EP2420 *Advanced Project Comparing Linear Regression, Random Forest Regression, and Neural Network Regression to estimate Service Metrics*

Vignesh Purushotham Srinivas

November 26, 2021

Project Overview

The primary objective of this project is to develop machine learning techniques to estimate service-level metrics from infrastructure metrics. This project involves training various regression models (e.g. linear regression, random forests regression, neural network regression, etc) to compare their performance for the service-level metrics estimation. Additionally, the project also focuses on feature selection, data pre-processing, outlier removal, and target variable distribution and prediction.

The dataset used for training the models is gathered from a testbed at KTH. The testbed consists of video-on-demand service, key-value store, and load generators [1]. In this report, we primarily focus on the performance estimation of the key-value store. The load generator is capable of generating various traffic patterns.

Background

Feature Selection

With the increased availability of data (mostly due to collaboration and technological developments that ease data collection), a very high-dimensional dataset is not unusual these days. Feature reduction is a crucial process that selects highly correlated and relevant features from a large set of features. Feature selection has many benefits, such as reducing computation time, improving performance, and a better understanding of the underlying model [2].

Linear Regression

Linear Regression is a statistical technique that models the relationship between the input variables and targets by fitting a linear equation over the observed data. The fitted model can then be used as a linear predictor function to estimate the target values for unknown input parameters [3].

The fit model is in the form:

$$Y = aX + b, \tag{1}$$

where X is the input parameter, Y is the target, a is the slope of the predictor line, and b is the Y-intercept. The training process of the Linear Regression model minimizes the residual sum of squares between the input variables and the observed target for every sample in the data. In Equation 2, variables a and b are optimized in order to minimize the error of Y in estimated and observed sample in the dataset.

Random Forest

Random Forest is an ensemble that is employed to solve regression and classification problems. Random Forest ensemble consists of a large number of individual decision trees. Ensemble learning is a technique that combines many classifiers to provide solutions to complex problems. The random forest algorithm computes the outcome based on the predictions of the decision trees. Each individual tree in the random forest computes a class prediction and the class with the most votes becomes our model's prediction or even an averaging mechanism can be employed.

Bagging and bootstrap aggregating are used for training random forests. Bagging is an ensemble algorithm that trains multiple decision trees on different subsets of training data and combines predictions from all trees of the forest. The primary reason behind the outstanding performance of the random forest model as a classification algorithm is that the combined output from multiple uncorrelated models always exceeds the predictions made by individual constituent models.

Some interesting properties of random forests:

1. The precision of the model increases with the number of trees in the forest
2. It does not require aggressive hyper-parameter tuning.
3. Since random forests train multiple trees, it is not affected by over-fitting.
4. Every tree is trained on a random subset of features and arbitrary splitting points.

Neural Network

Neural Network has its inception in attempts of mathematical representing the information processing in biological systems. In biological systems, each neuron receives a signal from the synapses and gives output after processing the signal. Analogous to a biological brain, a neuron in a neural network performs a dot product between the inputs and weights, adds biases, applies an activation function to produce outputs. Many layers of neurons are chained together to form a deep neural network that can learn complex relationships from inputs to produce one/many outputs.

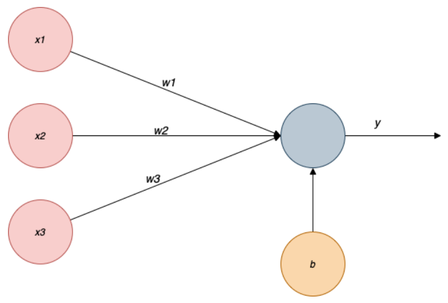

Figure 1: A 2-layer neural network

Figure 1 show a two layer neural network(1 input layer, 1 output layer, 0 hidden layers). The output y for the above neural network is computed as follows:

$$y = \sigma(w^T x + b) \tag{2}$$

where, w^T is the weight matrix, x is the input matrix, and b is the bias term for the neuron. Similarly, outputs from layers can be connected to the more forward layers to form a deep neural network architecture.

Unlike other machine learning models, Neural Networks can be designed for both regression and classification problems. However, tuning the hyper-parameters of the network can be consider as a cumbersome process, and usually requires multiple iteration of grid-searching or baby-sitting for each hyper parameter of the model.

Test bed and Data Description

The dataset used in this project is collected from a testbed located in the KTH lab. The testbed consists of 10 hosts running ubuntu and their clocks synchronized by NTP. Two applications: a VoD service and a Key-Value store, are deployed on the tested. The deployed hosts are interconnected using a virtualized network, consisting of Open vSwitch, and an SDN controller. The testbed also contains clients that responsible for requesting services from the applications and load-generators that can emulate service traffic from different locations in the network [4]. The load generator creates two patterns of traffic:

1. *Periodic traffic:* requests are modeled based on Poissons process whose arrival rate is defined by a sinusoidal function with start value P_S, amplitude of P_A for a duration of 60 minutes.
2. *Flashcrowd-load traffic:* flashcrowd model described in [5] is used to model the traffic with an arrival rate of F_S and peaks at flash events. The flash events occurs over a period of 60 minutes and is follows using a normal distribution.

All experiments conducted in this report uses flashcrowd traffic pattern. The next section discusses the dataset in more detail.

Task 1 - Data Exploration and Pre-processing

The primary objective of this task is to perform exploratory sub-tasks and to understand the above trace better. Some sub-tasks involve feature selection, computing correlation matrix, plotting, prepossessing.

The given traces consist of device metrics (X) collected from the service and routing infrastructure and the service level metric (Y) collected from the clients making service requests. The device metrics consist of approximately 1750 features and 14000 samples. For example, X consists of CPU utilization, memory utilization, context switches, paging information, interrupt rates, I/O statistics, etc. The metrics collected from the OpenFlow switches include TX and RX rates of the network cards. Additionally, all the samples in the traces are timestamped. The target matrix consists of three features: TimeStamp, ReadsAvg, WritesAvg. "ReadAvg" is the average latency in milliseconds for the key-value store to perform a read operation. Similarly, "WritesAvg" is the latency to perform a write operation on the key-value store.

Feature Selection

In order to reduce the number of features present in the device metrics dataset, a tree-based feature selection is used. Specifically, an ExtraTreesRegressor model with 500 estimators is used to select the top 16 features in the dataset. To verify the feature selection a correlation matrix is computed and plotted.

Figure 2 shows the correlation between various features and targets. The axes represent the top 16 features selected in the previous step. It can be inferred that all the selected features exhibit a high correlation with each other. Also, the selected features have at least 70% correlation with the targets of the model.

Data Pre-processing

Since various metrics possess different ranges, normalization or scaling of the data can highly benefit the regression model. At this point, mix-max scaling in applied to the design matrix to make all the samples lie within the range: [0-1]. In the later phases of the project different pre-processing methods are to be applied and compared.

Task 2 - Estimating Service Metrics from Device Statistics

The pre-processed dataset is split into training and testing sets. The training set includes 70% of the samples and the remaining 30% consists the testing set. The training set is used to train the following methods:

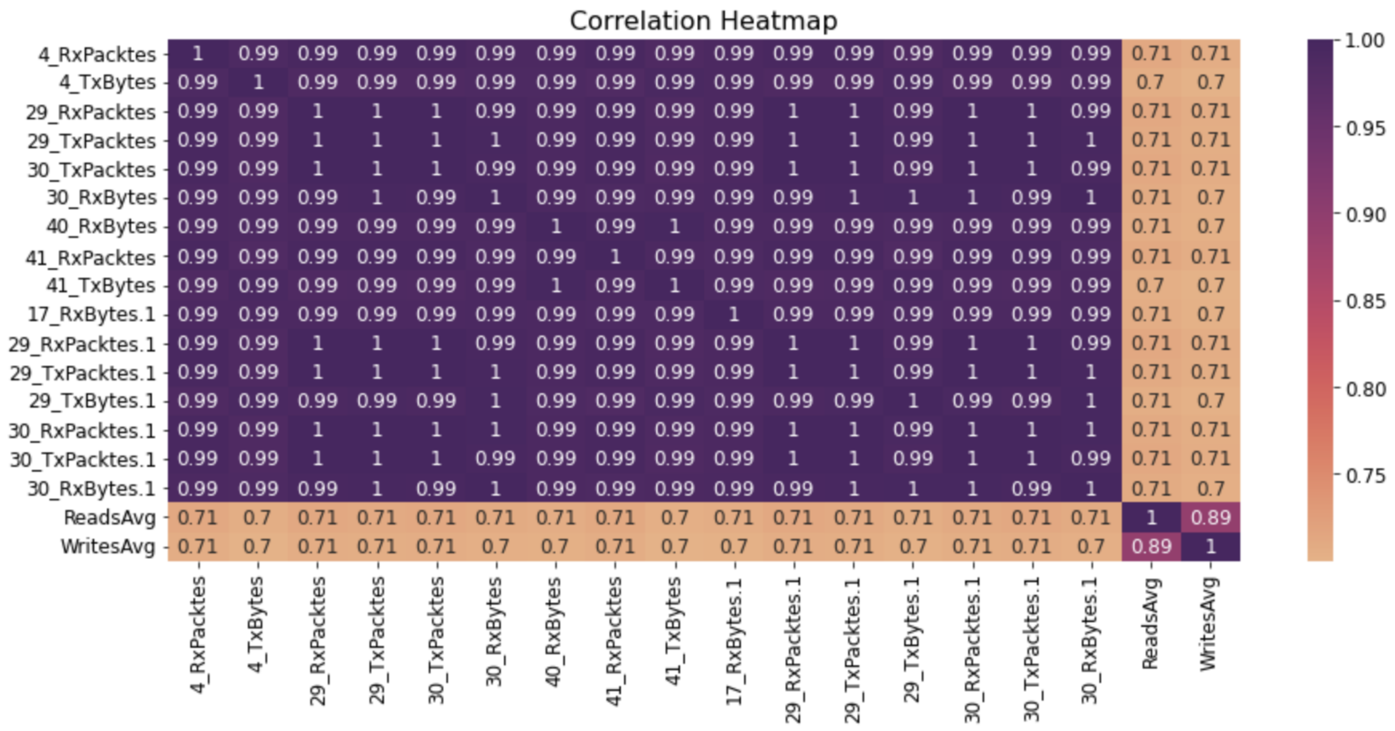

Figure 2: Correlation Heatmap

linear regression, random forest regression, and neural network regression. For each model, the estimation error is computed using *Normalized Mean Absolute Error (NMAE)* and compared.

Method	ReadsAvg	WritesAvg
linear regression	0.0271	0.0279
random forest regression	0.0232	0.0239
neural network regression	0.024	0.0248

Table 1: NMAE for various models

Table 1 show the performance of the models. The estimation error of the the random forest regression and neural network regression are simialar and perform better than the linear regression model.

Method	Hyper parameters	value
random forest	n_estimators	500
	criterion	gini
neural network regression	hidden layers	[100, 50]
	max iter	675
	activation	tanh
	batch size	200
	learning rate	constant (init=0.001)

Table 2: Hyper parameters used for training

Table 2 show the hyper parameters used to train the models models. A basic grid search method was used to search for the best values of hyper parameters. For random forest regression the number of estimators was searched in the list [100, 200, 500, 700]. Similarly, for neural networks the hidden layers were searched in the list [[100, 50, 10], [200, 100, 50], [100, 50], [100, 10]]. If the neural network is deep then the accuracy drops due to vanishing gradients problem. The number of iteration was searched from [200, 600, 1000].

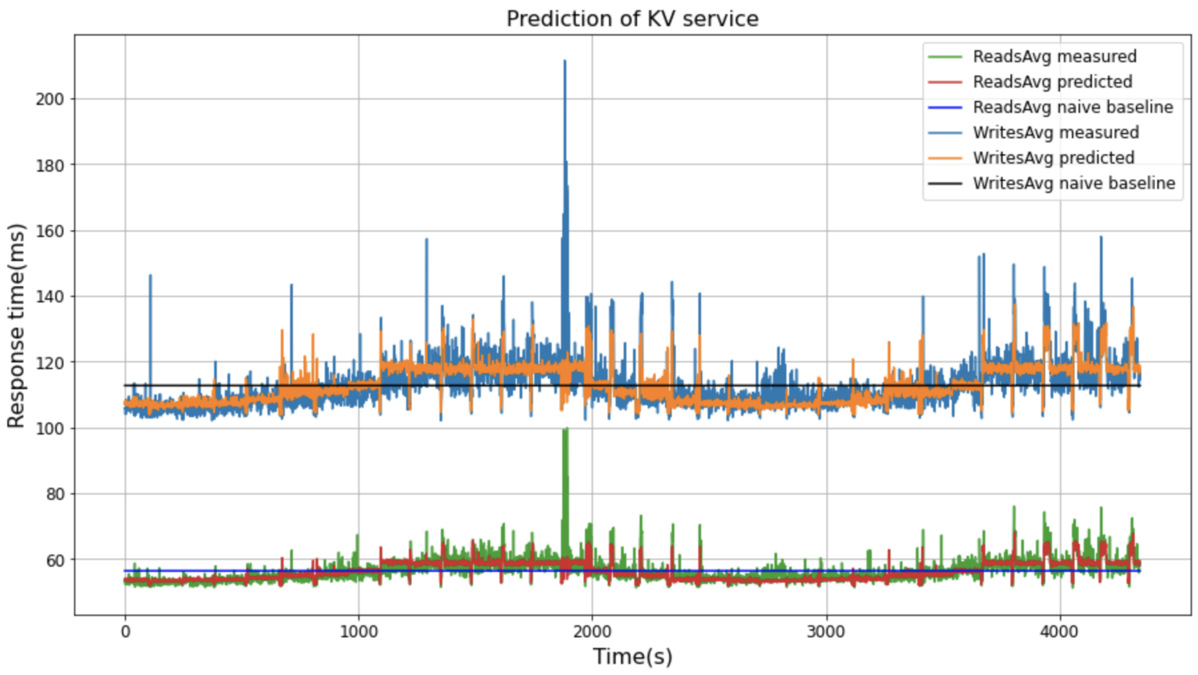

Figure 3: Service metric prediction using random forest regression

Figure 3 shows the estimation of the service metrics of the key-value store using random forest regression with column scaled data. Naive estimator for ReadsAvg = 56.4ms and WritesAvg = 113ms

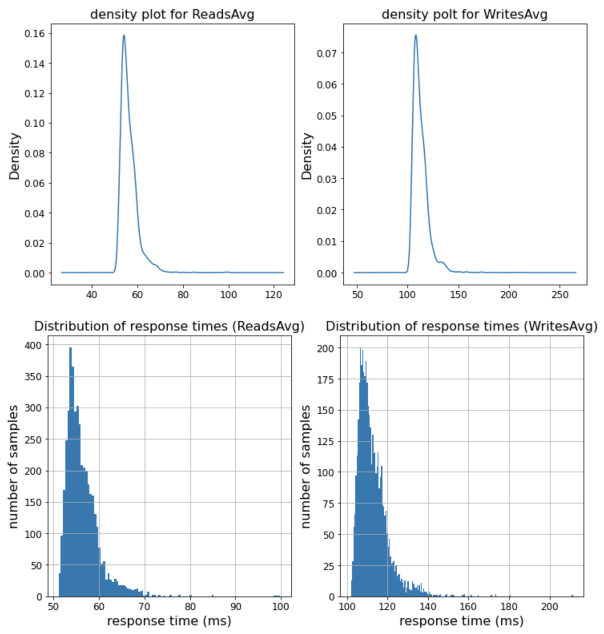

Figure 4: Density plot and a histogram for the target values on the test set (response times).

Figure 4 plots the density distribution of the response time for both ReadsAvg and WritesAvg in the test dataset. It can be inferred that most sample for the ReadsAvg is approximately around 55ms and around 115 ms for WritesAvg.

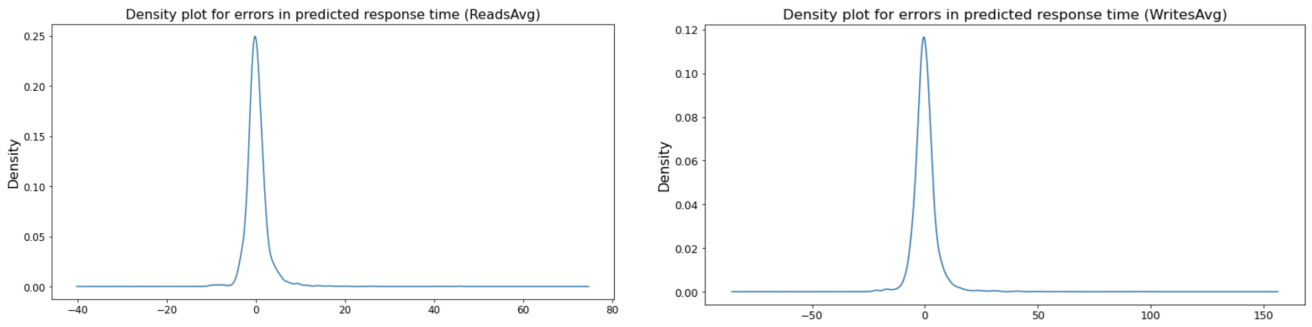

Figure 5: Density plot for estimation errors

Figure 5 show the density plot for the estimation errors when the metrics were predicted using random forest regression. The plots show a narrow spike centered around 0. This shows that the prediction accuracy is high with few errors.

Task 3 - Studying the Impact of Data Pre-processing and Outlier Removal on the Prediction Accuracy

Preprocessing

This section focuses on finding the best pre-processing technique for the device metrics. Pre-processing data is an essential step to ensure the enhanced performance of machine learning models. Since data collection is a loosely controlled process, various features tend to have different limits and ranges. Hence pre-processing step ensures that all the features in the dataset are linearly normalized/scaled/standardized. This work employs different pre-processing techniques as below, and can be applied either row-wise or column-wise:

1. Normalization: linearly scale the values of each feature column/sample row so that its L2 -norm becomes 1.
2. Scaling: linearly map the values of each feature column/sample row so that all they lie within the interval [0,1].
3. Standardization: linearly map the values of each feature column/sample row so that they have 0 mean and a variance of 1.

	linear regression	random forest	neural regression
scaling (row-wise)	0.0462, 0.0477	0.038, 0.0395	0.039, 0.0401
scaling (column-wise)	0.0287, 0.0294	0.0279, 0.0299	0.0266, 0.0273
standardization (row-wise)	0.0468, 0.0485	0.0383, 0.0401	0.0523, 0.0562
standardization (column-wise)	0.0281, 0.0288	0.0275, 0.0285	0.027, 0.0276
normalization (row-wise)	0.0315, 0.0325	0.0303, 0.0318	0.034, 0.0339
normalization (column-wise)	0.035, 0.0361	0.0548, 0.0638	0.0349, 0.036
un-pre-processed	0.0283, 0.029	0.0276, 0.0287	0.0264, 0.0272

Table 3: NMAE(ReadsAvg, WritesAvg) comparison with various pre-processing techniques

From Table 3 it can be seen that, pre-processing data column-wise delivers more accuracy than row-wise pre-processing. While the NMAE of a model trained with standardized and scaled data is similar, the model trained with normalized data is higher. Standardized data used on the three models, delivers better performance, when compared to the other two pre-processing techniques.

Outlier Removal

Outliers are unusual samples in the dataset. Outliers can distort statistical analyses or reduce the accuracy of prediction models. An outlier is a sample whose component has an absolute value that is larger/smaller than a given threshold T.

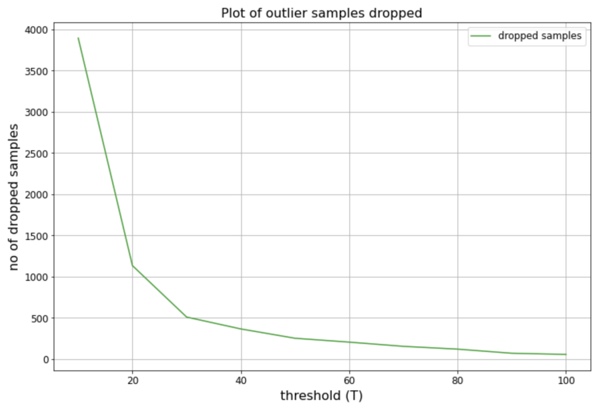

Figure 6: Number of samples dropped with threshold T

Figure 6 show the number of sample dropped with various values of threshold T during outlier removal.

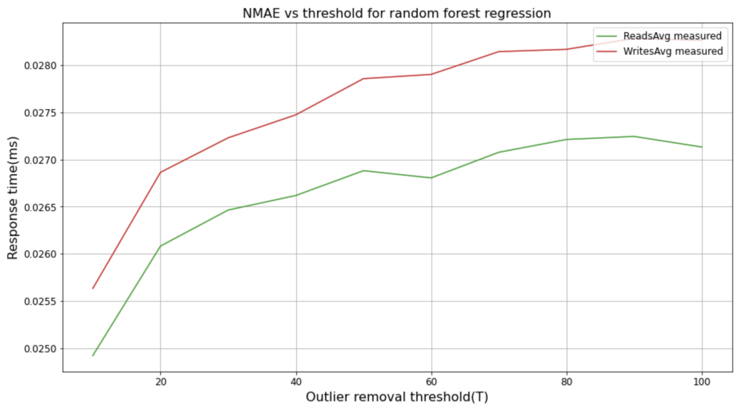

Figure 7: Number of samples dropped with threshold T

Figure 7 show the number of sample dropped with various values of threshold T during outlier removal. The NMAE of the model increases (accuracy decreases) with the number of outliers in the dataset. With the smallest threshold (T=10), approximately 3900 samples are removed, make the dataset cleaner. This accounts to the lower NMAE(highest accuracy) of the regression model.

Task 4 - Predicting the Distribution of Target Variables using Histograms

This task involves discretizing the target space (Y) into smaller fixed classes and then using a histogram predictor to predict the probability of Y given an X. In simpler terms, this technique converts the above-stated regression problem to a classification problem. The response time(Y) for ReadsAvg and WritesAvg is divided into 20 smaller classes, and converted to represent the corresponding classes instead of the real response time using a histogram predictor.

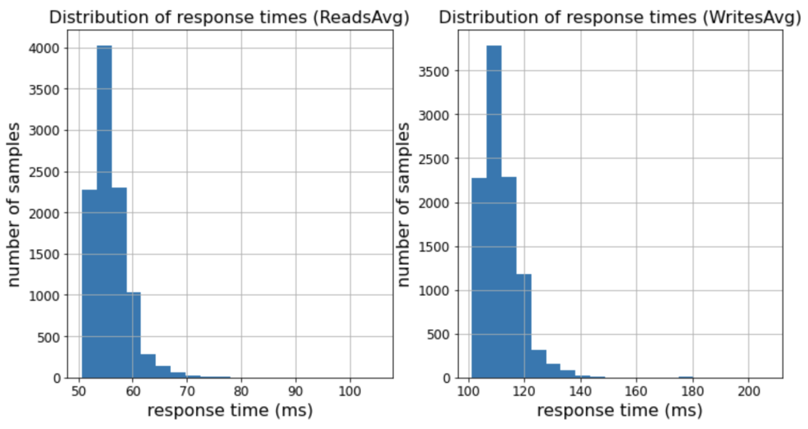

Figure 8: The targets space(Y) discretized into 20 bins

Figure 8 shows the distribution of response times for ReadsAvg and WritesAvg into into different classes(bins=20).

A random forest classifier is trained to predict the density of each class. Figure 9 show the models output for the estimation task. The model is trained on 20 target class with standardized dataset and outlier removal threshold of T=80 (1% of the samples removed from the dataset). The constant lengths of the spikes in the predicted estimate represents the height of the corresponding predicted class. The Naive Estimator reads the following values for ReadsAvg = 54.3ms and WritesAvg = 109ms.

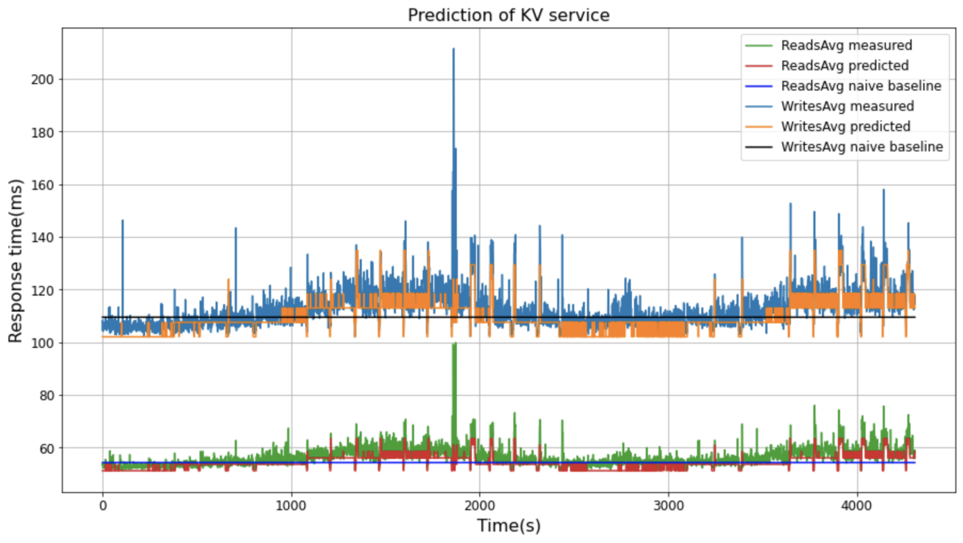

Figure 9: Service metrics estimation from a random forest classification model

	NMAE(ReadsAvg)	NMAE (WritesAvg)
Random forest regression	0.0232	0.0239
Random forest regression with column standardized data	0.0275	0.0285
Random forest classifier	0.044	0.043

Table 4: NMAE(ReadsAvg, WritesAvg) comparison for regression and classification

Table 4 compares the the NMAE for regression vs classification for the service level metrics prediction. Regardless of the high NMAE, the classification model generalize well for prediction problems, and one possibility for the high error rate is that the device metrics usually tend to be very noisy and the predicted classes values are standardized. Its is possible to achieve a better accuracy by increasing the number of classes during binning of target space

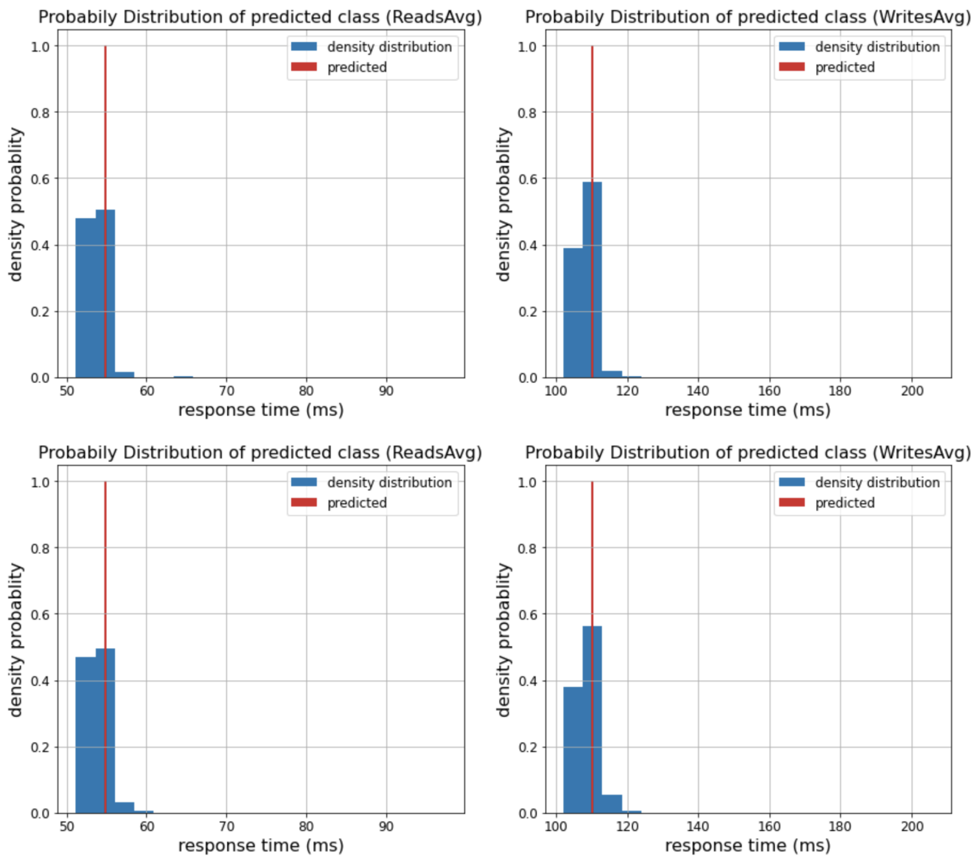

Figure 10: Probablity distribution of classes for 2 sample values from X. The red bars show the predicted classes.

Figure 10 show the density probability of the 20 prediction classes for two randomly selected samples of X. The upper two plots shows the ReadsAvg and WritesAvg for the first sample and the lower plots represents the density for the second sample. The predicted classes for the first sample is [1, 1] and second sample is [0, 0].

Task 5 - Predicting Percentiles of Target Metrics

The objective of this task is to analytically compute and verify the different percentile values of the target metrics. We estimate the 20th, 50th, and 95th percentiles for the train dataset from the histograms derived above and verify the values using [6] on the test data. To compute the estimate of the percentile, the density of individual bins are summed until the required percentile is reached. The density of a bin is defined as the ratio of number of samples in that bin to the total number of sample in all bins.

$$estimate = \sum_{t=1}^{20} \frac{x^{(t)}}{total} \leq a_{perc} \tag{3}$$

where, $x^{(t)}$ is the number of samples in each bin, $total$ is the total number of samples, and $a_{perc} = \{0.2, 0.5, 0.95\}$

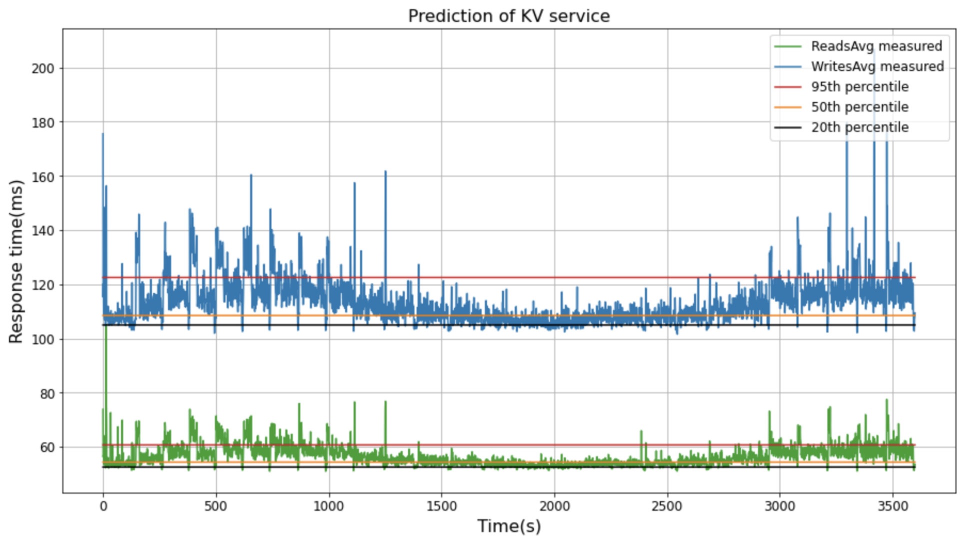

Figure 11: 20th, 50th, 95th percentiles of the target metric for the first 3600 samples

	20th percentile	50th percentile	95th percentile
ReadsAvg	52.503ms	54.319ms	60.674ms
WritesAvg	104.949ms	108.458ms	122.493ms

Table 5: Estimate of various percentile values for target metrics Y

Figure 11 and Table 5 show the 20th, 50th and 95th percentile values of the target Y for both ReadsAvg and WritesAvg.

The Glivenko-Cantelli theorem [6] can be used to verify the estimated percentile values. Using equation 4, where $perc = 0.2, 0.5, 0.95$, $(x^{(t)}, y^{(t)}) = 1......n$ are the sample in the test set, $a_{perc}(x^{(t)})$ is the predicted percentile for the value $x^{(t)}$, and {Q} denotes the indicator function, which takes the value 1 if Q is true, and the value 0 otherwise.

$$\frac{1}{n} \sum_{t=1}^{n} \mathbb{1}\{y^{(t)} \leq a_{perc}(x^{(t)})\} \tag{4}$$

20th percentile estimation [ReadsAvg: 0.0882, WritesAvg: 0.109] 50th percentile estimation [ReadsAvg: 0.464, WritesAvg: 0.39] 95th percentile estimation [ReadsAvg: 0.942, WritesAvg: 0.948]

	20th percentile	50th percentile	95th percentile
ReadsAvg	0.0882	0.464	0.942
WritesAvg	0.109	0.39	0.948

Table 6: Percentile estiamte using Glivenko-Cantelli theorem

Table 6 show the percentiles computed using [6]. It can be seen that the estimated value from the histograms are close to the Glivenko-Cantelli estimated values.

Discussion

From the above sections its is evident that service level metrics can be successfully estimated using machine learning models. Even though the estimation are not 100% accurate, it is possible to reduce the error by employing outlier removal, per-processing data, aggressively tuning the model's hyper parameters, etc. The three models presented above perform similarly in terms *NMAE*. However, random forests and neural networks slightly out perform linear regression. Also, these models are very promising for metrics estimation as they offer an plethora of tuning options.

Pre-processing the dataset is an essential step for service-level metrics prediction. As shown above, pre-processing can be applied in many different ways. However, some general techniques can be derived from the above experiments. For example, applying pre-processing over the range of the metric (column-wise) works better than sample-wise. Also, standardization works better than normalization and scaling, or at least in the context of this problem. Outlier removal is also considered a pre-processing technique and improves the accuracy of the prediction model

The service metrics regression problem is transformed into a classification problem by discretizing the target space into smaller sub-classes. Even though the NMAE achieved with random forest classification in this work is higher than the other trained models, tuning the binning of target space can lead to high accuracy [7].

References

[1] F. S. Samani, H. Zhang, and R. Stadler, "Efficient learning on high-dimensional operational data," in *2019 15th International Conference on Network and Service Management (CNSM)*, pp. 1–9, 2019.

[2] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014. 40th-year commemorative issue.

[3] "Linear regression."

[4] R. Stadler, R. Pasquini, and V. Fodor, "Learning from network device statistics," *Journal of Network and Systems Management*, vol. 25, pp. 672–698, 2017.

[5] I. Ari, B. Hong, E. Miller, S. Brandt, and D. Long, "Managing flash crowds on the internet," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.*, pp. 246–249, 2003.

[6] M. Talagrand, "The glivenko-cantelli problem," *The Annals of Probability*, pp. 837–870, 1987.

[7] F. Samani, R. Stadler, C. Flinta, and A. Johnsson, "Conditional density estimation of service metrics for networked services," *IEEE Transactions on Network and Service Management*, vol. 18, pp. 2350–2364, 06 2021.