EP2420 Project 1
Estimating Service Metrics from Device Measurements

Shriya Bhatija

November 27, 2022

Project Overview

In this project, different regression models were trained in order to predict service-level metrics from infrastructure measurements. The service is either a video-on-demand system (VoD) or a key-value store (KV) with respective service-level metrics video frame rate and response time experienced by the client. Moreover, for each service a load generator was built that controls the load on the server [1]. In particular, for the VoD application it controls the number of active VoD sessions and for the KV store it manages the KV operations issued per second [1]. Moreover, there are two load patterns, namely the *periodic*-load pattern as well as the $flashcrowd$-load pattern; for more details see [1]. Here, the flashcrowd-load pattern is the one of interest.

The project is divided into five tasks with each of those having different emphasis'. However, the overall objective is to apply several machine learning models in conjunction with various pre-processing techniques and evaluate their performances. This report aims to describe the results obtained.

Background

Before stating the results from the individual tasks, I will briefly describe animportant concept in the context of machine learning, namely the $train - validation - test\ split\ technique$ as in [2].

In supervised learning, given a labelled dataset the main incentive is often to train a suitable model that fits the dataset and generalises well to unseen instances. In particular, a model needs to be trained, hyperparameters need to be tuned and the final model has to be evaluated. This is when applying the train-validation-test split is going to be useful. The general idea is as follows: Using the same dataset for model training and model evaluation might make the model inherit unwanted bias so we distinctly split the original dataset into a training set, a validation set and a test set. An appropriate splitting ratio may differ from case to case but 60% - 20% - 20% and 80 % - 10% - 10% splits are often used in practice.

The training set is then employed to train the model, i.e. to find the optimal model parameters such that a pre-defined loss function over the training set is sufficiently small. The validation set is then used to evaluate the previously obtained model in an unbiased manner and to perform hyperparameter tuning. Depending on the validation loss, one may want to change the hyperparameter settings for another model training. In the case of neural networks, the validation set can even be utilized simultaneously to model training. In this case, after each epoch the validation loss will be evaluated along with the training loss such that overfitting can be detected and early stopping may be deployed. Finally, the test set will be employed to evaluate the final model on unseen data. As a general note, it is important to remember that all the splits should have a similar underlying distribution as otherwise the validation and test set will be unsuitable for evaluating the model obtained from the training set.

In this project we want to fit several regression methods and evaluate their performances after having applied different data pre-processing techniques to the orginal measurements. In order to evaluate the performances of the regression methods, a train-test split with splitting ratio 70%-30% will be applied. When training the neural network, a portion of the training data will be set aside for a validation dataset and the validation loss will be evaluated after each epoch. This will help monitor the convergence of the neural network.

Task I - Data Exploration and Pre-processing

The aim of this task is to decribe the dataset and to perform different pre-processing techniques that will be needed later on. As indicated in the overview section, this project deals with two datasets, one corresponding the KV store service and another to the VoD service. They are called JNSM_KV_flashcrowd_1 and JNSM_KV_flashcrowd_1, respectively. For each dataset two csv files were given - one consisting of measurements and the other consisting of the target values. A list of such measurement pairs is called a trace. The traces here are based on observations collected once per second during several hours from a running system. The service metrics of interest for the KV store service is *ReadsAvg*, which is the average read latency for obtaining responses over a set of operations per second. In the case of the VoD service, the service metric of interest is *DispFrames* which is the number of displayed video frames per second.

Task I.1: Describing the dataset

As a first step the csv files need to be read and the values may be stored into arrays for future computations. Here, it is important to consider that the first two columns in the original files are only intended to index the data samples as well as the target values and therefore may be removed. The number of samples in the datasets is equal to the number of measurements whereas the number of features corresponds to the number of columns of the measurements. See Table 1 for the results.

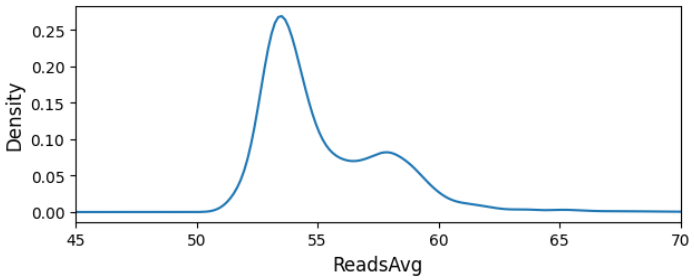
Next, we want to compute the means and standard deviations of the service metrics. For this, one can use the built-in np.mean() and np.std() functions in the numpy library. See again Table 1.

	No. of samples	No. of features	mean service metric	std service metric
KV store service	9500	1723	55.25	2.75
VoD service	17500	1670	22.05	4.33

Table 1: Some characteristics of the datasets JNSM_KV_flashcrowd_1 and JNSM_VoD_flashcrowd_1.

Further, the density curves for the target values of both datasets have been plotted in Figure 1 below:

(a) Density plots for the target values of the KV service.



(b) Density plots for the target values of the VoD service.

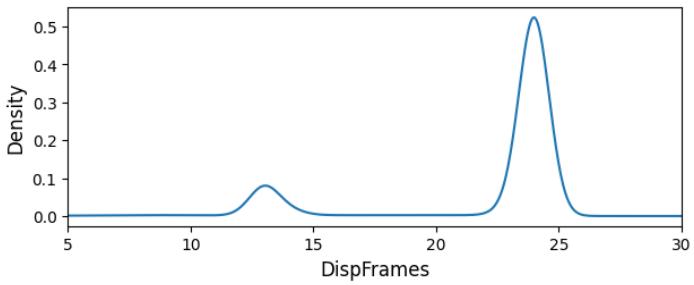


Figure 1: Density plots for the target values ReadsAvg and DispFrames.

The following observations can be made: The values for ReadsAvg are sharply concentrated near its mean. There is a tendency to deviate from the mean, however it is a small deviation which can also be noticed by observing that its standard deviation is quite low. For DispFrames the density plot suggests that the values are concentrated at two main regions, one being at approximately 24 and the other around 13. However, at the value 24 there is a higher density, meaning that most of the values for DispFrames are distributed close to the value 24.

Task I.2: Pre-processing the data

Now, different transformations of the original measurements will be conducted using functions from sklearn.preprocessing. Each of the following three functions is applied once to the feature columns and then again to the sample rows resulting in six new design matrices for each dataset.

(a) $l2$-normalization: We want to normalize the feature columns and sample rows of both datasets to unit length. For this task one can use the normalize() function and apply it to the original design matrices. Note that this function acts row-wise. Hence, if one wants to normalize the feature columns, the transpose is required.

(b) Restriction to interval: The task now is to map the feature columns and sample rows into the interval $[0, 1]$ which can be done using the MinMaxScaler() function. This function acts column-wise, so if one wants to apply it to the sample rows, the transpose is again required.

(c) Standardization: For the last part of this task, we want to standardize the feature columns and sample rows to mean 0 and unit variance. StandardScaler() is the suitable function for this task. It again acts column-wise, so for scaling the sample rows one needs to apply the function to the transpose of the original design matrix.

Task I.3: Reducing the feature matrices

The aim of this task is to reduce the original measurements by selecting the top 16 features using tree-based methods. I.e. we want to obtain the 16 features that have the highest correlation with the target values of interest. Using the DecisionTreeRegressor() and SelectFromModel() functions one can fit a regression tree to the data and select the top 16 features according to the model obtained. Note that we fit the data only using the target values of interest. The top 16 features are named after their position in the design matrix starting from 0. At the end, the transform() function gives the new and reduced measurements consisting of only those 16 feature columns. After this, I then went on to concatenate the new design matrices with the corresponding target values and applied the corr() function to obtain the correlation matrices. These can then be used to plot the heatmaps, see Figure 2.

Figure 2: Heatmaps for the correlation of features and target values.

Heatmap corresponding to the KV store service:
Observe that there are some features which are almost entirely uncorrelated to the target values as well as to every other of the previously selected features. Further, there is a set of features which have almost perfect correlation amongst each other, meaning that the measurements of those features probably behave very similarly. Also note that some features have almost no correlation with the target values whereas other features have a moderately high positive correlation to the target values.

Heatmap corresponding to the dataset JNSM_VoD_flashcrowd_1:
This heatmap shows some interesting patterns: On the one hand there are many pairs of features that are almost entirely uncorrelated and on the other hand many pairs which have a moderately high positive correlation. Further there is one feature which has a negative correlation with most other features. Almost all features have a negative correlation to the target values of "DispFrames". There are even some, that have very little correlation to the values of "DispFrames", suggesting that the features that were not selected will probably be almost entirely uncorrelated to this specific target score.

Task II - Estimating Service Metrics from Device Statistics

In order to estimate service metrics from the device statistics, different regression models will be trained and evaluated on both datsets. In particular, we will deal with linear regression, random forest regression and neural networks.

Task II.1: Model training

To complete this subtask, I trained three different models on the training set (produced in task I.3) using the methods linear regression, random forest regression, and neural network regression.

Linear regression is straightforward to train since it does not require any hyperparameter selection. The training is also quite fast since we are dealing with a very simple model with not too many parameters.

In the case of random forests, our hyperparameters of interest are the number of trees to be trained and the maximal depth of each tree. Using randomized search, I found that good hyperparameters for each dataset could for instance be: `JNSM_KV_flashcrowd_1`: {'n_estimators': 24, 'max_depth': 8} and `JNSM_VoD_flashcrowd_1`: {'n_estimators': 18, 'max_depth': 6}. Hence, I trained a random forest for each dataset with these hyperparameters.

Lastly, I trained a three-layer feed forward network where all of the hidden layers have relu activation and a total of 10 neurons each. The networks will be trained using the Adam optimization technique and the mean squared error as a performance indicator. Here, there are a lot of hyperparameters one could optimize over (e.g. number of layers, activation function, learning rate, batch size, epochs,...). In order for a randomized search to be successful here, one would need a lot of iterations which is computationally expensive. Therefore, I decided to continue by doing trial and error in order to obtain a good set of hyperparameters. I found that a necessary convergence comes along with the number of epochs being at least 5000. However, I further implemented early stopping to avoid passing through epochs once convergence has stopped. This will not let the model overfit to the training data and lead to a better generalisation.

Task II.2: Model training with validation split

In order to evaluate a model on its ability to generalise well, one needs to test its performance on unseen data. For this we apply a train-test split where 70% of the data will be used for training and the remaining 30% will be set aside for testing. I went on to train the three different regression models on the training set in the same fashion as described in the previous section. Using the obtained models, one can make predictions for the target values in the test set. As a next step, we will utilize an appropriate metric to compare these estimations with the ground truth values.

Task II.3-4: Model Accuracy and Evaluation

After the the three different models have been trained on the training set, they now need to be evaluated on the test set. The estimated values will be compared with the ground truth values in terms of the

normalized mean square error (NMAE), see the table below:

	Linear Regression	Random Forest	Neural Network
KV store service	0.0198	0.0178	0.0156
VoD service	0.1121	0.0819	0.0768

Table 2: NMAE between ground truth and predictions for different models.

We make the following observations: The random forest regressor outperforms linear regression. However, this is not surprising since random forests are not limited by an underlying linear model such as linear regression is. In fact, one could even expect that it may perform significantly better than a linear regression model given "better" hyperparameters. Further, observe that the performance of the neural network even exceeds the random forest regressor. This is also not surprising since neural networks have a high capacity in terms of model fitting. Moreover, all regression methods perform worse on the VoD trace as compared to the model performances on the KV trace. It implies that it might be difficult to learn the behavior of DispFrames which may even suggest that DispFrames is random in nature. However, to validate such a claim further investigations will need to be conducted.

Task II.5: Naive Method

In this task, I implemented a naive model, in which for each sample it predicts a constant value which is the mean of the target values in the training set. To evaluate this model's performance, the NMAE was applied on the test set and the predictions. The scores are displayed below:

	Naive Method
KV store service	0.0379
DispFrames	0.1426

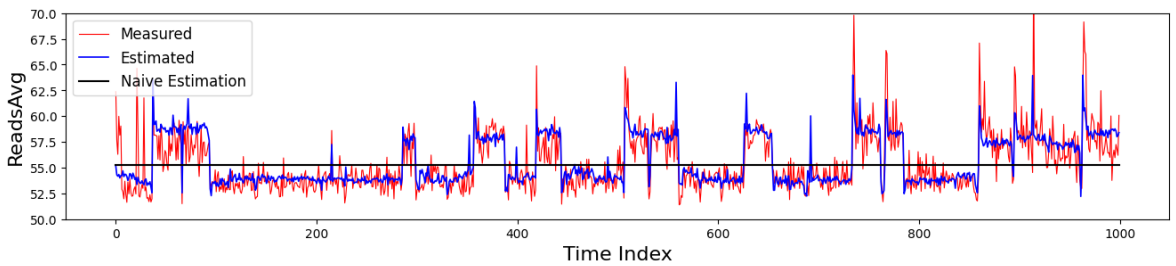
Table 3: NMAE for the naive method.

Observe that in comparison to the other methods the naive method generally performs poorer throughout all target scores which is to be expected since it does not involve any meaningful learning.

Task II.6: Time series plot

For this subtask I chose the linear regression model and plotted the time series of the measurements and of the model estimations for the target values on the test set up to the 1000th sample. There is one plot corresponding to each target score, see Figure 3. We observe that in all cases the naive method is pretty poor at estimating the target values which is of course to be expected. Further, the plots suggest that linear regression is able to predict the target values for ReadsAvg better than for DispFrames. This observation can also be confirmed by comparing the respective normalized mean absolute errors, as displayed in Table 2. This may be explained by the fact that linear regression is only able to fit a linear model but the values for DispFrames behave in a more complex manner that a linear model is only able to capture very poorly.

(a) KV store service



(b) VoD service

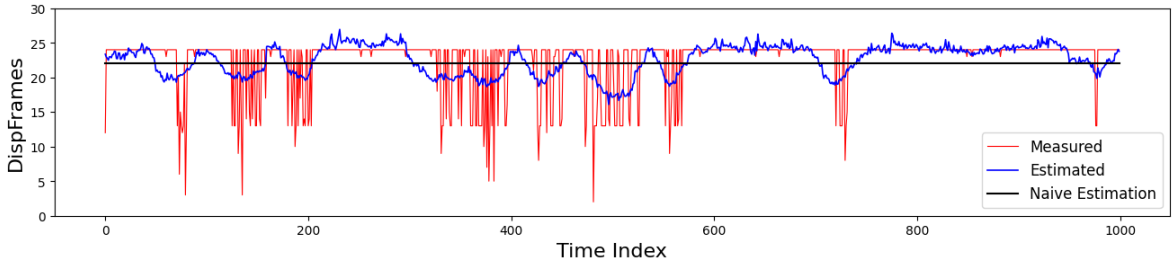


Figure 3: Time series of predicted and measured target values. The predictions are based on a linear regression model.

Task II.7: Density plots and histograms of the target values

In this task, one should produce a density plot and a histogram for the target values of the test set. This can help us understand the test split in a better way. Generally, we would like for these densities to be similar to the ones of the orginal datasets, as explained in the section about train-validation-test split. For better visualisation, each pair of density function and histogram is plotted into one figure, see Figure 4. One can make the following observations:

The density of the values for ReadsAvg has two peaks - one higher peak at approximately the value 53 and a lower one at the value 58. This roughly aligns with the distribution on the original dataset which is what we want. For DispFrames the density plot suggests that the values on the test set are concentrated at two main regions, one being at approximately 24 and the other around 13. However, at the value 24 there is a significantly higher density, meaning that most of the values for DispFrames are distributed close to the value 24. Hence, the distribution of DispFrames is very similar to the one on the original dataset.

(a) KV store service (b) VoD service

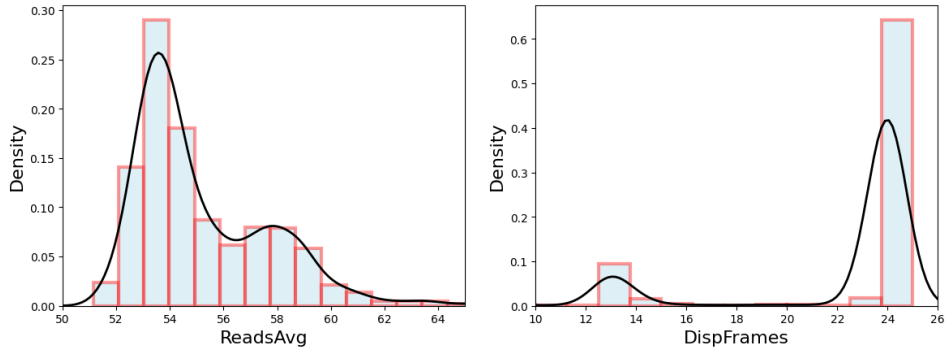


Figure 4: Density plots and histograms for the target values.

Task II.8: Density plots for estimation errors

In a previous task the normalized mean absolute errors were computed which give us an indication as to how much the estimations deviate from the measurements on average. In order to have clear picture about the actual distribution of the estimation errors, they have been plotted in below figure. Here, estimation error translates to the absolute error between measurement and estimation. Observe that the densities in the case of the neural network are quite wide and not sharply concentrated at zero as one would hope. Further the estimation errors regarding DispFrames are very differently distributed depending on the model.

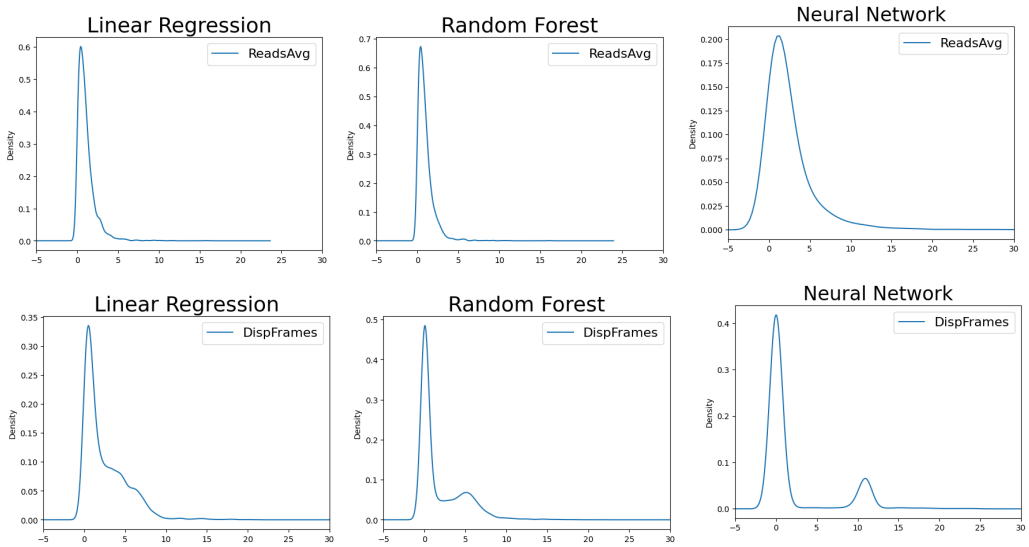


Figure 5: Density plots of estimation errors corresponding to different models.

Task II.9: Discussion

Taking all the above results into consideration, it is pretty astonishing that linear regression and random forests are able to achieve such good predictions, in particular for the dataset JNSM_KV_flashcrowd_1. It goes to show that sometimes the simplest models can yield satisfactory results. However, in general it worth noting that the the neural network performs best while taking the longest time to train, see Table 4. This is not surprising since the model has to go though 5000 epochs with a batch size of 64. It illustrates that for a neural network on a relatively very small and compact dataset may require quite a bit of training time. The ability for a neural network to outperform simple parametric models is present but comes at a cost of computational time.

	Linear Regression	Random Forest	Neural Network
KV store service	0.0331s	0.6275s	48.37s
VoD service	0.0132s	1.1545s	67.43s

Table 4: Computational overhead for the different models.

Task III - Studying the Impact of Data Pre-processing and Outlier Removal on the Prediction Accuracy

This tasks aims to investigate to which degree data-preprocessing and outlier removal may improve model performance. Data-preprocessing is an important step since certain models are sensitive to fluctuations in the data. Further, real world data is often noisy and there may even be measurement errors. In such a case outlier removal can be an important technique in order to make the models more robust.

Task III.1: Pre-processing

In Task I we performed six different forms of pre-processing to the design matrix of the original device measurements. Particularly, there were three different methods, each applied to both the feature columns and the sample rows:

(i) l^2 Normalization,

(ii) Restriction to interval [0,1],

(iii) Standardization to mean zero and unit variance.

Therefore, together with the unprocessed measurements, we have seven different design matrices. In the following task each of those pre-processing techniques will be evaluated in terms of the regression methods of interest.

For brevity, he following notation will be used:

X_1 : Design matrix of unprocessed device measurements;
X_2 : Design matrix of column-wise l^2-normalization;
X_3 : Design matrix of row-wise l^2-normalization;
X_4 : Design matrix of column-wise interval restriction;
X_5 : Design matrix of row-wise interval restriction;
X_6 : Design matrix of column-wise standardization;
X_7 : Design matrix of row-wise standardization.

Task III.2: Comparative study of different pre-processing techniques

We aim to investigate, for each of the regression methods, which of the pre-processing techniques performs best in terms of prediction error. For this purpose each regression method - linear regression, random forests and neural networks - will be trained and evaluated in terms of NMAE on all seven measurements.

Hyperparameters

The random forest regressor will be trained using 20 trees and a maximum tree depth of 10, which are hyperparameters that have indicated a satisfying model performance in Task II. As for neural networks, I decided to implement a three-layer feed forward network where all of the hidden layers have relu activation and a total of 10 neurons each. The networks will be trained using the Adam optimization technique and the mean squared error as a performance indicator. Together with a batch size of 64 and 5000 epochs, this network should have a high capacity for fitting the data.

Dataset JNSM_KV_flashcrowd_1

	Linear Regression	Random Forest	Neural Network
X_1	0.01635	0.01614	0.01593
X_2	0.01661	0.01618	0.01597
X_3	0.01633	0.01658	0.01608
X_4	0.01645	0.01611	0.01600
X_5	0.01637	0.01609	0.01611
X_6	0.01635	0.01607	0.01545
X_7	0.01632	0.01736	0.01621

Table 5: NMAE for different combinations of regression methods and pre-processing techniques corresponding to the dataset JNSM_KV_flashcrowd_1.

For the dataset JNSM_KV_flashcrowd_1 observe that on average linear regression performs worst throughout the different pre-processing techniques while the neural network performs best. This might be due to the fact that neural networks have a high model flexibility and a low bias. It is even to to expected that tuning some hyperparamaters - such as tree depth for random forests or number of hidden layers for neural networks - might improve the performance of both regression methods. Further, it can be seen that the pre-processing method of column-wise standardization works well on all regression methods while row-wise standardization seems to be least suitable.

Dataset JNSM_VoD_flashcrowd_1

	Linear Regression	Random Forest	Neural Network
X_1	0.10125	0.06152	0.06106
X_2	0.10712	0.06228	0.05944
X_3	0.10169	0.06934	0.06123
X_4	0.10146	0.06134	0.05821
X_5	0.10137	0.07121	0.06228
X_6	0.10129	0.06138	0.05802
X_7	0.10172	0.07076	0.06471

Table 6: NMAE for different combinations of regression methods and pre-processing techniques corresponding to the dataset JNSM_VoD_flashcrowd_1.

For the dataset JNSM_VoD_flashcrowd_1 we observe again that on average linear regression performs worst throughout the different pre-processing techniques while the neural network performs best. Compared to the previous dataset, here we can see higher error rates which indicate that the regression task might be more difficult on JNSM_VoD_flashcrowd_1. In addition to that, note that the error is significantly lower for the non-linear models indicating that a linear model might be a poor choice for this particular dataset. Further, the pre-processing method of column-wise standardization works well on all regression methods while row-wise standardization seems to be least suitable.

Hence, for both datasets column-wise standardization is the most promising pre-processing method.

Task III.3: Outlier removal

Outlier removal can be an important pre-processing tool that may enhance model performance. We call a sample an outlier when one of its components has an absolute value larger than a given threshold T. In this task I will compute and plot the number of outliers as a function of T. The outlier detection will be applied to the design matrix whose feature columns are standardized. The following plots are obtained for each dataset:

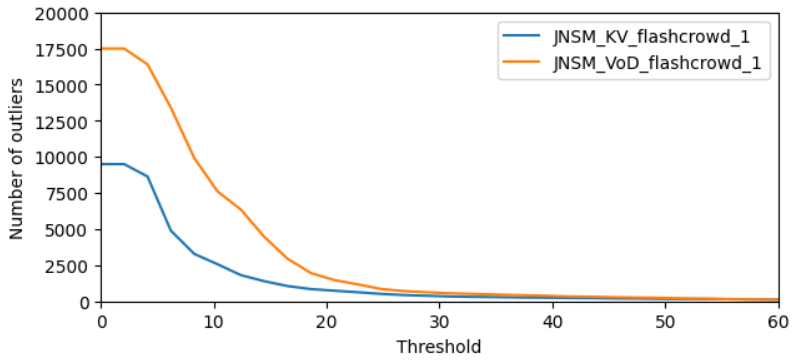


Figure 6: Number of outliers given different thresholds.

Observe that the number of outliers decreases with increasing threshold values. This is to be expected since the columns were standardized and therefore have mean zero and unit variance. The number of outliers in the dataset JNSM_VoD_flashcrowd_1 decreases faster since it starts out with more samples.

Task III.4: Effects of outlier removal on model performance

In this task we want to investigate the error of the linear regression and random forest models in function of the threshold T. I start out with the dataset where the feature columns are standardized and from there produce reduced datasets $S_1, \cdots, S_{10}$ for $T = 10, 20, 30, ...100$. For each S_i linear

regression and random forest methods will be applied and the resulting models will again be evaluated in terms of NMAE. The resulting plots for the dataset JNSM_KV_flashcrowd_1 are shown below:

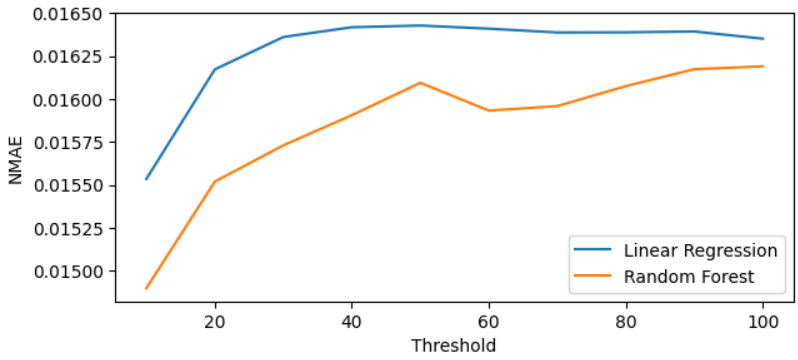


Figure 7: NMAE prediction error for the dataset JNSM_KV_flashcrowd_1 given different thresholds.

Observe that for every threshold value the error corresponding to the random forest regressor is lower than the one corresponding to linear regression. This is not surprising since tree based methods generally have a lower bias than linear models. Further, the error increases as the threshold value grows which is also to be expected since less and less outliers, i.e. noise, will be removed. The plots seem to plateau which is probably a result of the fact that the number of outliers plateaus with growing threshold values, see Figure 6.

Tak IV - Predicting the Distribution of Target Variables using Histograms

After having trained different regression models in conjunction with several pre-processing techniques, we will now try a different method for predicting the service metrics and evaluate the performance in comparison to the results obtained in task II and task III. The basic idea of this task is to discretize the target space into classes and predict the class with the highest probability according to a random forest classifier.

Task IV.1: Dataset

For this task, the measurements that have been the outcome of pre-processing and outlier removal will be used. In particular, in terms of pre-processing the column wise standardization will be applied since it gives the most promising results in Task III. For outlier removal thresholds that keeps 99% of the samples in the datasets will be chosen. This will enable us to use most of the data for training and at the same time get rid of significant outliers. In order to obtain the corresponding threshold values, I defined a function that counts the number of outliers given a threshold and a dataset. With a bit of trial and error, I obtained a threshold of 69 for the KV store service data and a threshold of 54 for the VoD service data. Outliers corresponding to these thresholds will then be removed. On the resulting datasets a train-test split with a ratio of 80%-20% will be applied for future model fittings.

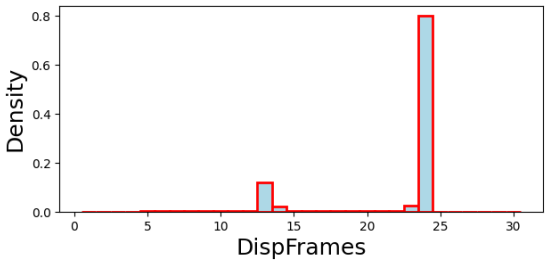
Task IV.3: Discretizing the space of Dispframes

In order to train a random forest classifier, the target spaces of both datasets need to be discretized. For the dataset JNSM_VoD_flashcrowd_1 we are going to consider a histogram on the interval $y \in [0.5, 30.5]$ with a bin size of 1. This results in 30 bins with mid points 1,2,...,30. Later on, each bin of the histogram will be considered as a separate class. For a visualization, see Figure 8.

Task IV.4: Discretizing the space of ReadsAvg

In order to discretize the target space of ReadsAvg, consider a histogram on the interval $y \in [y_{min}, y_{max}]$, where y_{min} is the minimum target value in the training set and y_{max} is the maximal one. Here, we have $y_{min} \approx 51.09$ and $y_{max} \approx 80.53$. The interval will then be divided into 20 bins of equal size. Later, each bin will be considered as a seperate class. For an illustration, see the figure below:

(a) KV store service



(b) VoD service

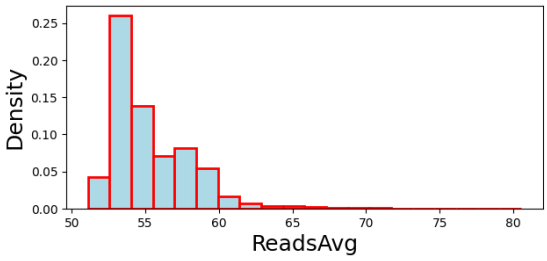


Figure 8: Discretization of the target spaces ReadsAvg and DispFrames.

Note that this gives an estimation for the densities of the target spaces in way such that increasing the number of bins will make the histograms converge to the actual density plots.

Task IV.5: Fitting a Random Forest Classifier

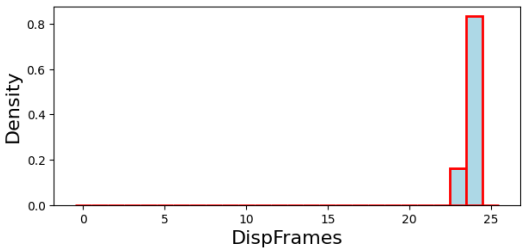
In this subtask, each bin of the histograms will be considered as a separate class and a random forest classifier will be trained to predict the probability for each class. Each target value in the training set as well as in the test set will be assigned to the class corresponding to its bin. If an instance of the test set exceeds the interval of the histogram, it will be assigned to its closest class. In the case of the KV store service, the classes are floating point numbers and therefore a label encoder will be needed to convert the classes into integer values that a random forest classifier can accept. Now, a random forest classifer will be trained on the training set. The built in function predict_proba() of the random forest model is then able to compute the probability of each class for all the training and test instances.

Task IV.6: Accuracy of the Random Forest Classifier

In order to evaluate the accuracy of this model, we need to compute the NMAE of the predicted values with respect to the measured values over the test set. For the dataset JNSM_VoD_flashcrowd_1 a NMAE of 0.070 was computed whereas for the dataset JNSM_KV_flashcrowd_1 we have an NMAE of 0.019. When comparing these values with the results obtained in Tasks II and III, one may conclude that the Random Forest Regressor is more suitable for the KV store service whereas the Random Forest Classifier is more suitable for the VoD Service. This is to be expected since in the case of the KV store service, the bin sizes seem to be too large for being able to make close predictions. The fact that the Random Forest Classifier outperforms the Random Forest Regressor in the case of the VoD service is also not too surprising since the video frame rate is integer valued to begin with and thus a classification model may be a natural choice.

Task IV.7: Illustration of predicted class probabilities

For illustration purposes, I will choose the first two samples from the test set and draw the two predicted histograms. In the case of the VoD service, we have the following two histograms:



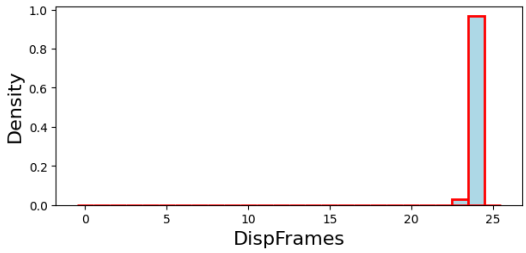


Figure 9: Predicted class probabilities for two samples from the test set of the VoD trace.

Observe that in both cases the highest probability was assigned to the value 24. However, for the first sample the measured target value is 12 which the Random Forest Classifier tried to capture by

assigning some probability to the value 23. One may conduct further investigation to the question as to whether this is an artifact or if the classifier is only able to predict the value 24 for most of the instances. The measured target value for the second sample was actually 24, so the model assigns almost probability 1 to the class 24. This behavior may be a result of the fact that the classes here are highly unbalanced which makes it difficult to learn a reliable model.

Next, the predicted histograms for two samples from the KV trace will be drawn:

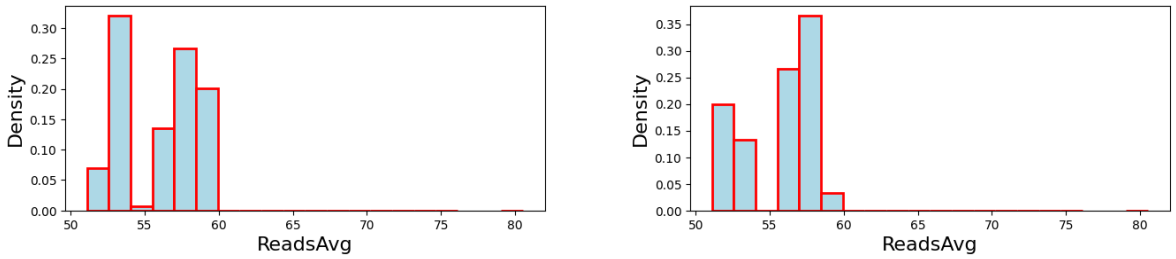


Figure 10: Predicted class probabilities for two samples from the test set of the KV trace.

In this case observe that the probabilities are much more spread out amongst different classes which is to be expected from previous results. The random forest classifier will predict the value 53.29 for the first sample since the probability for that value slightly exceeds the other class probabilities. The measured response time for that sample is 58.49 which makes the prediction quite poor since it did not even get assigned to its bin value. For the second sample the model will predict the value 57.71 whereas the measured response time is 59.98 which is a closer prediction but still does not correspond to the bin of the measured response time. Also note that the probability for the third class is close to zero for both samples. At this point it is difficult to pinpoint the exact reason but further investigations may reveal why this is the case.

Task V - Predicting Percentiles of Target Metrics

The aim of this task is to better understand the distribution of the target metrics in terms of different percentile values. In particular, the histogram estimator from Task IV will be used to predict the 20th, 50th, and 95th percentile values of the target values ReadsAvg and DispFrames.

Task V.2: Background

A quantile $q \in [0, 1]$ of a probability density function $p(y)$ on $\mathbb{R}$ has value $a \in \mathbb{R}$ if $\int_{-\infty}^{a} p(y)dy = P(a)$, where $P(a)$ denotes the cumulative distribution function (cdf) of $p(y)$ [3]. Given q our objective is to estimate the 20th, 50th, and 95th quantiles for the target values. Task IV outputs histograms for the distribution of the target values as well as class probabilities for each sample. In order to compute the percentiles, the histograms need to be converted into a plot of cumulative probabilities.

Let $B = \{b_0, \ldots, b_n\}$ denote the bin edges and let p_i be the cumulative probability on the interval $[b_{i-1}, b_i]$ for $1 \leq i \leq n$. The value $a(q)$ for the quantile q is then estimated by finding the largest $k \in \{1, \ldots, n\}$ where $p_k < q$, and interpolating between the bin edge b_k and the next edge b_{k+1} [3]:

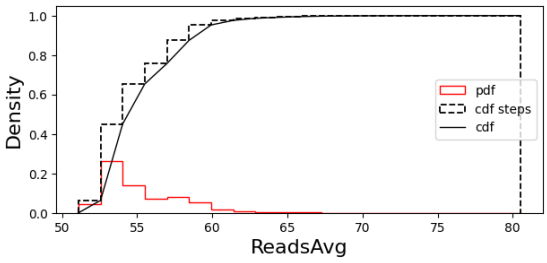
$$a(q) = b_k + (b_{k+1} - b_k)\frac{q - p_k}{p_{k+1} - p_k}.$$

This formula is going to be utilized for the upcoming tasks.

Task V.3: Percentile values

In this subtask, I will compute the 20th, 50th, and 95th percentile values of the target values on the training set used in Task IV. Firstly, let us plot the distribution of the target values:

(a) KV store service



(b) VoD service

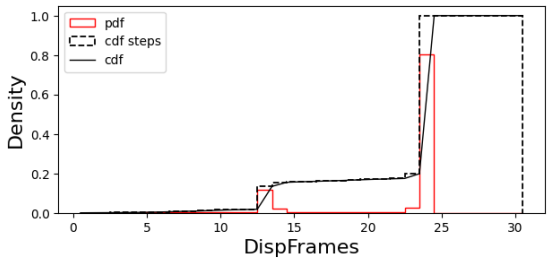


Figure 11: Cdf of the target values ReadsAvg and DispFrames corresponding to the VoD flashcrowd trace and the KV flashcrowd trace, respectively.

Using these visuals, one can compute the 20th, 50th and 95th percentiles as described in the section above. The following values are are obtained:

	$q = 0.20$	$q = 0.50$	$q = 0.95$
KV store service	53.08	54.39	59.40
VoD service	23.50	23.88	24.44

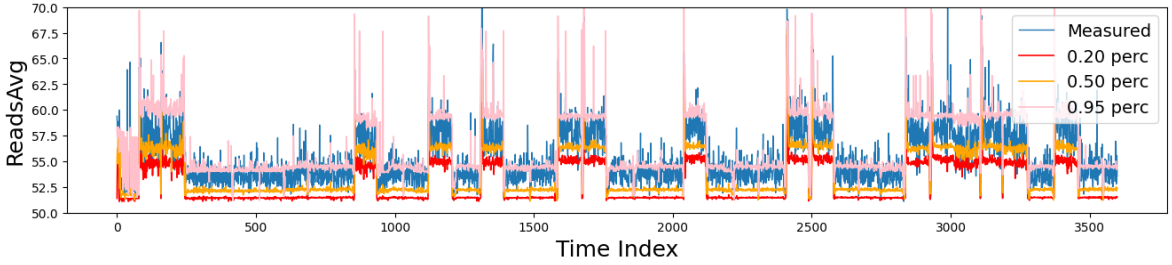
Table 7: Estimated 0.20, 0.50 and 0.95 quantiles of the target values for the KV flashcrowd trace and for the VoD flashcrowd trace.

Observe that for the KV store service the values for all three quantiles are somewhat spread throughout a portion of the target space which is to be expected since the density for ReadsAvg is also spread out and not concentrated in a single region. For the VoD service one observes the opposite. The values for all quantiles are close to the value 24 which is also not too surprising since this is the point of highest density. Further, DispFrames is integer-valued since it represents the number of displayed frames per second. Hence, in reality the value for $q = 0.2$ and $q = 0.5$ is 24 whereas the value for $q = 0.95$ is 25.

Task V.4: Predicted percentile values

In this subtasks, the predicted 20th, 50th and 95th percentiles of the target values for each sample need to be computed. Note that the predicted percentile value for an individual data sample can be obtained in the following way: Firstly, for each sample we can plot a histogram of the predicted class probabilities that were obtained via a random forest classifier as described in task IV. Based on these predicted class probabilities one may compute the necessary percentile values. The following plot shows the predicted 20th, 50th and 95th percentile values together with the measured values for the first 3600 samples of the datasets (corresponding to the first hour of the experiment):

(a) KV store service



(b) VoD service

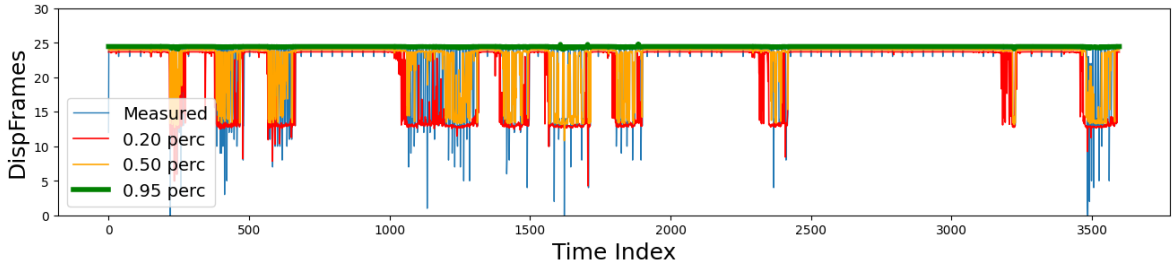


Figure 12: Predicted 20th, 50th and 95th percentile values for KV data trace and VoD data trace.

Note that for the KV service, all percentile values mostly align with the slope of the measurements. This is desired since it means that the model has learned the distribution of ReadsAvg to a certain extent even tough the model performance was unsatisfactory compared to the regression models. Further, the predicted values for the 20th and 50th percentiles seem to be very close to each other implying that for each sample there is only a low density between the target values of these percentiles.

Observe that for the VoD service, the predicted 95th percentile seems to be stable at values of 24 and 25 for almost all displayed data samples. This is to be expected because the measured DispFrames values never exceed the value 25. Further, notice that the 20th and 50th percentiles seem to somewhat align with the measured values. This implies that the random forest classifier is able to assign higher probability to the measured class than in the sample case of task IV.7. This means that even though the classes are highly unbalanced the model is able to learn the underlying distribution of DispFrames to a certain degree.

Task V.5: Accuracy of the predicted percentile values

In order to evaluate the accuracy of the predicted percentile values we make use of the *Glivenko – Cantelli Theorem* (see e.g. [4] and [3]) which states that

$$\text{est}(perc) = \frac{1}{n}\sum_{t=1}^{n} \mathbb{1}(y^{(t)} \leq a_{perc}(x^{(t)}))$$

is a consistent, unbiased estimator for $perc = 0.2, 0.5$ and 0.95 over the entire input space. Here, we take $(y^{(t)}, x^{(t)})$, $t = 1, ..., n$, as the samples of the test set and $a_{perc}(x^{(t)})$ is the predicted percentile value for $x^{(t)}$ given $perc$. For the KV service as well as for the VoD service, the estimators for all percentiles values are displayed in the table below:

	est(0.2)	est(0.5)	est(0.95)
KV store service	0.181	0.238	0.631
VoD service	0.167	0.189	1

Table 8: Estimators for the 0.20, 0.50 and 0.95 percentiles given the KV flashcrowd trace and for the VoD flashcrowd trace.

Note that for the KV service est(0.2) and est(0.5) are quite close to each other which can also be confirmed by exmanining Figure 12. It is surprising that the estimator for $perc = 0.95$ is this low. Further investgation might reveal why exactly this is the case.

We observe that for the VoD trace the estimators for $perc = 0.2$ and 0.5 lie very close to each other while est(0.2) is slightly lower than est(0.5). This is of course to be expected since est($perc$) is increasing w.r.t. $perc$. It is also a given that $a_{perc}(x^{(t)})$ increases for increasing percentile values. Further, it is not surprising that we have est(0.95)=1 since for almost all data samples the 95th percentile has a value greater than 24 and almost all values for DispFrames are less or equal than 24.

Discussion

In this section, the key points of this report will be discussed. Recall that the overall objective was to apply several machine learning models in conjunction with various pre-processing techniques to investigate how model fitting behaves for the KV trace as well as for the VoD trace. In order to get there, task I was about better understanding the datasets and applying some pre-processing techniques. There, it was found that the density of ReadsAvg is somewhat evenly spread whereas the density for DispFrames is mainly concentrated at the value 24. Further, the heatmaps showed that only few features correlate to the target values of interest. In Task II, linear regression, random forest regression and neural networks were trained on both datasets. It was concluded that a neural network performs best on both datasets which is not too surprising since it has the highest model capacity. However, model fitting seemed to be significantly more tricky for the VoD trace. This fact may need further investigation to discover why this is the case. In order to improve model performance, the outlier removal technique was utilized in task III. We found that outlier removal with an appropriate threshold can indeed improve model performance. Moreover, we deducted that the column-wise standardization technique is the most promising pre-processing method throughout all regression models. Task IV was about exploring the possibility of discretizing the target spaces and training a random forest classifier which unfortunately lead to even poorer estimations. Finally, in task V we wanted to better understand the distribution of the target metrics in terms of different percentile values. It was found that the distribution of the target values, especially for Dispframes, is too uneven for good estimations of the percentile values.

Summing up it can be said that model training is a complex process and quite a few techniques may be applied to enhance model performance. Here, we tried certain pre-processing techniques and outlier removal which did indeed bost the model fitting. Particularly, there are many more design choices of the neural network that one could try to further improve the model.

References

[1] Rolf Stadler, Rafael Pasquini, and Viktoria Fodor. Learning from network device statistics. *Journal of Network and Systems Management*, 25:1–27, 10 2017.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[3] Forough Shahab Samani, Rolf Stadler, Christofer Flinta, and Andreas Johnsson. Conditional density estimation of service metrics for networked services. *IEEE Transactions on Network and Service Management*, 18(2):2350–2364, 2021.

[4] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.