EP2420 Project 2 - Final Report

Lisa Bjelm, Léo Lebuhotel

October 9, 2024

1 Project Overview

This report discusses the application of machine learning techniques for forecasting service-level metrics of Video-on-Demand (VoD) and Key-Value (KV) store services. The data were collected from a KTH testbed and the FedCSIS 2020 challenge dataset.

The objective of the project is to apply different regression methods (Linear Regression, Recurrent Neural Networks) and to use time series analysis to make predictions of the future based on past and present data.

In Task I, we predict future values based on historical data using linear regression techniques. The analysis encompasses data pre-processing, feature selection using tree-based methods, and model evaluation to determine the most effective forecasting approach.

In Task II, we will use Recurrent Neural Networks for forecasting. Similarly, as in Task I, we pre-processed the data, removed the outliers, and created different datasets on which we trained different non-linear LSTM models, one for each lag and horizon value.

In Task III, we proceed to do time series analysis. The objective is to apply traditional uni-variate time-series analysis methods, by considering only the target values $y^{(t)}$ of the trace and not the input values anymore. For that, we test the stationarity of the time series by using the Augmented Dicky-Fuller Test and the Auto Correlation function.

In Task IV, we focus on the KTH dataset and perform model fitting and forecasting using the following time series models :

- Auto Regression (AR)

- Moving Average (MA)

- Auto Regressive Integrated Moving Average (ARIMA)

- Exponential Smoothing method

We evaluate those four methods to determine the most effective forecasting approach.

2 Background

2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are artificial neural networks for handling time series. Unlike traditional feed-forward neural networks, RNNs are bi-directional and have connections that loop back on themselves, allowing them to maintain a memory of previous inputs. In an RNN, each unit, or neuron, receives the current input $x^{(t)}$ and information from the previous step in the sequence. This feedback loop enables RNNs to capture dependencies and patterns in sequential data. In this way, RNNs are well-suited for time series analysis.

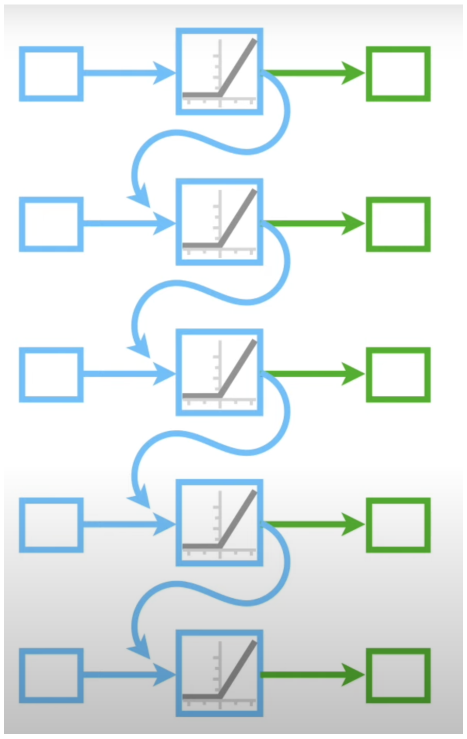

Figure 1: Structure of a Recurrent Neural Network

In Figure 1, we have 5 layers. The number of layers corresponds to $l + h + 1$ with $l = 3$ the lag value and $h = 1$ the horizon value. The layer at the bottom takes for input the data $x^{(t)}$ but also the output from the previous layer $x^{(t-1)}$ and predicts $x^{(t+1)}$. If we continue until the first layer, we can understand that its input is only $x^{(t-l)}$.

The term "layer" is not appropriate. In Figure 1, this is a representation of one unrolled layer of the Recurrent Neural Network. Thus, the neurons all share the parameters (weight and bias). The input layer has l neurons, and the output layer has h neurons.

Another distinguishing characteristic of RNNs is that they share parameters across each network layer. While feed-forward networks have different weights across each node, RNNs share the same weight parameter within each network layer. These weights are still adjusted through back-propagation and gradient descent to facilitate reinforcement learning.

Let's call w the shared weight of each neuron from the previous schematic, and set the bias b to 0. The final output of this layer will be proportional to $w^4x^{(t-3)}$. If w is less than 1, the output value will tend towards 0, and we face the vanishing gradient problem. Conversely, if w is bigger than 1, the output value will tend towards infinity, and we face the exploding gradient problem.

RNNs leverage the back-propagation through time (BPTT) algorithm to determine the gradients. BPTT differs from the traditional approach because it sums the errors from each layer at each time step. After all, they share the parameters at each layer. Through this process, RNNs tend to run into two problems; exploding gradients and vanishing gradients, where the gradients diminish exponentially over time during training, making it difficult for the network to learn long-range dependencies.

A Long Short-Term Memory network is a specific version of a Recurrent Neural Network designed to avoid the vanishing gradient problem, which occurs when gradients diminish exponentially during back-propagation, hindering the training of networks on long sequences.

LSTMs solve this problem using a unique additive gradient structure that includes direct access to the forget gate's activations. This structure allows the network to encourage desired behavior from the error gradient using frequent gate updates at every time step of the learning process.

Furthermore, LSTMs include the Constant Error Carousel (CEC), which ensures the error can flow through many time steps without vanishing or exploding. This is achieved by introducing gates controlling the information flow, allowing the LSTM to keep or discard information based on relevance.

3 Data description

4 Task I

4.1 Methodology

4.1.1 Data pre-processing

The dataset was first standardized, and outliers were removed to ensure data quality.

Standardizing the data corresponds to applying the following operation to each column: $x_{scaled} = \frac{(x-\mu)}{\sigma}$ with μ the sample mean for each feature, and σ the standard deviation for each feature. In that way, we have a mean of 0 and a standard deviation of 1 in each column, and we can use that standardization to remove the outliers.

The outliers are the samples with a feature value above a certain threshold T. Standardization is useful here because all the features must be on the same scale for this threshold to have meaning. We plotted the percentage of outliers in our dataset against the value of threshold.

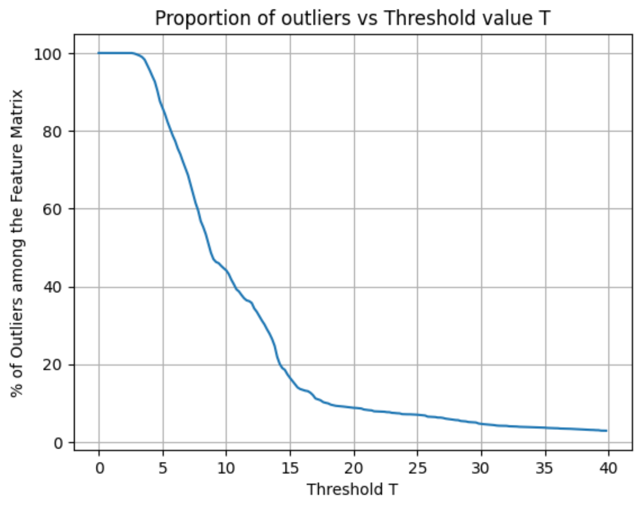

Figure 2: Proportion of outliers in the dataset against the threshold value T

From 2, we can see that we should use a threshold of 28 to keep 95% of the data and remove the remaining outliers. It is good to note that this value seems high. Since we standardized the data, a threshold of 3 should be enough to remove 5% of the outliers (for Gaussian-distributed data).

This suggests that the data is not normally distributed. Since we consider a large number of features, the likelihood that one of them is not normally distributed is high. Our outlier removal methods considers a point as ”outlier” as soon as one of its feature values is beyond the threshold. The features have different underlying distributions. If one of them is not normally distributed, the threshold value has no reason to be equal to 3.

At the end, the value of the threshold is not important, what matters is that we effectively removed 5% of the outliers.

The final step in the pre-processing of the data is to select 16 features, based on their importance, using a RandomForestRegressor. We train a Random Forest with the whole dataset, with 50 estimators, and we keep the 16 features that contribute to are the most informative in the building of the tree (by the information gain).

In that way, we simplified the data a lot and we can move forward an train a linear model to perform forecasting.

4.1.2 Model Training

The training of linear regression models was a critical phase of this project, focusing on forecasting with varying time lags and horizons. This subsection elaborates on the detailed steps involved in this process.

Data Structuring for Time Series Analysis Initially, the dataset was structured to facilitate time series analysis. In this context, ’lag’ refers to the number of time steps into the past used to predict future values, while ’horizon’ denotes the number of future time steps we aim to predict. Specifically, for each time point t, the model inputs consist of a sequence of features from $x(t-l)$ to $x(t)$, where l is the lag. Correspondingly, the outputs are the values from $y(t)$ to $y(t+h)$, with h being the horizon. This structuring converts the problem into a supervised learning task, allowing the linear regression model to learn the relationship between past (lagged) and future (horizon) data points.

Lag and Horizon Selection We trained multiple models with varying lags and horizons to understand their impact on forecasting accuracy. The lags and horizons were varied from 0 to 10, creating a matrix of model configurations. A lag of 0 implies using only the current time point for prediction, while a larger lag indicates the inclusion of more past data. Similarly, different horizons represent the extent of the future we are trying to predict. This comprehensive approach allowed us to examine the models’ performance across a spectrum of immediate to more distant forecasts.

Linear Regression Model Training A linear regression model was trained for each lag and horizon. Linear regression was chosen for its simplicity and interpretability, essential in understanding the underlying patterns in time series data. The model aims to find a linear relationship between the input features (past data points) and the target variable (future values).

The training involved fitting the model to the training data, which includes finding the optimal coefficients (weights) for the input features that minimize the prediction error. We used the Normalized Mean Absolute Error (NMAE) as our evaluation metric, which provides a normalized measure of the average magnitude of the errors in a set of predictions without considering their direction.

4.2 Results

4.2.1 Model Evaluation

The linear regression models were evaluated using their Normalized Mean Absolute Error (NMAE) for varying lags and horizons. The NMAE values are summarized in the table below:

l/h	0	1	2	3	4	5	6	7	8	9	10
0	0.1188	0.1191	0.1191	0.1192	0.1192	0.1193	0.1194	0.1194	0.1194	0.1195	0.1195
1	0.1186	0.1186	0.1187	0.1187	0.1187	0.1188	0.1189	0.1189	0.1190	0.1191	0.1192
2	0.1178	0.1179	0.1180	0.1181	0.1182	0.1183	0.1183	0.1184	0.1185	0.1186	0.1186
3	0.1172	0.1173	0.1174	0.1175	0.1175	0.1176	0.1177	0.1177	0.1178	0.1179	0.1180
4	0.1166	0.1167	0.1167	0.1168	0.1169	0.1170	0.1170	0.1171	0.1172	0.1172	0.1173
5	0.1164	0.1165	0.1166	0.1167	0.1167	0.1168	0.1169	0.1169	0.1170	0.1171	0.1171
6	0.1164	0.1164	0.1165	0.1165	0.1166	0.1167	0.1167	0.1168	0.1169	0.1169	0.1170
7	0.1162	0.1163	0.1164	0.1164	0.1165	0.1166	0.1166	0.1167	0.1168	0.1168	0.1169
8	0.1163	0.1163	0.1164	0.1165	0.1165	0.1166	0.1167	0.1167	0.1168	0.1168	0.1169
9	0.1162	0.1163	0.1163	0.1164	0.1165	0.1165	0.1166	0.1166	0.1167	0.1167	0.1168
10	0.1163	0.1163	0.1164	0.1165	0.1165	0.1166	0.1166	0.1167	0.1168	0.1168	0.1169

Table 1: NMAE Values for Linear Regression Models at Different Lags (l) and Horizons (h)

4.2.2 Feature Importance Analysis

For the model with $l = 10$, the feature importance analysis revealed different sets of significant features for horizons $h = 1$ and $h = 10$. The top 16 features for these horizons are listed below:

- For $h = 1$: [172, 137, 108, 79, 156, 140, 153, 76, 169, 95, 127, 111, 124, 143, 175, 159]
- For $h = 10$: [137, 153, 63, 31, 79, 12, 127, 169, 47, 124, 174, 143, 159, 156, 172, 175]

For $l = 10$, our feature matrix X has 176 features : 16 features from the first feature selection times 11 elements. It is interesting to see which of the first 16 features are mostly used. We get the rest of the division of the two previous features vectors by 16.

We obtain :

- For $h = 1$: [12, 9, 12, 15, 12, 12, 9, 12, 9, 15, 15, 15, 12, 15, 15, 15]
- For $h = 10$: [9, 9, 15, 15, 15, 12, 15, 9, 15, 12, 14, 15, 15, 12, 15, 15].

That corresponds to features '4_kbdirty', '4_ldavg.1', 2_all_.idle' for $h = 1$ and the same for $h = 10$ with also '2_kbinact'. That means that for both horizons, the features used are the same.

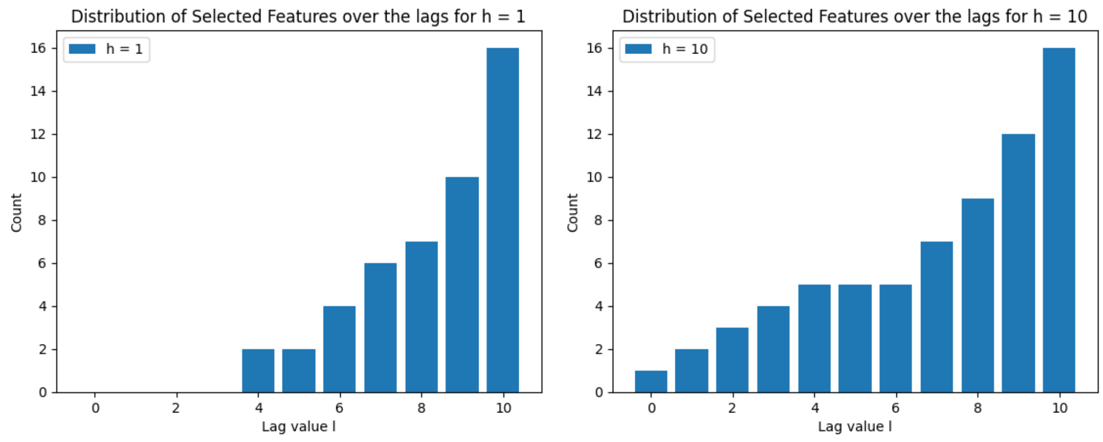

Figure 3: Distribution of the selected features over the lags 0 to 10 for $h = 1$ (left) and for $h = 10$ (right).

5 Task II

5.1 Methodology

The methodology involved pre-processing the datasets through normalization, standardization, and outlier removal. Feature selection was conducted using a tree-based method, reducing the features to 16. The primary forecasting models used were Linear Regression and Neural Network Regression, emphasizing Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks.

5.1.1 Model Training

The data was pre-processed using standardization and outlier removal techniques, like in Task I.

Feature selection was performed using a tree-based method, narrowing down to the top 16 features. We then created the training and test sets. Their structure depends on the value of the lag l and the horizon h. The LSTM model's dataset was structured to accommodate different lag and horizon values.

To build different LSTM models, we used the Sequential class from the Keras library. It was used for a layer-by-layer configuration of the LSTM model. This architectural clarity was crucial in implementing the LSTM model. We could easily modify the number of units, the activation functions, the learning rate of the optimizer, etc.

The structure of the LSTM model is the following : We used a Sequential Model with 50 units, each with ReLU as an activation function. Since we want to do regression, the output layer has a Linear activation function. We used Adam Optimizer with a learning rate $l = 0.001$ and two different loss functions: Mean Squared Error (MSE) and Normalized Mean Absolute Error (NMAE).

The shape of the data was a significant factor. We had to reshape the data X_{train} with a shape of $(n_{samples}, l + 1, n_{features})$ with l the lag value.

In our quest to forecast service-level metrics for Video-on-Demand (VoD) and Key-Value (KV) store services, we embarked on a methodology encompassing a series of nuanced steps. This journey involved intricately weaving through data pre-processing, keenly focusing on normalization, standardization, and outlier removal, and subsequently delving into the realms of machine learning with a specific emphasis on Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks.

5.1.2 Evaluation Metrics

To measure the effectiveness of our models, we employed two key metrics in the training phase: Mean Squared Error (MSE) and Normalized Mean Absolute Error (NMAE). MSE provided a measure of the average of the squares of the errors, essentially capturing the variance in prediction errors. NMAE, on the other hand, offered a normalized version of the mean absolute error, providing a relative measure of error normalized against the mean of the target values.

These metrics were instrumental in assessing the models under different configurations, shedding light on their predictive capabilities and ability to generalize across various scenarios.

In conclusion, blending thorough data pre-processing with advanced LSTM modeling and comprehensive evaluation, this methodological approach was meticulously designed to develop a robust forecasting system. This system is not just a theoretical construct but is geared to effectively handle the complexities and unpredictability of real-world time series data in VoD and KV store services.

5.2 Results

5.2.1 Model Evaluation

The LSTM models were trained using MSE on the one hand and NMAE on the other hand, and they were evaluated using Normalized Mean Absolute Error

(NMAE) for varying lags and horizons. The results were tabulated, providing a comprehensive view of the model's performance across different configurations.

Mean Squared Error as loss The MSE values in Table 2 across different lags and horizons indicate that the models perform better with higher lag values. This means that using more data from the past efficiently predicts the future. However, the increase in error at higher horizons implies a challenge in long-term forecasting. This trend suggests that the models are more effective at capturing short-term dependencies in the data.

l/h	0	1	2	3	4	5	6	7	8	9	10
0	0.116	0.113	0.117	0.132	0.122	0.120	0.115	0.126	0.150	0.126	0.142
1	0.115	0.113	0.116	0.115	0.117	0.127	0.137	0.137	0.127	0.147	0.125
2	0.116	0.116	0.118	0.117	0.121	0.119	0.133	0.133	0.127	0.143	0.134
3	0.115	0.113	0.121	0.116	0.118	0.138	0.135	0.141	0.122	0.135	0.126
4	0.117	0.115	0.118	0.122	0.118	0.124	0.126	0.126	0.133	0.134	0.136
5	0.115	0.117	0.121	0.124	0.122	0.122	0.127	0.132	0.140	0.119	0.131
6	0.115	0.114	0.120	0.123	0.117	0.123	0.124	0.119	0.125	0.121	0.128
7	0.114	0.116	0.117	0.120	0.122	0.122	0.118	0.127	0.125	0.125	0.128
8	0.115	0.114	0.113	0.122	0.120	0.123	0.128	0.123	0.127	0.131	0.129
9	0.116	0.114	0.115	0.123	0.122	0.125	0.123	0.127	0.122	0.131	0.129
10	0.114	0.115	0.121	0.123	0.120	0.117	0.122	0.124	0.123	0.133	0.122

Table 2: NMAE Values for LSTM Models at Different Lags (l) and Horizons (h) using the Mean Squared Error as loss during training (without validation split)

Including a validation set in Table 3 generally resulted in higher MSE values, indicating a more conservative approach in training. The models here might be less over-fitted to the training data, potentially offering better generalization. However, the increased error rates suggest a trade-off between generalization and immediate performance on the training set. We used 20% of the training set as the validation set.

l/h	0	1	2	3	4	5	6	7	8	9	10
0	0.185	0.126	0.128	0.121	0.147	0.143	0.122	0.137	0.146	0.123	0.122
1	0.182	0.142	0.139	0.117	0.116	0.138	0.133	0.140	0.130	0.145	0.139
2	0.182	0.126	0.127	0.123	0.119	0.123	0.121	0.133	0.123	0.122	0.138
3	0.179	0.126	0.129	0.119	0.122	0.123	0.135	0.126	0.119	0.113	0.123
4	0.173	0.124	0.120	0.126	0.123	0.123	0.135	0.124	0.116	0.118	0.138
5	0.152	0.123	0.115	0.126	0.127	0.121	0.126	0.133	0.141	0.140	0.119
6	0.158	0.125	0.118	0.118	0.123	0.119	0.127	0.130	0.122	0.132	0.148
7	0.157	0.126	0.122	0.123	0.118	0.123	0.117	0.121	0.136	0.130	0.134
8	0.157	0.122	0.116	0.118	0.123	0.121	0.119	0.127	0.123	0.122	0.128
9	0.171	0.122	0.121	0.127	0.128	0.126	0.127	0.122	0.116	0.129	0.113
10	0.170	0.129	0.125	0.123	0.127	0.125	0.122	0.120	0.120	0.118	0.142

Table 3: NMAE Values for LSTM Models at Different Lags (l) and Horizons (h) using the Mean Squared Error as loss during training (with validation split of 20%)

Normalized Mean Absolute Error as loss Like MSE, the NMAE values shown in Table 4 are lower for models trained without a validation set, particularly at lower lags. This finding aligns with the expectation that these models may be overfitting to the training data, leading to lower error rates on the training set but potentially poorer performance on unseen data.

l/h	0	1	2	3	4	5	6	7	8	9	10
0	0.101	0.103	0.104	0.107	0.109	0.099	0.098	0.110	0.108	0.111	0.114
1	0.102	0.106	0.107	0.110	0.105	0.108	0.102	0.115	0.109	0.114	0.112
2	0.101	0.101	0.107	0.108	0.104	0.104	0.117	0.111	0.119	0.114	0.105
3	0.100	0.103	0.106	0.101	0.106	0.107	0.106	0.113	0.114	0.114	0.106
4	0.100	0.105	0.102	0.108	0.105	0.112	0.104	0.106	0.109	0.116	0.114
5	0.100	0.101	0.107	0.103	0.114	0.105	0.111	0.114	0.125	0.113	0.113
6	0.099	0.099	0.109	0.107	0.113	0.113	0.110	0.112	0.112	0.111	0.114
7	0.100	0.102	0.108	0.108	0.114	0.109	0.110	0.113	0.111	0.108	0.111
8	0.100	0.103	0.104	0.107	0.107	0.115	0.105	0.117	0.113	0.113	0.109
9	0.099	0.100	0.109	0.104	0.110	0.109	0.107	0.113	0.111	0.115	0.112
10	0.099	0.101	0.106	0.107	0.107	0.111	0.114	0.109	0.110	0.109	0.110

Table 4: NMAE Values for LSTM Models at Different Lags (l) and Horizons (h) using the NMAE as loss during training (without validation split)

The NMAE values in Table 5 are higher when models are trained with a validation set. This suggests that while potentially less accurate on the training set, these models might be better at handling unseen data. This balance between training accuracy and generalizability is crucial in practical forecasting applications.

l/h	0	1	2	3	4	5	6	7	8	9	10
0	0.160	0.108	0.110	0.109	0.111	0.116	0.106	0.109	0.117	0.111	0.102
1	0.154	0.115	0.112	0.116	0.118	0.140	0.110	0.117	0.117	0.099	0.099
2	0.142	0.111	0.110	0.108	0.113	0.111	0.114	0.102	0.118	0.102	0.096
3	0.132	0.113	0.107	0.114	0.132	0.111	0.133	0.103	0.118	0.100	0.114
4	0.150	0.112	0.109	0.102	0.100	0.112	0.109	0.119	0.109	0.100	0.101
5	0.143	0.113	0.107	0.100	0.105	0.106	0.117	0.106	0.114	0.104	0.102
6	0.123	0.111	0.109	0.102	0.104	0.111	0.110	0.102	0.100	0.112	0.103
7	0.132	0.107	0.108	0.102	0.107	0.105	0.100	0.100	0.105	0.103	0.111
8	0.141	0.107	0.106	0.112	0.107	0.108	0.110	0.104	0.101	0.107	0.105
9	0.131	0.105	0.107	0.108	0.109	0.104	0.106	0.105	0.109	0.112	0.106
10	0.127	0.106	0.107	0.107	0.103	0.104	0.100	0.104	0.107	0.107	0.107

Table 5: NMAE Values for LSTM Models at Different Lags (l) and Horizons (h) using the NMAE as loss during training (with validation split)

5.3 Discussion

Analyzing the LSTM models across various lags and horizons has unveiled some critical insights into the dynamics of time series forecasting. One of the key observations is the models' pronounced proficiency in capturing short-term dependencies, as evidenced by the lower error rates at smaller lag values. This characteristic underscores the LSTM's inherent strength in incorporating and projecting near-term trends and patterns within the data. However, the models encounter challenges as the forecasting horizon extends, reflected in the increased error rates for longer horizons. This shift illuminates a fundamental aspect of time series forecasting: the complex balance between short-term accuracy and long-term predictive capability.

Moreover, using a validation set introduces an exciting dimension to the model training process. The higher error rates observed in this setup suggest that while these models might exhibit a slightly reduced performance on the training data, they potentially offer a more robust framework for generalizing to unseen data. This finding is particularly salient in real-world applications, where the ability to generalize and adapt to new data is as critical as the model's accuracy on known data.

6 Task III

6.1 Methodology

The methodology involved pre-processing the datasets through normalization, standardization, and outlier removal, similar to what was done in Tasks I and II. In Task III, we focused solely on the target values $y^{(t)}$, disregarding the input values $x^{(t)}$. We employed a standardization method for outlier removal, setting a threshold to retain 99% of the data, ensuring a robust representation of the overall trend while eliminating extreme anomalies.

In Task III, we proceeded with a test of the stationarity of a time series using the Augmented Dicky-Fuller (ADF) test. Then, we computed the Auto-Correlation function of our time series concerning different lag values. Lastly, an analyze was performed on the results to define which time series models and parameters may perform best for the project.

6.1.1 Augmented Dicky-Fuller Test

The Augmented Dicky-Fuller test determines if a time series is stationary. This involves analyzing if the mean and auto-covariance of the series are time-independent. The test was conducted on random sequences of 50, 500, and 5000 values from the time series data, considering significance levels of 1%, 2%, and 5%.

The ADF test equation at time t is:

$$\Delta y(t) = \alpha + \beta t + \gamma y(t-1) + \sum_{i=1}^{p} \delta_i \Delta y(t-i) + \epsilon_t$$

6.1.2 Auto-Correlation Function

The Auto-Correlation Function (ACF) computes the correlation of observations in a time series concerning different lag values.

We computed this Auto-Correlation function for KTH and FedCSIS traces and plotted its values against the lag values l.

The formula for this function at lag l is given by:

$$\text{ACF}(l) = \frac{\sum_{t=1}^{n-l}(X_t - \bar{X})(X_{t+l} - \bar{X})}{\sum_{t=1}^{n}(X_t - \bar{X})^2}$$

where n is the total number of observations in the time series, X_t is the value of the time series at instant t and $\bar{X}$ is the mean of the time series values.

6.1.3 Time Series Models

- **AR Model:** An Auto Regressive model of order p is expressed as:

  $$y(t) = c + \sum_{i=1}^{p} \phi_i y(t-i) + \epsilon(t)$$

  where ϕ_i are the model coefficients, c is a constant, and $\epsilon(t)$ is white noise.

- **MA Model:** A Moving Average model of order q is given by:

  $$y(t) = \mu + \epsilon(t) + \sum_{i=1}^{q} \theta_i \epsilon(t-i)$$

  where θ_i are the model coefficients, μ is the mean, and $\epsilon(t)$ are the error terms.

- **ARIMA Model:** The Auto-Regressive Integrated Moving Average model, ARIMA (p, d, q), combines differencing with AR and MA models. It is defined as:

  $$(1 - \sum_{i=1}^{p} \phi_i L^i)(1-L)^d y(t) = (1 + \sum_{j=1}^{q} \theta_j L^j)\epsilon(t)$$

  where L is the lag operator, ϕ_i are the coefficients of the AR part, θ_j are the coefficients of the MA part, d is the order of differencing, and $\epsilon(t)$ is white noise.

6.2 Results and Analysis

The ADF tests and ACF plots for both traces revealed distinct characteristics of each time series.

KTH Trace:

- The ADF test indicated varying degrees of stationarity across different sequence lengths and significance levels. Notably, the series appeared non-stationary for a length of 500 and a 5% significance level.
- The ACF plots suggested potential models for time series forecasting, with certain lags showing significant correlation.

FedCSIS Trace:

- The ADF test results were more indicative of non-stationarity, especially at shorter sequences.
- The ACF plots for the FedCSIS trace exhibited different patterns than the KTH trace, influencing the choice of time series models.

6.2.1 Auto-Correlation Function (ACF) Plots

6.2.2 KTH Trace

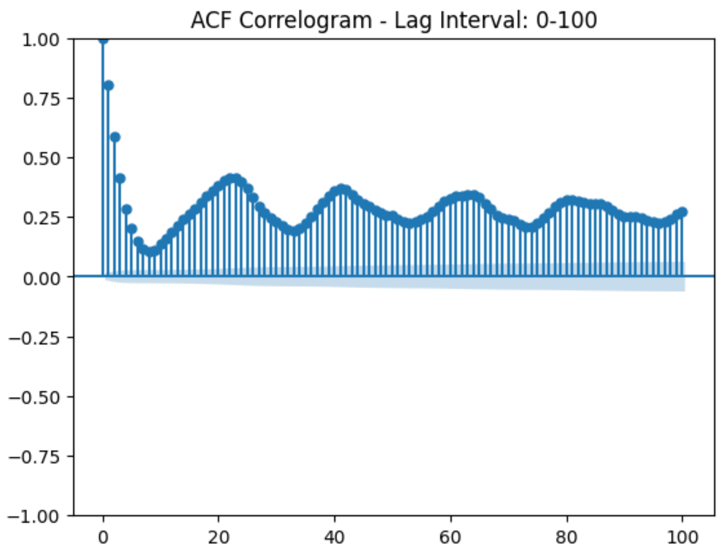

Figure 4: ACF Correlogram for KTH trace with lag interval 0-100

Figure 4 exhibits a sharp decline from lag 0 to around lag 5, followed by a series of damped oscillations that gradually taper off. This pattern suggests that the immediate past strongly influences the current value, which quickly diminishes as you go back in time. A lag size of around 5-10 might be suitable for an AR component.The oscillations indicate a possible seasonal or cyclical pattern within the first 100 lags. This could suggest the suitability of a Mixed Autoregressive Moving Average (ARMA) model, where the autoregressive part captures the decay in correlation over time, and the moving average part captures the cyclical pattern.

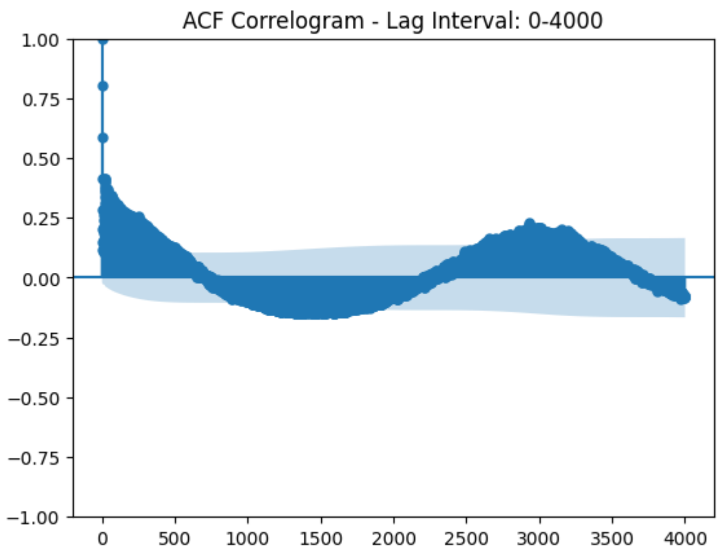

Figure 5: ACF Correlogram for KTH trace with lag interval 0-4000

The plot in Figure 5, with a much wider lag interval, shows a gradual decline with some fluctuations that persist over a longer range of lags. A slow decay in autocorrelation over thousands of lags could imply that the time series is integrated, meaning it requires differencing to make it stationary. This is a characteristic that an Autoregressive Integrated Moving Average (ARIMA) model with $d = 1$ can capture and an AR component with a lag size of 5-10 could still be relevant.

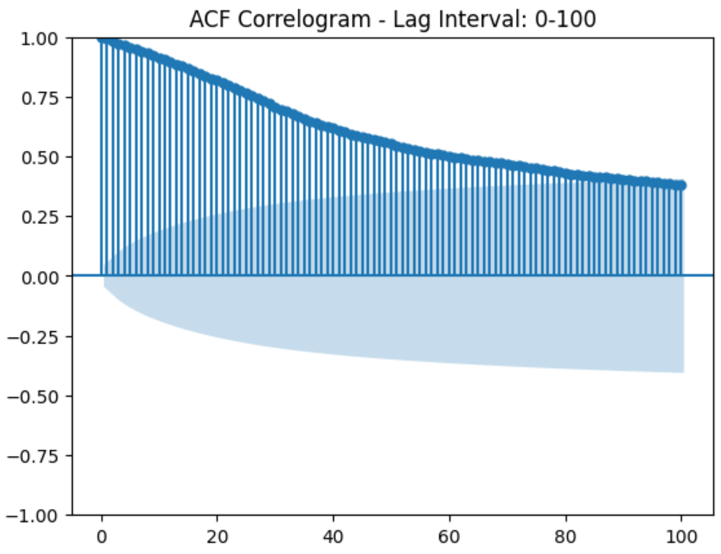

Figure 6: ACF Correlogram for FedSCIS Trace with lag interval 0-100

FedSCIS Trace Figure 6 shows a very strong initial correlation that decreases linearly as the lags increase. This suggests that recent past values strongly influence future values, with this influence waning as we move back in time. Such a pattern could indicate an Autoregressive (AR) process, where the current value is a sum of past values weighted by some factors.

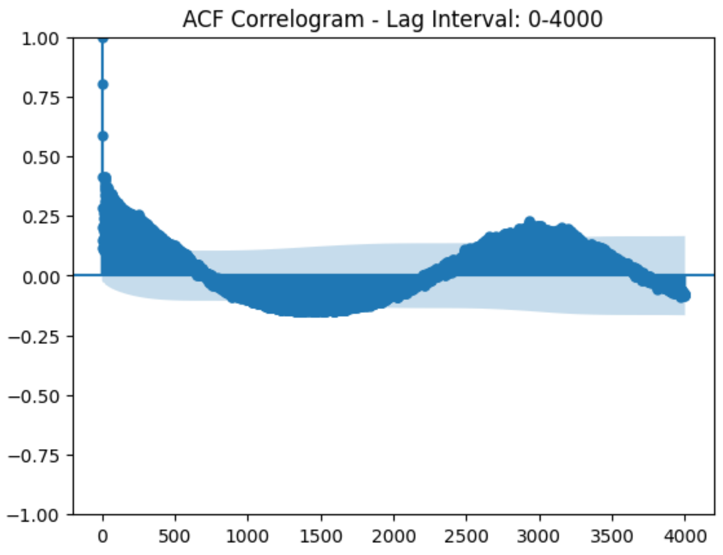

Figure 7: ACF Correlogram for FedSCIS Trace with lag interval 0-1896

For this broader lag interval in Figure 7, the ACF plot displays a fluctuating pattern that does not show a clear exponential decay or sinusoidal pattern, which would typically suggest an AR or MA process. Instead, the correlation oscillates around zero without a regular pattern, indicating that complex dynamics may be at play that are not immediately obvious. This could imply that the time series has some degree of non-stationarity or that multiple underlying processes affect the series.

6.3 From Non-Stationary to Stationary

A stationary time series is one whose properties do not depend on the time the series is observed. In contrast, a non-stationary time series is one where the statistical properties change over time.

Converting a stationary dataset to a non-stationary one can be achieved through several methods. One common approach is the introduction of a trend. This can be done by adding a deterministic trend, a consistent, predictable, directional pattern. For instance, a linear trend can be added to the data by adding a sequence of increasing numbers to the original dataset. We can also use log transformation to stabilize the variance in time series data, particularly when a positive trend is present.

Another method is to introduce seasonality into the data. Seasonality is a characteristic of a time series in which the data experiences regular and predictable changes that recur every calendar year. This can be achieved by adding a sine or cosine function to the data with a specific period. We can also differentiate the time series.

Yet another way to induce non-stationarity is by adding a unit root to the data. A unit root is a feature of some stochastic processes (such as random walks) that can cause statistical properties to change over time. This can be done by taking the cumulative sum of the original data, which will result in a random walk process.

The reason for converting a stationary dataset to a non-stationary one is often to simulate real-world scenarios. By transforming a stationary dataset into a non-stationary one, we can create a more realistic model of the world, which can help us understand the underlying dynamics of these phenomena and make accurate predictions.

7 Task IV

7.1 Methodology

7.1.1 Time Series Forecasting Models

We applied different time series models to the standardized and outlier-filtered KTH dataset. The models and their respective configurations are as follows:

- **Autoregression (AR)**: Models were evaluated for AR parameter $p = 1, \dots, 10$.
- **Moving Average (MA)**: Evaluated for MA parameter $q = 1, \dots, 10$.
- **ARIMA**: Parameters set as $d = 1$, $p = 1, \dots, 10$, and $q = 1, \dots, 5$.
- **Exponential Smoothing**: Applied with $l = 10$ and smoothing constants $\alpha = 0.2, 0.5, 0.8$.

7.2 Evaluation Metric

The Mean Absolute Error (MAE) was used as the evaluation metric to assess the performance of each model. MAE measures the average magnitude of prediction errors, offering a straightforward interpretation of forecasting accuracy.

7.3 Results

The forecasting models' performance varied across different parameters. Key observations are displayed in the below sections.

AR Models
Showed a general trend of decreasing MAE with increasing values of p, suggesting better performance with higher-order models. The MAE values are: [0.1774, 0.0843, 0.0852, 0.0850, 0.0837, 0.1668, 0.0817, 0.0797, 0.0748, 0.0712].

MA Models
They exhibited variable performance, with certain configurations yielding low MAE values. The MAE values are: [0.1616, 0.4113, 0.4113, 0.4113, 0.4113, 0.1616, 0.4113, 0.4113, 0.4113, 0.0036].

ARIMA Models
Demonstrated a complex pattern where some combinations of p and q led to lower error rates. The MAE matrix is as follows:

0.2481	0.2479	0.2478	0.2477	0.2477
0.0020	0.0048	0.0047	0.0048	0.0019
0.0022	0.0045	0.0048	0.0057	0.0009
0.0024	0.0050	0.0056	0.0037	0.0057
0.0025	0.0020	0.0047	0.0037	0.0064
0.2470	0.2444	0.2447	0.2484	0.2436
0.0030	0.0049	0.0046	0.0014	0.0054
0.0033	0.0053	0.0068	0.0032	0.0053
0.0034	0.0034	0.0052	0.0043	0.0061
0.0037	0.0036	0.0035	0.0047	0.0085

Exponential Smoothing
The performance varied slightly with different values of α, but overall, the model showed consistent results. The MAE values are: 0.2: 0.2494, 0.5: 0.2497, 0.8: 0.2497.

7.4 Discussion

In Task IV, our exploration of different time series models yielded insightful findings, emphasizing the intricate nature of forecasting. The varying MAE values across models underscore the importance of selecting the appropriate model and tuning its parameters to the specific characteristics of the dataset.

Autoregression (AR) Models demonstrated improved performance with increased order, highlighting their strength in capturing the linear dependencies in the time series. However, the choice of order must balance complexity and overfitting risks.

Moving Average (MA) Models showed inconsistent performance, suggesting that they might be more sensitive to the peculiarities of the dataset. Their variable effectiveness could also point to the complexity of the error structures in the data.

ARIMA Models, combining AR and MA components, displayed a nuanced performance pattern. Some parameter combinations led to significantly lower errors, indicating the potential of ARIMA models in capturing both short-term and long-term dependencies and integrating aspects of the time series.

Exponential Smoothing showed consistent performance across different smoothing constants, reinforcing its utility as a reliable and straightforward forecasting approach, particularly for datasets with a strong level of noise or fluctuation.

These results reveal the multifaceted nature of time series forecasting. The effectiveness of a model is contingent not only on the data's inherent properties but also on the specificities of the forecasting horizon and the underlying assumptions about the data's structure.

7.5 Conclusion

Task IV has been a journey of discovery into the complexities of time series forecasting. We have gained a deeper understanding of the dynamics in forecasting the KTH dataset through meticulous application and evaluation of various models.

This task underscored the criticality of model selection and parameter tuning in time series analysis. Each model has strengths and limitations, and their performance can vary significantly based on the dataset's characteristics. Autoregression models, with their ability to capture linear dependencies, proved effective for our dataset, but their success is closely tied to the choice of model order. Similarly, the variable performance of MA and ARIMA models highlighted the intricacies of dealing with error structures and mixed dependencies in time series data.

The consistent performance of Exponential Smoothing models reaffirms their value as a robust and straightforward choice for datasets with high variability. This insight is precious for datasets with strong noise components or lack clear patterns.

In summary, Task IV has equipped us with valuable practical skills in applying various time series forecasting methods and enriched our understanding of the theoretical underpinnings that drive their performance. These insights are indispensable for developing effective and reliable forecasting solutions in the realm of service-level metrics for VoD and KV store services. Our analysis confirms the nuanced nature of time series data in service-level forecasting.