EP2420 *Project 1: task II*

Yiyi Miao

November 7, 2025

Project Overview

In this project, we train regression models that map IT and network infrastructure measurements X to predictions of service-level metrics Y. The measurements are obtained from a Video-on-Demand (VoD) service and a Key-Value store (KV) service, and the service-level metrics are Video Frame Rate and Response Time experienced by clients.

Using machine-learning techniques, the problem is to find a function (i.e., train a model) $M : X \to \hat{Y}$, such that $\hat{Y}$ closely approximates Y for a given X. Formally, we predict the expectation of $P(Y|X)$. If Y is numeric, the problem is referred to as a regression problem; if Y is a class or category, the problem is called a classification problem. This project considers both of these problems.

The machine-learning methods you will use are linear regression, random forest regression and classification, and neural network regression.

To estimate M, measurement pairs (or observations) of the form (x, y) are needed. A list of such measurement pairs ordered by time stamps is called a *trace*. The traces in this project are based on observations collected once per second during several hours from a running system.

Task II

Dataset Preparation

Let $X' \in \mathbb{R}^{N\times 18}$ denote the feature matrix selected in Task I, and $Y \in \mathbb{R}^N$ the target variable. The target for the KV dataset is ReadsAvg, and for the VoD dataset, it is DispFrames.

The dataset was randomly split into training and testing subsets as follows:

$$X', Y \xrightarrow{\text{split}} \begin{cases} (X_{\text{train}}, Y_{\text{train}}) & 70\% \\ (X_{\text{test}}, Y_{\text{test}}) & 30\% \end{cases}$$

To ensure reproducibility, a fixed random seed was used during the split.

Evaluation Metric

Model performance is quantified using the Normalized Mean Absolute Error (NMAE), defined as

$$\text{NMAE} = \frac{1}{\bar{y}} \cdot \left(\frac{1}{n} \sum_{i=1}^{n} \left| y_i - \hat{y}_i \right| \right), \tag{1}$$

where $\bar{y}$ is the mean of the test targets, y_i are the measured values, and $\hat{y}_i$ are the predicted values. The computational overhead is measured as the model's training time in seconds.

1 Models and Training Setup

Baseline

The baseline predictor outputs the mean of the training targets:

$$\hat{Y}_i = \bar{Y}_{\text{train}} \quad \forall i.$$

Linear Regression

A standard least-squares linear regression model was trained using the closed-form solution:

$$\hat{\boldsymbol{\beta}} = (X_{\text{train}}^\top X_{\text{train}})^{-1} X_{\text{train}}^\top Y_{\text{train}}.$$

Random Forest Regressor

A Random Forest ensemble model was trained with $n_{\text{estimators}} = 100$ decision trees, min_samples_split=2 and default hyperparameters. The model captures nonlinear relationships by averaging the predictions of multiple trees.

Neural Network Regressor

A feedforward multilayer perceptron (MLP) was implemented using Keras. The network consists of two to three hidden layers with ReLU activations, optimized using the Adam optimizer and early stopping based on validation loss. A small grid search over hidden units, learning rate, and dropout rate was performed to identify the best configuration.

2 Results and Discussion

2.1 Model Accuracy and Training Time

Table 1 summarizes the performance of the trained models on the KV dataset. The Random Forest (M_2) model achieved the lowest NMAE (0.019), indicating the highest accuracy. The Linear Regression (M_1) model was the fastest to train, while the Neural Network (M_3) was by far the most computationally expensive due to the hyperparameter search. All three machine learning models significantly outperformed the Naive Baseline which simply outputs the mean value of the test set.

Table 1: Performance comparison of models on the KV test set.

Model	**NMAE**	**Training Time (s)**
Linear Regression	0.021	0.015
Random Forest	0.019	1.142
Neural Network (best)	0.023	335.278
Naive Baseline (mean)	0.041	N/A

Table 2 summarizes the performance on the VoD dataset. Similar to the KV results, the Random Forest (M_2) model provided the best accuracy with an NMAE of 0.062. The VoD dataset proved more complex to model, as indicated by the higher NMAE values across all models compared to the KV dataset. Again, M_1 was the fastest, and M_3 was the slowest and all models have a clear improvement over the Naive Baseline.

Table 2: Performance comparison of models on the VoD test set.

Model	**NMAE ↓**	**Training Time (s) ↓**
Linear Regression	0.119	0.009
Random Forest	0.062	2.062
Neural Network (best)	0.115	627.180
Baseline (mean)	0.174	N/A

2.2 Neural Network Hyperparameter Search

A grid search was performed to find effective hyperparameters for the M_3 (Neural Network) models for both datasets. The search process was performed iteratively in two stages.

In the first stage, I started with a relatively broad search range for each parameter (e.g., hidden layer sizes of 32, 64, 128, learning rates of 0.001, 0.0005, and 0.0001, dropout rates of 0.1–0.3). This wide exploration allowed me to identify the general region where the validation loss decreased most significantly.

In the second stage, I refined the grid around the promising parameter combinations discovered in the first stage. By slightly adjusting one parameter at a time and narrowing the search intervals, I could "zoom in" on the area with the lowest validation loss. This progressive search strategy helps to balance computational efficiency with optimization accuracy.

To prevent overfitting, I applied early stopping based on validation loss and restored the best weights from training. The final model achieved stable convergence and offered the best trade-off between training time and accuracy.

KV Model (M3) The KV model was constructed with two hidden layers. The best performance (NMAE: 0.023) was achieved with the following hyperparameters:

- neurons_layer1: 64
- neurons_layer2: 4
- learning_rate: 0.008
- dropout_rate: 0.1

VoD Model (M3) The VoD model used a deeper architecture with three hidden layers. The best performance (NMAE: 0.115) was achieved with these parameters:

- neurons_layer1: 64
- neurons_layer2: 64
- neurons_layer3: 32
- learning_rate: 0.0007
- dropout_rate: 0.2

2.3 Time Series Comparison

For the time series analysis (Step 5), the M_2 (Random Forest) model was selected as it provided the best NMAE for both datasets. Figure 1 and Figure 2 for both KV and VoD show the measured versus predicted target values for 1000 test samples, sorted by timestamp. In both figures, the 'Estimation (M_2)' line closely tracking the 'Measured' line. It successfully captured the dynamic peaks and troughs of the service metrics. In contrast, the 'Naive estimation' was a flat line representing the training set mean, which failed to capture any of the system's temporal behavior.

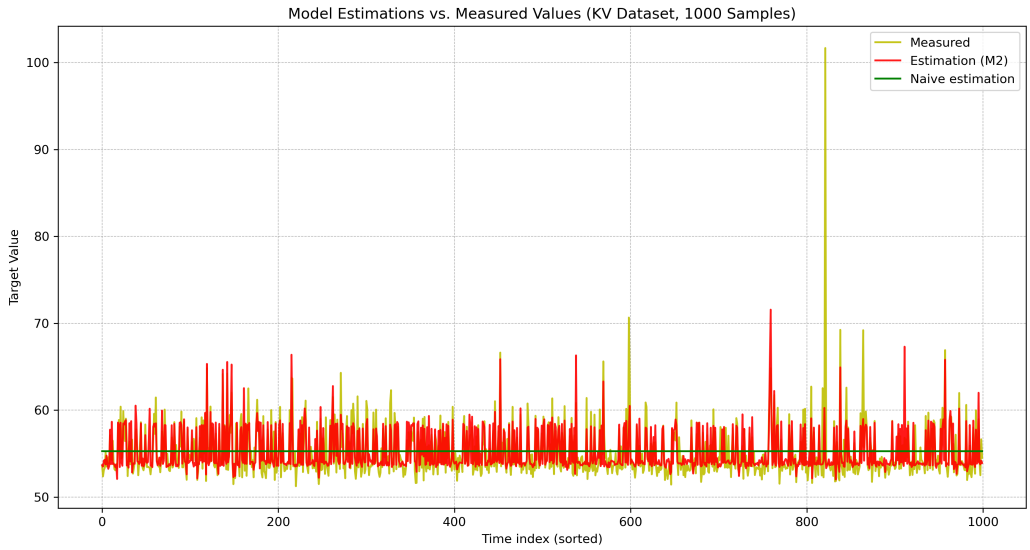

Figure 1: Measured vs. predicted targets for 1000 test samples.

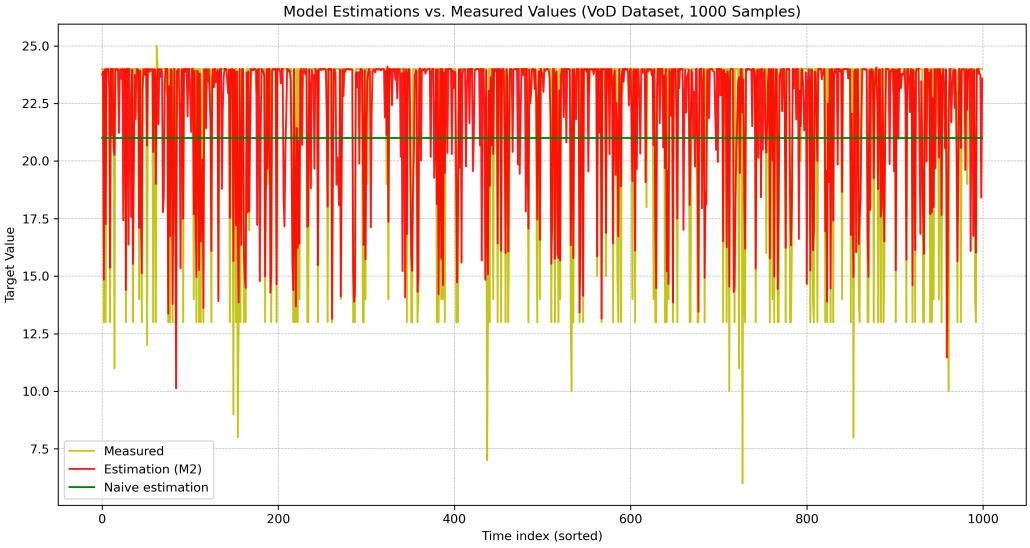

Figure 2: Measured vs. predicted targets for 1000 test samples.

2.4 Target Value Distribution

The histogram and kernel density estimate (KDE) of the test targets are plotted in and Figure 3 Figure 4, revealing distinct distributions for the two services. For the VOD dataset, bin size is 1 frame; for the KV dataset, 1 ms.

- **KV (ReadsAvg):** The distribution was observed to be unimodal and sharply peaked around its

mean, indicating relatively low variance in the response time.

- **VoD (DispFrames):** The distribution was observed to be multimodal (primarily bimodal), with significant clusters of values. This suggests that the system operates in different states, leading to different typical frame rates.

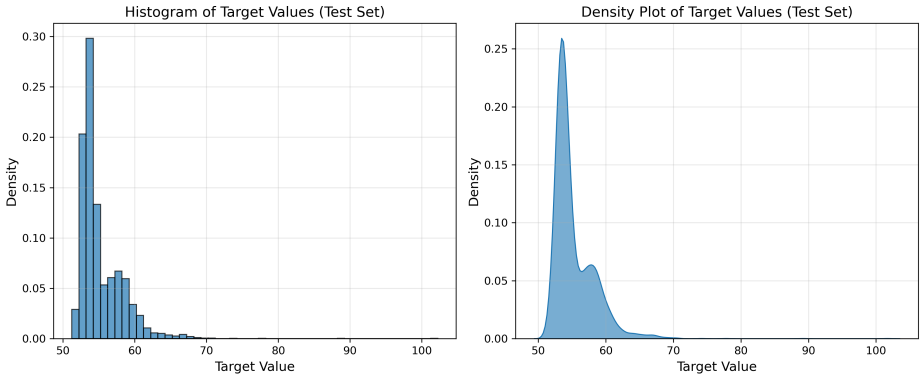

Figure 3: Residual density of the three regression models.

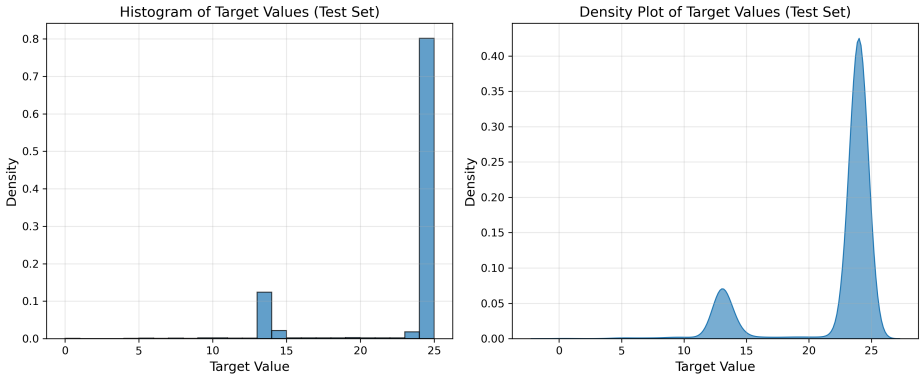

Figure 4: Residual density of the three regression models.

2.5 Error Density

The density plots of the estimation errors $(y_j - \hat{y}_j)$ for all three models (Step 7) provided a clear comparison of model precision, shown in Figure 5 and Figure 6.

- For both KV and VoD datasets, the M_2 (**Random Forest**) model's error distribution was the narrowest and most sharply peaked around zero. This indicates the highest precision and lowest bias among the three models.
- The M_1 (**Linear Regression**) and M_3 (**Neural Network**) models showed wider and flatter error distributions, signifying larger and more varied errors in their predictions.

The error statistics for the KV dataset were:

- **M1 Error:** Mean = 0.017, Std = 2.443, Min = −82.680, Max = 47.028

- **M2 Error:** Mean = -0.039, Std = 1.804, Min = -14.852, Max = 47.346
- **M3 Error:** Mean = 0.340, Std = 2.076, Min = -6.538, Max = 48.502

The error statistics for the VoD dataset were:

- **M1 Error:** Mean = 0.097, Std = 3.587, Min = -21.069, Max = 7.952
- **M2 Error:** Mean = 0.100, Std = 2.654, Min = -17.409, Max = 11.403
- **M3 Error:** Mean = -0.088, Std = 4.720, Min = -25.868, Max = 9.367

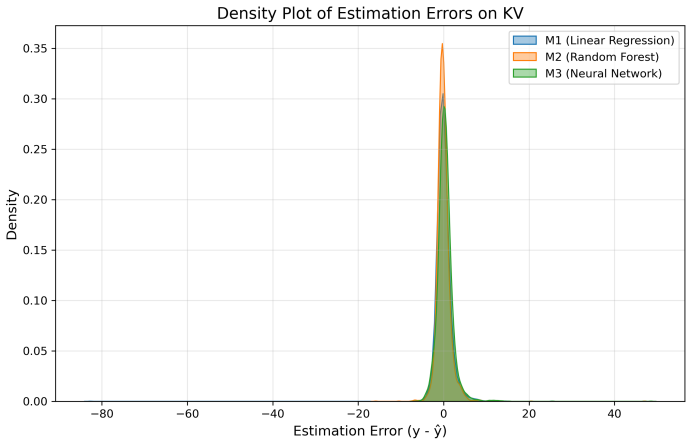

Figure 5: Residual density of the three regression models.

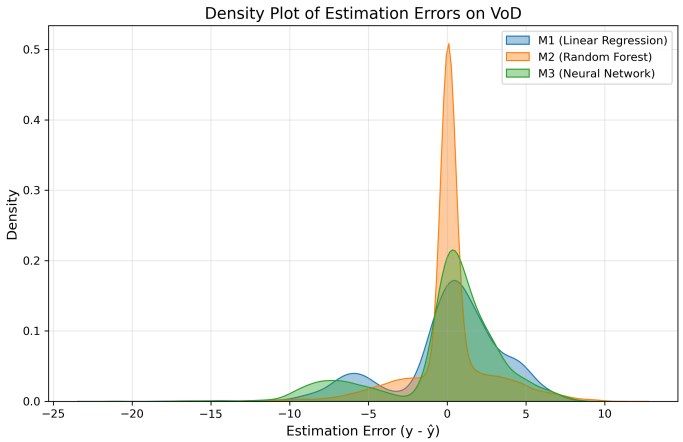

Figure 6: Residual density of the three regression models.

Discussion

Based on the results from Task II, we can draw the following conclusions:

Accuracy: The M2 (Random Forest Regression) model was the most accurate, achieving the lowest NMAE for both the simple KV dataset (0.019) and the more complex VoD dataset (0.064). All machine learning models significantly outperformed the Naïve Baseline, demonstrating their ability to learn the relationship between infrastructure measurements and service metrics.

Computational Overhead: The M1 (Linear Regression) model was by far the fastest to train (0.015s and 0.009s). The M2 (Random Forest) model was also very efficient, with training times of just 1.024s and 2.006s. The M3 (Neural Network) model was extremely computationally expensive (335s and 627s), almost entirely due to the extensive hyperparameter search required to find a viable architecture.

Overall: For both services, the Random Forest (M2) model provides the best trade-off, delivering the highest accuracy with a very low computational overhead. The Linear Regression (M1) model, while fastest, is not as accurate, especially for the complex VoD data. The Neural Network (M3) did not outperform the Random Forest in this instance and required significantly more time and effort to tune. This suggests that tree-based ensembles are highly effective for this type of tabular, high-dimensional operational data.