window size of 9, however in this report the best performance was SVR with a window size of 7.

Chapter 4
Discussion

4.1 Conclusions

From the results, KNN and SVR both outperformed LR. This was the same in all three of the data categories. KNN was the best performing for decreasing data, meanwhile SVR was the best performing for both increasing and fluctuating data. In Aljulayfi and Djemame (6), SVR was the best performing for decreasing data, while NN was the best performing for increasing and fluctuating data. This result indicates that the Context-Aware Prediction Framework (CAPF) in Aljulayfi and Djemame (6) should use KNN instead of SVR in the decreasing part of the daily workload pattern.

Window Size	Dec MAE	Dec MAPE	Dec RMSE	Inc MAE	Inc MAPE	Inc RMSE	Fluc MAE	Fluc MAPE	Fluc RMSE
2	KNN	SVR	SVR/ KNN	LR	LR	LR	SVR	SVR/ KNN	SVR
3	LR	LR/KNN	LR	SVR	ALL	LR	LR	LR	LR
4	KNN	KNN	KNN	KNN	KNN	SVR	LR	ALL	LR
5	KNN	SVR/ KNN	KNN	SVR	SVR	SVR	SVR	SVR	KNN
6	LR/ SVR	LR/ SVR	LR	KNN	ALL	KNN	SVR	SVR	SVR
7	KNN	SVR/ KNN	KNN	SVR	SVR	SVR	SVR	SVR	SVR
8	KNN	SVR/ KNN	KNN	KNN	ALL	SVR	SVR	SVR	KNN
9	KNN	KNN	SVR	KNN	SVR/ KNN	SVR	KNN	KNN	KNN

Table 4 - table for the best performing ML model across each window size, performance metric, and data category. Colour coded such that LR is red, SVR is green, KNN is blue, and where all models were equal is black.

Table 4 shows the best performing ML model for each window size, evaluation metric, and data category. This table shows that LR performed well at low window size across all the evaluation metrics and data categories. This suggests that with systems with a small amount of previous data to train the models on, LR is potentially the best ML model.

In the high window sizes, 8 and 9, KNN appears to be the best performing model across the evaluation metrics and data categories. This suggests that systems with large amounts of previous data to train the models on, KNN is potentially the best ML model.

SVR was fairly consistent across the window sizes, generally being the best ML model for some of the evaluation metrics for each window size. It appears that SVR outperforms the other ML models in fluctuating data more so than the other data categories. This suggests that SVR is the best ML model for data that fluctuates without clearly increasing or decreasing.

KNN performed best for the decreasing data category. There are two potential factors that could cause this finding. Firstly, the KNN model was tuned for the decreasing data category, this means the tuning has been specialised for this data category. This would have an influence on why KNN performed particularly well for the decreasing data category. Another possible factor is that the decreasing data category, 797 records, is far smaller than the increasing, 2205 records, and fluctuating, 2231 records, data categories. This could mean that KNN outperforms SVR on smaller datasets.

Finally, from figures 6, 7, 8, 14, 15, and 16, the performance of SVR and KNN appear to follow similar trends in the decreasing and fluctuating data categories. Both models appear to perform similarly in each metric and vary at the same rate as window size changes. This similarity in performance highlights how, despite SVR beating KNN in the increasing and fluctuating data categories, both models would perform similarly in practise as the performance of each was consistently close. The performance of SVR and KNN diverge in the increasing data category, as demonstrated in figures 10, 11, and 12, perhaps highlighting the strength of SVR in this data category.

4.2 Ideas for future work

This report could be expanded by looking at finding the best hyperparameters for each data category for the KNN model. In this report, only decreasing data was used to find the best hyperparameters for KNN. Tuning the model for each data category may cause KNN to outperform SVR in the increasing and fluctuating data category, therefore meaning it would be the best ML model across all three data categories. This could then facilitate a

comparison to a KNN model, the best performing model for increasing and fluctuating data in Aljulayfi and Djemame (6), to identify which performs better.

Another idea for future work would be looking at the effect of larger window sizes. Due to the large amount of traffic and potential data that EC centres would utilise, larger window size than 9 could be used for workload prediction. Also, the results from this report show a positive correlation between window size and performance for fluctuating data, in addition to not appearing to show a negative correlation between increasing window size and the performance of the ML models in decreasing and increasing data. Therefore, an investigation of window sizes larger than 9 could be beneficial for the prediction performance of ML models.

In addition, future work could examine the effect of changing the training/testing ratio from 80/20 and evaluating the effects this has on the performance of the KNN model. Despite Aljulayfi and Djemame (6) showing that 80/20 outperformed 70/30 for LR, SVR, and NN, it is unclear whether KNN would perform better with a 70/30 split or an 80/20 split.

Another possible area to look at is how each ML model performs on a range of training data sizes. Some ML models perform better with smaller training datasets, and some perform better with larger training datasets. An investigation into how this effects different ML models could be potentially useful for EC centres, as depending on the centre’s capacity for previous data storage, different ML models may perform better than others due to the available training dataset size.

The final area for future work identified is looking at different sized time intervals. This report only covered time intervals of 1 minute. An investigation into the performance of ML prediction algorithms on smaller and larger time intervals has the potential to improve the performance of the prediction algorithms and provide greater utility to the implementation of these algorithms in EC centres.

List of References

1. Rafie, M. Autonomous Vehicles Drive AI Advances for Edge Computing. [Online]. 2021. [Accessed 27 April 2023]. Available from: https://www.3dincites.com/2021/07/autonomous-vehicles-drive-ai-advances-for-edge-computing/
2. Al-Dhuraibi, Y., Paraiso, F., Djarallah, N. and Merle, P. Elasticity in Cloud Computing: State of the Art and Research Challenges. IEEE Transactions on Services Computing. 2018, **11**(2), pp.430–447.
3. CloudFlare. What is the cloud?. [Online]. [no date]. [Accessed 27 April 2023]. Available from: https://www.cloudflare.com/en-gb/learning/cloud/what-is-the-cloud/
4. Moreno-Vozmediano, R., Montero, R. S., Huedo, E. and Llorente, I. M. Efficient resource provisioning for elastic Cloud services based on machine learning techniques. Journal of Cloud Computing Advances, Systems and Applications. 2019, **8**, article no:5 [no pagination].
5. Wikipedia. Service-level agreement. [Online]. 2023. [Accessed 24 April 2023]. Available from: https://en.wikipedia.org/wiki/Service-level_agreement
6. Aljulayfi, A. and Djemame, K. A Machine Learning based Context-aware Prediction Framework for Edge Computing Environments. Proceedings of the 11th International Conference on Cloud Computing and Services Science. 2021, ISBN 978-989-758-510-4; ISSN 2184-5042, SciTePress, pp.143-150.
7. Mali, K. Everything you need to Know about Linear Regression!. 04 October. Analytics Vidhya. 2021. [Online]. [Accessed 17 April 2023]. Available from: https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/
8. Liu, R., Pan, D., Xu, Y., Zeng, H., He, Z., Lin, J., Zeng, W., Wu, Z., Luo, Z., Qin, G. and Chen, W. A deep learning–machine learning fusion approach for the classification of benign, malignant, and intermediate bone tumors. European Radiology. 2022, **32**, pp.1371–1383
9. Amiri, M. and Mohammad-Khanli, L. Survey on prediction models of applications for resources provisioning in cloud. Journal of Network and Computer Applications. 2017, **82**(1084-8045), pp.93–113.
10. Liu, B., Guo, J., Li, C. and Luo, Y. Workload forecasting based elastic resource management in edge cloud. Computers and Industrial Engineering. 2020, **139**(0360–8352), pp.1–12.
11. Miller, T. Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence. 2019, **267**, p.8.

12. Guo, Y., Wang, S., Zhou, A., Xu, J., Yuan, J. and Hsu, C. User allocation-aware edge cloud placement in mobile edge computing. Software: Practice and Experience. 2020, **50**(5), pp.489-502.
13. Sethi, A. Support Vector Regression Tutorial for Machine Learning. 27 March. Analytics Vidhya. 2020. [Online]. [Accessed 17 April 2023]. Available from: https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/
14. Dobilas, S. Support Vector Regression (SVR) — One of the Most Flexible Yet Robust Prediction Algorithms. 20 December. Towards Data Science. 2020. [Online]. [Accessed 17 April 2023]. Available from: https://towardsdatascience.com/support-vector-regression-svr-one-of-the-most-flexible-yet-robust-prediction-algorithms-4d25fbdaca60
15. Nyuytiymbiy, K. Parameters and Hyperparameters in Machine Learning and Deep Learning. 30 December. Towards Data Science. 2020. [Online]. [Accessed 24 April 2023]. Available from: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac
16. Sharp, T. An Introduction to Support Vector Regression (SVR). 03 March. Towards Data Science. 2020. [Online]. [Accessed 17 April 2023]. Available from: https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2
17. GeeksForGeeks. Support Vector Regression (SVR) using Linear and Non-Linear Kernels in Scikit Learn. [Online]. [no date]. [Accessed 18 April 2023]. Available from: https://www.geeksforgeeks.org/support-vector-regression-svr-using-linear-and-non-linear-kernels-in-scikit-learn/
18. Brownlee, J. 4 Strategies for Multi-Step Time Series Forecasting. 08 March. Machine Learning Mastery. 2017. [Online]. [Accessed 18 April 2023]. Available from: https://machinelearningmastery.com/multi-step-time-series-forecasting/
19. Harrison, O. Machine Learning Basics with the K-Nearest Neighbors Algorithm. 10 September. Towards Data Science. 2018. [Online]. [Accessed 18 April 2023]. Available from: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
20. Miller, M. The Basics: KNN for classification and regression. 18 October. Towards Data Science. 2019. [Online]. [Accessed 18 April 2023]. Available from: https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955
21. Shetty, C. Time Series Models. 22 September. Towards Data Science. 2020. [Online]. [Accessed 24 April 2023]. Available from: https://towardsdatascience.com/time-series-models-d9266f8ac7b0

22. Amazon. What Is A Neural Network?. [Online]. [no date]. [Accessed 20 April 2023]. Available from: https://aws.amazon.com/what-is/neural-network/
23. IBM. What are neural networks?. [Online]. [no date]. [Accessed 20 April 2023]. Available from: https://www.ibm.com/topics/neural-networks
24. Richter, T. Data Noise and Label Noise in Machine Learning. 1 July. Towards Data Science. 2021. [Online]. [Accessed 24 April 2023]. Available from: https://towardsdatascience.com/data-noise-and-label-noise-in-machine-learning-98c8a3c8322e
25. Deepchecks. Noise in Machine Learning. [Online]. [no date]. [Accessed 19 April 2023]. Available from: https://deepchecks.com/glossary/noise-in-machine-learning/
26. Gupta, P. Cross-Validation in Machine Learning. 5 June. Towards Data Science. 2017. [Online]. [Accessed 25 April 2023]. Available from: https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f
27. Kumar, A. Model Complexity & Overfitting in Machine Learning. 29 May. Vitaflux. 2022. [Online]. [Accessed 25 April 2023]. Available from: https://vitalflux.com/model-complexity-overfitting-in-machine-learning/#What_is_model_complexity_why_it%E2%80%99s_important
28. Wikipedia. Mean absolute error. [Online]. 2023. [Accessed 19 April 2023]. Available from: https://en.wikipedia.org/wiki/Mean_absolute_error
29. Indeed. What Is MAPE? A Guide to Mean Absolute Percentage Error. [Online]. 2023. [Accessed 19 April 2023]. Available from: https://www.indeed.com/career-advice/career-development/what-is-mape
30. Wikipedia. Mean squared error. [Online]. 2022. [Accessed 19 April 2023]. Available from: https://en.wikipedia.org/wiki/Mean_squared_error
31. Mishra, A. Metrics to Evaluate your Machine Learning Algorithm. 24 February. Towards Data Science. 2018. [Online]. [Accessed 18 April 2023]. Available from: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234
32. Glen, S. Mean Absolute Percentage Error (MAPE). [Online]. [no date]. [Accessed 19 April 2023]. Available from: https://www.statisticshowto.com/mean-absolute-percentage-error-mape/
33. Li, Y., Zhou, A., Ma, X., and Wang, S. Profit-aware Edge Server Placement. IEEE Internet of Things Journal. 2022, **9**(1), pp.55-67.
34. Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A., and Shen, X. Delay-aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach. IEEE Transactions on Mobile Computing. 2021, **20**(3), pp.939-953

35. Sguangwang.com. The Telecom Dataset (Shanghai Telecom). [Online]. 2018. [Accessed 3 February 2023]. Available from: http://sguangwang.com/TelecomDataset.html
36. Great Learning. Hyperparameter Tuning with GridSearchCV. [Online]. 2022. [Accessed 20 April 2023]. Available from: https://www.mygreatlearning.com/blog/gridsearchcv/
37. Scikit-learn. sklearn.linear_model.LinearRegression. [Online]. [no date]. [Accessed 5 March 2023]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression
38. Scikit-learn. sklearn.svm.SVR. [Online]. [no date]. [Accessed 5 March 2023]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR
39. Scikit-learn. sklearn.neighbors.KNeighborsRegressor. [Online]. [no date]. [Accessed 5 March 2023]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html
40. Scikit-learn. sklearn.model_selection.train_test_split. [Online]. [no date]. [Accessed 3 March 2023]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
41. Scikit-learn. sklearn.model_selection.GridSearchCV. [Online]. [no date]. [Accessed 14 March 2023]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV
42. Scikit-learn. sklearn.metrics.mean_absolute_error. [Online]. [no date]. [Accessed 10 March 2023]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html#sklearn.metrics.mean_absolute_error
43. Scikit-learn. sklearn.metrics.mean_squared_error. [Online]. [no date]. [Accessed 10 March 2023]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html#sklearn.metrics.mean_squared_error
44. Scikit-learn. sklearn.metrics.mean_absolute_percentage_error. [Online]. [no date]. [Accessed 10 March 2023]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_percentage_error.html#sklearn.metrics.mean_absolute_percentage_error
45. pandas. pandas.read_excel. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html
46. matplotlib. Matplotlib.pyplot.show. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

47. matplotlib. Matplotlib.pyplot.plot. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.show.html
48. NumPy. numpy.zeros. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.zeros.html
49. NumPy. numpy.linspace. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.linspace.html
50. NumPy. numpy.mean. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.mean.html
51. NumPy. numpy.std. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.std.html
52. NumPy. numpy.polyfit. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html
53. NumPy. numpy.poly1d. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.poly1d.html
54. NumPy. numpy.sqrt. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.sqrt.html
55. NumPy. numpy.array. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.array.html
56. NumPy. numpy.append. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://numpy.org/doc/stable/reference/generated/numpy.append.html
57. Openpyxl. Documentation. [Online]. [no date]. [Accessed 28 April 2023]. Available from: https://openpyxl.readthedocs.io/en/stable/
58. BCS. Code of Conduct. [Online]. [no date]. [Accessed 01 May 2023]. Available from: https://www.bcs.org/media/2211/bcs-code-of-conduct.pdf

Appendix A
Self-appraisal

A.1 Critical self-evaluation

Overall, I am pleased with what I have achieved with this report. I feel the results and insights about the performance of the K-Nearest Neighbour model have advanced the understanding of machine learning based workload prediction models for edge computing. These results have been contextualised with the performance of Support Vector Regression and Linear Regression to determine the effectiveness of the K-Nearest Neighbour algorithm. Finally, the use of the sliding window technique has provided extra data on the use of K-Nearest Neighbour regression for prediction mechanisms.

I feel the coverage of the literature was sufficient and was explained to a deep enough level where a general computer science audience could follow the points referenced from the literature. A further examination of the literature could have deepened the project and provided further insights into the existing literature and final results; however, I feel that the core of the literature has been covered and provides the necessary information and context to support the conclusions in this report.

The key areas of improvement in this project would have been tuning the KNN model for each type of training data, this would help determine the upper limit of the effectiveness of KNN in this application in increasing and fluctuating data, testing different time intervals (not just minute intervals) and training-testing splits, and finally by directly comparing the performance of KNN against Neural Networks. This direct comparison would have provided deeper insights into how effective KNN is for workload prediction and help determine the optimal algorithms for workload prediction systems. One final addition would have been the use of cross-validation for the LR and SVR algorithms. This would ensure the models do not suffer from overfitting, however the results from the testing data indicate that overfitting was not an issue as both algorithms performed well.

A.2 Personal reflection and lessons learned

One of the key lessons I learnt from this process was the importance of continuous and sustained effort. This project has run through the entirety of my third year at university, and the overall workload was considerable. A more balanced approach to work throughout the year would have improved the final report, covered some of the areas of improvement, and ensured that the final weeks were less work intensive.

Another lesson learned was in the skills of report writing. As computer science is primarily a coursework and test-based course, the skills required for report writing have not played a large part of my course up until this year. This module and report have been a great opportunity to develop the skills required for report writing and will be useful into the future.

Finally, the utility of planning. Planning on what to achieve on small, medium, and large scales can be very useful in larger projects. Having clear aims for the day, week, and months ensures progress is steady, and ensures workload is balanced across the duration of the project. I used planning fairly well in this project, particularly once entering the second semester. This meant that despite lots of work in the final weeks, the workload was always manageable, and the final product was of a good quality.

A.3 Legal, social, ethical, and professional issues

A.3.1 Legal issues

This project has no legal issues to consider. The dataset used in this project, The Shanghai Telecom dataset, is available free of charge for education and non-commercial purposes. All work is referenced appropriately, and nothing prohibited by the requirements of the report, or the university, have been breached.

A.3.2 Social issues

This project has no impact on social issues due to it being an examination of a technical problem on edge computing systems. This means that social issues do not need to be covered in this report.

A.3.3 Ethical issues

This project has no ethical issues to consider. The findings of this report have no effect on any ethical considerations and cannot be used to harm anyone. All data used is anonymous and cannot be used to identify any individuals.

A.3.4 Professional issues

All work undertaken in this project follows the British Computing Society (BCS) code of conduct (58). This ensured the project was conducted in a professional way throughout and so has no issues to consider.

Appendix B
External Materials

B.1 Dataset

- The Shanghai Telecom Dataset.

B.2 Python Libraires

- Scikit-learn (37-44).
- Pandas (45).
- Matplotlib (46-47).
- Numpy (48-56).
- Openpyxl (57).