TIME-SERIES FORECASTING OF LIVE TV CHANNEL VIEWS ON OTT PLATFORMS

MUSTAFA EFEKAN PEKEL

BOĞAZİÇİ UNIVERSITY

2025

TIME-SERIES FORECASTING OF LIVE TV CHANNEL VIEWS ON OTT PLATFORMS

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Management Information Systems

by

Mustafa Efekan Pekel

Boğaziçi University

2025

Time Series Forecasting of Live TV Channel Views on OTT Platforms

The thesis of Mustafa Efekan Pekel

has been approved by:

Assist. Prof. Aysun Bozanta Hakyemez _____
(Thesis Advisor)

Prof. Bertan Yılmaz Badur _____

Assoc. Prof. Selçuk Kıran _____
(External Member)

June 2025

DECLARATION OF ORIGINALITY

I, Mustafa Efekan Pekel, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all sources of ideas and words, including digital resources, which have been produced or published by another person or institution;
- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- this is a true copy of the thesis approved by my advisor and thesis committee at Boğaziçi University, including final revisions required by them.

Signature……...

Date…….....

ABSTRACT

Time-Series Forecasting of Live TV Channel Views on OTT Platforms

This study compares various time series models for forecasting live TV channel views on an OTT (Over-the-Top) streaming platform, based on volatile user behavior on digital platforms and increasing demand for live broadcasts. The study evaluates traditional statistical models (ARIMAX, SARIMAX, Exponential Smoothing), machine learning algorithms (Random Forest, Gradient Boosting, LightGBM, XGBoost), deep learning architectures (LSTM, GRU, RNN), and Transformer-based models such as FEDFormer, TimesNet, Autoformer. These models were tested on proprietary OTT platform data and benchmarked against publicly available datasets, including Wikipedia page views and weather data. A standardized sliding window approach was implemented to ensure consistent model training and evaluation across different input lengths and prediction horizons. Model performance was assessed using Mean Absolute Error (MAE), Mean Squared Error (MSE), and runtime. The findings reveal that Transformer-based models achieve the highest forecasting accuracy, particularly in dynamic and non-stationary environments, though at a significant computational cost. Tree-based models demonstrate an optimal balance between efficiency and accuracy, making them practical for real-time deployment. Traditional models remain highly effective for structured and seasonal datasets. These results can help in selecting forecasting models based on characteristics, system constraints, and application requirements, offering valuable insights for both researchers and industry professionals in the fields of time-series forecasting and OTT media services.

ÖZET

OTT Platformlarındaki Canlı TV Kanalları İzlenmelerinin Zaman Serisi Tahmini

Bu çalışmada dijital izleme platformlarındaki değişken kullanıcı davranışları ve artan kullanıcı canlı yayın talepleri baz alınarak, bir OTT (Over-the-Top) yayın platformundaki canlı televizyon kanallarının izlenmelerinin saatlik tahmini için çeşitli zaman serisi modellerini kıyaslamaktadır. Çalışma kapsamında, geleneksel istatistiksel modeller (ARIMAX, SARIMAX, Üstel Düzeltme), makine öğrenmesi modelleri (Random Forest, Gradient Boosting, LightGBM, XGBoost), derin öğrenme mimarileri (LSTM, GRU, RNN) ve Transformer temelli modeller (FEDFormer, TimesNet, Autoformer vb.) değerlendirilmiştir. Modeller, özel bir OTT platformunun veri seti üzerinden test edilmiş ve Wikipedia sayfa görüntülenmeleri ve hava durumu verileri gibi herkese açık veri setleri ile karşılaştırılmıştır. Tutarlı bir değerlendirme gerçekleştirilebilmesi için "kayan pencere" yaklaşımı benimsenmiş, farklı giriş uzunlukları ve tahmin aralıkları için model performansları Ortalama Mutlak Hata, Ortalama Kare Hata ve çalışma süresi sonuçlarına göre kıyaslanmıştır. Özellikle dinamik ve lineer olmayan verilerde Transformer tabanlı modellerin en yüksek doğruluk oranlarına ulaştığı fakat yüksek hesaplama maliyeti gerektirdiğini göstermektedir. Makine öğrenmesi modelleri verimlilik ve doğruluk açısından en dengeli modeller olup, gerçek zamanlı uygulamalar için ideal bir seçenek olmuşlardır. Geleneksel modeller ise iyi yapılandırılmış ve sezonluk veri setlerinde etkili olmaktadır. Bu sonuçlar zaman serisi tahmini ve internet medya hizmetlerinde çalışan araştırmacılar ve endüstri profesyonelleri için model karakteristikleri, sistem kısıtları ve uygulanabilirlik açısından model seçimine yardımcı olacaktır.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere appreciation to my thesis advisor, Assist. Prof. Aysun Bozanta, for her all-time support, guidance, and encouragement during the thesis process. Her persistence and belief in me were crucial to complete this work. I believe that this thesis would not exist without her motivation.

I am deeply grateful to my fiancée, İlayda Çınar whose patience, love, and support were important to me while dealing with different problems throughout all these years.

My heartfelt thanks go to my two closest friends, Serdar Gürer and Ezgi Yılmaz. I had to sacrifice our time together during this tough period. Their friendship remains invaluable, even when life gets in the way.

I would like to express my heartfelt gratitude to my family for their lifetime support for my whole journey and especially to my sister Şevval, for her lifelong companionship throughout my journey.

I also sincerely thank my company for their support and for providing access to the data that formed the foundation of this research.

This thesis is the result of many forms of support, and I am truly grateful to each and every person who understood me and stood by me throughout my academic career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX TABLES

ABBREVIATIONS

AIC	Akaike Information Criterion
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Averages
ARIMAX	ARIMA with Exogenous Inputs
ARMA	Autoregressive Moving Average
BIC	Bayesian Information Criterion
CDN	Content Delivery Network
CNN	Convolutional Neural Network
CTV	Connected TV
EMA	Exponential Moving Average
FTA	Free-to-Air
GB	Gradient Boosting
GRU	Gated Recurrent Unit
KNN	K-Nearest Neighbors
LightGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
MA	Moving Averages
MAE	Mean Absolute Error
MSE	Mean Squared Error
NOAA	National Oceanic and Atmospheric Administration
OTT	Over-the-Top
RNN	Recurrent Neural Networks
SARIMA	Seasonal Autoregressive Integrated Moving Averages

SARIMAX	SARIMA with Exogenous Inputs
SVOD	Subscription-based video-on-demand
QoE	Quality of Experience
THUML	Tsinghua University Machine Learning Group
TSLib	Time Series Library
XGBoost	Extreme Gradient Boosting

CHAPTER 1

INTRODUCTION

OTT streaming platforms have grown significantly over the last ten years, transforming cable TV and traditional broadcasting models by delivering content directly to viewers over the Internet. Since gathering large amounts of data using traditional methods is challenging, the shift from older content access methods to online platforms has resulted in an exponential increase in the volume of data generated. These streaming platforms can track user interactions including watching habits, service quality, subscription trends, and engagement metrics across all devices. According to Statista Market Insights (2024), the number of OTT platform users will reach 4.92 billion in 2029 up from 3.09 billion in 2019 worldwide. On the other hand, the number of traditional TV users will only increase by 0.49 billion in the same period. As OTT platforms expand their content libraries and user bases, the ability to accurately forecast demand for resources such as bandwidth, storage, and content production becomes increasingly crucial. According to Baym (2024), Netflix could not provide a satisfactory user experience during the Tyson vs. Paul fight because it did not anticipate a demand of more than 65 million viewers simultaneously.

Time-series forecasting, a technique that analyzes historical data to predict future values, offers a framework for optimizing resource allocation in this dynamic environment (Faloutsos et al., 2018). Time-series forecasting is an important tool that OTT platforms can utilize to get actionable insights from temporal data in order to meet user needs and demands, allocate resources effectively, and make data-driven decisions in general. Methods and algorithms may produce different outputs based

on the input and specified variables. For example, predicting the number of viewers for certain events or content to optimize server loads during peak usage times could help Netflix for further events. However, forecasting models have specific challenges due to the dynamic and non-linear nature of end-user preferences on OTT platforms. Recent developments in machine learning, deep learning, and hybrid methodologies can be useful to handle complex, non-linear, and non-stationary time-series data sets. Furthermore, as the choice of parameters like learning rate and number of layers can significantly affect predicting performance and accuracy, hyperparameter tweaking is essential to the optimization of these models. Hoque and Aljamaan (2021) highlight that optimal configurations in machine learning algorithms can lead to superior forecasting outcomes. The parameters involved can be adjusted using automated methods like grid search, random search, and Bayesian optimization to make sure the models produce better outcomes in various situations. Furthermore, Weerts et al. (2020) emphasize the potential performance loss when tuning is ignored compared to manual tuning approaches.

The rapid growth of OTT platforms has significantly impacted traditional digital broadcasting technologies such as Content Delivery Networks (CDNs) with high-volume traffic. According to Arkenberg et al. (2019), OTT video traffic has become 60% of global internet bandwidth and legacy CDN providers are facing pressure from cloud, telecom, and media companies to build their own networks. This situation has created both opportunities and challenges to optimize global-scale systems. For example, Netflix's migration from a geo-based system to latency-aware load balancing reduced the delays by 1-3% while saving millions of dollars in infrastructure costs (Behnam & Fedorov, 2023). Similarly, Twitch's global infrastructure development shows the need for smart load balancing between data

centers in order to maximize efficiency and minimize possible bottlenecks (Puri, 2022). However, these technical improvements come with some environmental trade-offs. For instance, Afzal et al. (2024) stated that the infrastructure for video streaming already generates 306 million tons of CO_2 annually. Therefore, forecasting is a key part of not only optimizing needed resources but also making sure that Quality of Experience (QoE) remains satisfactory across different types of users.

This study explores the application of traditional and modern time-series forecasting techniques and methods to increase resource and capacity management efficiency for a Turkish OTT platform. It aims to compare forecasting methodologies by evaluating their capabilities in handling the complexities of non-linear OTT platform data by integrating traditional techniques like Exponential Smoothing, machine learning algorithms (e.g., Random Forest, Gradient Boosting methods), and deep learning models (e.g., LSTM, GRU, Transformers) with the support of domain expertise. These methodologies will be applied to real-world OTT usage data, with performance evaluation based on key error metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and runtime. The study will evaluate model performances in predicting hourly viewership metrics, focusing on forecasting accuracy. This study investigates the following research questions:

RQ1: Which forecasting models including statistical, machine learning, deep learning models, or Transformer-based architectures are most effective in capturing the dynamic and non-linear nature of OTT user behavior for hourly usage prediction?

RQ2: How does hyperparameter tuning impact the performance of forecasting models in predicting OTT platform data?

RQ3: How do statistical, machine learning, and deep learning forecasting models perform across datasets with different characteristics including seasonality, trend complexity, and other structural properties?

This study presents several unique contributions to the field of time-series forecasting in the context of OTT platforms.

- It is one of the first studies to comprehensively evaluate time-series forecasting methodologies, specifically for OTT platform data. This domain has not received enough interest in academics while other areas such as energy consumption, finance, and healthcare received significant attraction.
- The study evaluates 27 different models, encompassing traditional statistical methods, machine learning algorithms, and advanced deep learning architectures, and provides an extensive comparison of their forecasting performance across three different datasets with varying characteristics.
- The study measures the importance of hyperparameter tuning in optimizing forecasting models by applying grid search to enhance model performance.

The findings offer actionable insights for OTT platforms and providers to improve resource allocation, enhance user experience, and support strategic data-driven decisions in an increasingly competitive market.

The rest of the paper is organized as follows. Section 2 reviews the relevant literature on time-series forecasting algorithms, focusing on their applications in resource management, website traffic prediction, and the OTT industry. Section 3 details the datasets, forecasting algorithms, methodological framework, evaluation

metrics, and experimental setup employed in this study. Section 4 presents the results of the numerical experiments, including insights derived from different datasets. Section 5 provides a discussion of the findings, assessing their validity and alignment with existing literature. Finally, Section 6 concludes the paper by summarizing key insights and outlining directions for future research.

CHAPTER 2

LITERATURE REVIEW

The related literature is reviewed in two main sections. First, time-series forecasting algorithms are examined, and their strengths and limitations under specific conditions are discussed. Next, existing studies in which time-series forecasting has been applied to resource management, website traffic prediction, and the OTT industry are explored.

2.1 Models for time series forecasting

Time series forecasting has grown in importance as a research and application area over time in a variety of areas, including finance, healthcare, consumption, and the climate (Chakraborty et al., 2019; Faloutsos et al., 2018; Khan & Alghulaiakh, 2020; Zeng, Chen, Zhang, et al., 2023; Zhou, Ma, Wen, et al., 2022). When it comes to system optimization, resource allocation, and strategic decision-making, accurate forecasting offers an operational advantage (Khan & Alghulaiakh; 2020; Zeng, Chen, Zhang et al. 2023; Zhang et al., 2020). The development of time-series forecasting models is summed up in this literature review, which divides them into three categories: statistical approaches, machine learning approaches, and deep learning approaches. Regarding interpretability, convenience of use, algorithmic design, and processing capacity, each category has particular advantages and disadvantages.

Despite significant evolution with increasing computational capabilities, classical statistical models are still retaining their position in time-series forecasting techniques due to their interpretability and ease of use (Chakraborty et al., 2019; Shelatkar et al., 2020). In fields with distinct, low-noise patterns, the Autoregressive

Integrated Moving Average with Exogenous Inputs (ARIMAX), its seasonal counterpart Seasonal ARIMAX (SARIMAX), and Exponential Smoothing (ES) have shown success (Alharbi & Csala, 2022; Bi et al., 2021). Moreover, despite advancements in machine learning and deep learning, traditional models often remain competitive, especially in environments with smaller datasets or when underlying patterns are strong and well-defined (Casado-Vara et al., 2021; Zhang et al., 2020). In such cases, classical methods can be more efficient and interpretable, without the computational burden of more complex algorithms (Taylor & Letham, 2017). Another notable strength of traditional models lies in the foundational statistics, which guide users through the process of model selection and validation. Methods such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) help researchers select the models while considering the complexity and avoiding overfitting (Saayman & Botha, 2015).

Because machine learning models can handle high-dimensional and high-volume data while capturing complex patterns and non-linear relations, they have grown in strength as a time series forecasting alternative (Zhang et al., 2020). These methods perform especially well in multivariate time-series forecasting applications and are particularly useful for utilizing lag features and exogenous variables. According to Masini et al. (2021), statistical methods and programmed computer models can be merged and used effectively by machine learning. For example, Random Forest introduced by Breiman (2001), is an ensemble method based on decision trees. This approach enables Random Forest to model complex interactions between variables while remaining robust to overfitting. This approach enables Random Forest to model complicated relations between variables while preserving robustness contrary to overfitting. Gradient boosting (GB), on the other hand,

iteratively constructs predictive models, reducing error at every round and improving outcomes with each additional learner (Masini et al., 2021). Because of its adaptability, it can be tailored to solve certain forecasting issues. For example, Extreme Gradient Boosting (XGBoost) uses level-wise tree growth and typically generates models that are easier to understand (Wang & Guo, 2020). On the other hand, Light Gradient Boosting Machine (LightGBM) lowers computing costs by using a leaf-wise growth technique (Kougioumtzidis et al., 2025; Zhang et al., 2020). However, compared to XGBoost, LightGBM is usually more difficult to scale.

Deep learning architectures significantly impact time-series forecasting by overcoming limitations like capturing long-range dependencies and complex patterns (Bi et al., 2021; Cho et al., 2014). Recurrent Neural Networks (RNNs) are foundational models that are designed to process sequential data by preserving a step to pass information from previous time steps (Shelatkar et al., 2020). Although RNNs are effective for short-term pattern capturing, they struggle while capturing long-term dependencies because of the vanishing gradient problem. Long Short-Term Memory (LSTM) networks were introduced by Hochreiter and Schmidhuber (1997) to preserve long-term dependencies by using gating mechanisms. Gated Recurrent Units (GRUs) offer a simplified alternative to LSTMs by combining the input and forget gates into a single update gate and merging the hidden and cell states (Cho et al., 2014). GRUs maintain competitive predictive performance to LSTMs with less computational power (Shelatkar et al., 2020).

The use of Transformer-based models is another significant turning point in time-series forecasting. According to Vaswani et al. (2017), Transformers were originally designed for natural language processing but have been adapted for time series forecasting, offering significant advancements in scalability and sequence

modeling. Transformers can capture temporal relationships without the limitation of sequential processing while leveraging automatic mechanisms. This has opened new avenues for research, particularly in real-time forecasting applications such as disaster prediction and epidemic modeling (Wang, Zheng et al., 2020). Transformers efficiently model relationships across all time steps in parallel, enabling superior handling of complex, non-linear trends. Architectures such as Informer, Autoformer, FEDformer, and TimesNet have introduced innovative techniques like seasonal-trend decomposition, frequency-enhanced blocks, and temporal 2D variation modeling to better address the unique challenges of sequential data (Wu et al., 2021; Zhou et al., 2021; Zhou, Ma, Wen, et al., 2022; Zuo et al., 2023). These models have demonstrated encouraging outcomes in several fields, such as financial forecasting, weather forecasting, and energy systems. However, there is still a dearth of thorough research in this field, despite its potential usefulness to time-series forecasting in a variety of domains. Additionally, recent findings question the effectiveness of Transformers in modeling temporal order, suggesting that simple linear models may outperform sophisticated Transformer variants under certain conditions, especially for long-term forecasts (Zeng, Chang, Zhang et al., 2023).

2.2 Applications of the models

Time series forecasting has been widely used in the financial sector to determine stock prices and risk assessments. Khan and Alghulaiakh (2020) stated that statistical models such as Autoregressive Integrated Moving Averages (ARIMA) and its extensions are commonly employed to forecast stock prices, exchange rates, and economic indicators in finance. These models leverage historical price data to identify trends and seasonal patterns, enabling investors to make informed decisions

(Khashei & Hajirahimi, 2017). Hybrid methodologies that integrate statistical techniques and machine learning methods are implemented in finance to improve the predictive performance of financial time series. The study by Wang and Guo (2020) illustrated the effectiveness of integrating ARIMA with XGBoost for predicting stock market volatility resulting in enhanced accuracy compared to traditional methods. On the other hand, hybrid methodologies have also been used to forecast epidemics, showcasing the effectiveness of these models in public health (Chakraborty et al., 2019). Additionally, Zeng, Kaur, Siddagangappa et al. (2023) proposed a hybrid model that uses convolutional neural network (CNN) and Transformers to forecast both short-term and long-term stock prices; the proposed model outperformed the Exponential Moving Average (EMA), ARIMA, and DeepAR.

Additionally, time series forecasting has an extensive usage area in resource management. For example, Zhang et al. (2020) proposed a LightGBM-based model to forecast passenger volumes in transportation systems demonstrating LightGBM's effectiveness in managing complex time series data. Eskandari et al. (2021) used ARIMA, LSTM, and GRU models to predict short-term electricity power load, achieving significant improvements in accuracy. In addition, hybrid approaches integrate various machine learning techniques. Zuo et al. (2023) proposed a model that combines TimesNet with CNNs, and He et al. (2024) developed a model that combines TimesNet, Crossformer, and LSTM to enhance forecasting electricity load.

Although studies on time series forecasting specifically for OTT platforms are limited, related studies have been conducted in adjacent fields, such as website traffic prediction, user engagement modeling, and cloud-based autoscaling for web applications. Time series forecasting models predict website traffic, helping

organizations optimize server capacity, enhance user experience, and schedule maintenance during low-traffic periods. According to Casado-Vara et al. (2021), the LSTM-based time series forecasting models managed to get precise results in predicting the website traffic of Wikipedia pages. Nunnagoppula et al. (2023) and MV et al. (2024) found that LSTM has more accuracy than CNN, ARIMA, Seasonal ARIMA (SARIMA), and K-Nearest Neighbors (KNN) in forecasting the load of Wikipedia pages. According to research (Mettu & Sasikala, 2018), LSTM outperformed ARIMA and Adaptive Boosting Regression in forecasting website traffic. On the other hand, Shelatkar et al. (2020) proposed a model that combines LSTM and RNN with ARIMA to achieve higher accuracy in predicting visits to India on Wikipedia. Similarly, Bi et al. (2021) emphasized the effectiveness of temporal convolutional and ensemble models like LSTM and XGBoost for forecasting Wikipedia traffic. Time series forecasting models are also applied to predict demand for digital services, such as social media platforms. For example, Abu Al-Haija et al. (2019) used Autoregressive Moving Average (ARMA) to predict the monthly active users of Facebook and Twitter. Cloud-based autoscaling is another important application area of time series forecasting for web applications that have fluctuated demand. Almeida et al. (2022) achieved 94% accuracy in CPU load prediction by developing a proactive auto-scaling strategy using ARIMA for an e-commerce platform while outperforming threshold-based auto-scaling methods. Guruge and Priyadarshana (2025) proposed a hybrid model combining Facebook Prophet and LSTM for proactive Kubernetes autoscaling demonstrating up to 90% improvement in prediction accuracy using real-world datasets. Also, Singh (2024) emphasized Facebook Prophet, LSTM, and GB models' impact to improve the accuracy of scaling decisions in cloud-based systems. Jiang (2021) conducted a

comprehensive evaluation of 13 deep neural networks such as LSTM and GRU on the bandwidth dataset of The State University of Ceará. In the telecommunications field, Kougioumtzidis et al. (2025) proposed extensive forecasting research utilizing ES, ARIMA, gradient boosting methods like XGBoost and LightGBM, and Transformers while applying Random Search and Grid Search as hyperparameter tuning strategies.

Forecasting methods have been deployed to support other OTT-related use cases such as popularity predictions, content recommendations, and delivery optimizations. Sá et al. (2021) developed a model to predict the future popularity of content on GloboPlay by implementing a Random Forest-based approach that achieves high accuracy for identifying potentially popular titles in order to improve the recommendation strategies of the platform. Also, Trzciński et al. (2017) proposed a model called Popularity-LRCN which uses a deep recurrent CNNs and LSTM units to classify popular videos by achieving over 30% improvement in predictive performance compared to traditional frameworks. Additionally, Famaey et al. (2013) introduced a predictive cache replacement algorithm based on forecasting content demand that achieved up to 20% improvements in cache hit rates.

Demand forecasting for Catch-up TV services, which form an integral part of OTT delivery optimization, has also been investigated in the literature. Nogueira et al. (2018) explored demand forecasting for Catch-up TV services as part of OTT delivery optimization, focusing on predicting storage and bandwidth requirements for resource management. Their conclusions are still applicable even if their study focuses on the delivery component of OTT platforms and differs from the current work that focuses on live content forecasting. Interestingly, the study found that

Random Forest performed better than other models, suggesting that it could be used in real-time streaming settings where precise resource prediction is essential.

2.3 Comparison to existing studies and contributions

Table 1 and Table 2 provide a structured summary of the existing closest studies to the OTT platform industry like web, network, and mobile traffic forecasting. Table 1 categorizes the studies based on their objectives, the targeted variables, and the used datasets. Table 2 indicates the methodological choices of these studies, mentioning which methods were used and which hyperparameter tuning strategies were applied.

This study contributes a comprehensive forecasting benchmark specifically designed for OTT platform traffic. This study utilizes both proprietary OTT platform data and external sources such as Wikipedia and weather data in order to achieve better evaluation and comparison, unlike most research that focuses on public datasets. Additional temporal and contextual features were generated by implementing feature engineering and domain expertise. This study integrates 27 different methods including statistical models, machine learning algorithms, deep learning architectures, and Transformer-based models while most of the studies limit themselves to a few methodologies. Also, this study employs grid search-based hyperparameter tuning across all methods except Transformer-based models to improve methods' efficiencies. Moreover, the study spans multiple datasets and prediction horizons to enhance the generalizability, applicability, and robustness of the methods and the results. These significant contributions provide OTT platforms with powerful tools to enhance user experience through improved content delivery,

enable more accurate resource planning, and build more efficient infrastructure systems through data-driven decision-making.

Table 1. Literature Review Summary

Reference	Objective	Variable	Dataset(s)
Shelatkar et al. (2020)	Web Traffic Forecasting	Number of visits	Wikipedia
Jiang (2021)	Web Traffic Forecasting	Bandwidth usage	The State University of Ceará
Bi et al. (2021)	Web Traffic Forecasting	Number of visits	Wikipedia
Casado-Vara et al. (2021)	Web Traffic Forecasting	Number of visits	Wikipedia
Nunnagoppula et al. (2023)	Web Traffic Forecasting	Number of sessions	Wikipedia
MV et al. (2024)	Web Traffic Forecasting	Number of visits	Wikipedia
Nogueira et al. (2017)	Catch-up TV Forecasting	Number of requests	MEO
Kougioumtzidis et al. (2025)	Mobile Network Traffic Forecasting	Call, SMS activity	Telecom Italia
This Study	OTT Platform Forecasting	Number of sessions	OTT Platform Wikipedia Weather

Table 2. Forecasting Models and Tuning Methods Used in Literature Review

Reference	Statistical Methods	Machine Learning Methods	Deep Learning Methods	Transformer-based Methods	Hyperparameter Tuning
Shelatkar et al. (2020)	AR, MA, ARIMA	X	LSTM-RNN	X	Grid Search
Jiang (2021)	MA, AR	X	LSTM, GRU, FCN, ResNet, ResCNN, TCN	X	X
Bi et al. (2021)	ARIMA, SARIMA	XGBoost, SVR	TCN, LSTM	X	X
Casado-Vara et al. (2021)	X	X	RNN, LSTM, GRU, TCN	X	X
Nunnagoppula et al. (2023)	X	X	LSTM, CNN	X	Grid Search
MV et al. (2024)	ARIMA, SARIMA	X	LSTM, KNN	X	Grid Search
Nogueira et al. (2017)	X	Random Forest, SVM	BRNN	X	Grid Search
Kougioumtzidis et al. (2025)	ES, ARIMA	XGBoost, LightGBM	LSTM, GRU, TCN, GAN	Vanilla Transformer	Grid Search Random Search
This Study	ES, ARIMAX, SARIMAX	XGBoost, LightGBM, Random Forest, GB	LSTM, GRU, RNN	Autoformer, Crossformer DLinear, FiLM, iTransformer, LightTS Non-stationary Transformer, PatchTST Pyraformer, Reformer, TimesNet TimeXer, TSMixer, WPMixer	Grid Search

CHAPTER 3

METHODOLOGY

The methodological approach to forecasting time series data for an OTT platform is explained in this section. A large dataset from a subscription-based video-on-demand (SVOD) platform is used, along with other publicly available time series datasets, considering the nature of streaming demand. The methodology includes data collection, experimental setup and design, hyperparameter tuning, and a comparative evaluation of statistical, machine learning, and deep learning-based forecasting models. By systematically applying these methods, forecasting frameworks that can handle the dynamic nature of OTT user behavior and streaming patterns are compared.

3.1 Data collection

It is important to know the characteristics and diversity of datasets to iteratively evaluate and improve the time-series forecasting models. Three different datasets are used in this study to ensure coverage of both public and proprietary data sources. To assess the generalizability and robustness of the model, two benchmark datasets from other domains, weather, and Wikipedia page views, are added to the primary dataset, which originates from a large-scale OTT platform.

OTT Dataset: The user activity data is obtained from an undisclosed SVOD platform with more than 500,000 users in Turkey. The platform offers movies, series, and documentaries in its video library, the top 50 FTA (Free-to-Air) channels, and 35 premium TV channels, including live sports, film, and entertainment content.

The dataset includes hourly viewership data aggregated from client-side applications, including the website, mobile applications, and connected TV (CTV) devices. This data was collected between 2022 and 2024 and includes channel information, streaming durations, geographic distribution, and device types.

The data is stored in a NoSQL database, which is optimized for handling large-scale time-series data. Queries were executed to extract and aggregate the necessary data indexes for analysis. To prepare the dataset for analysis, missing values were handled using both forward and backward-filling techniques to maintain temporal integrity. Feature engineering involved extracting key temporal attributes, such as day of the week, hour of the day, holiday indicators, and significant external events. Notably, the Turkey-Syria earthquakes on 6 February 2023 were marked as a critical timestamp, followed with high-volume of live TV channel viewership. Additionally, sports events on the platform were categorized into four tiers based on content availability and significance: FTA-Tier 1 (FTA content featuring major events such as matches of the Turkish national football and basketball teams or top-tier Turkish football and basketball clubs), FTA-Tier 2 (other FTA channel events), Premium-Tier 1 (Exclusive content featuring major matches or events of high significance, such as international games or matches involving top-tier teams), and Premium-Tier 2 (less significant exclusive sports events). To avoid biases in model predictions that could result from features being on different scales, all features were normalized to a [0,1] scale ensuring comparability across models. The resulting dataset was then divided with a 70-10-20 train-validation-test split, allowing for a thorough and robust evaluation of the models.

The Weather Dataset: The data collected by the National Oceanic and Atmospheric Administration (NOAA), includes climatological records from 1,600

U.S. locations over four years (2010-2013) with hourly readings of the target variable "wet bulb" alongside 11 climate features.

The Wikipedia Dataset: The user activity data was retrieved using the Wikimedia Analytics API (Pageviews), with parameters set to aggregate hourly page views across all access types, agents, and pages for the English Wikipedia. The data was collected from January 1, 2022, to January 1, 2024, which aligns with the time frame of the OTT platform dataset.

3.2 Forecasting models

A range of statistical, machine learning, and deep learning models were implemented to compare forecasting effectiveness. Using weighted averages of historical observations, ES is a straightforward but powerful forecasting method that gives more weight to recent data points. It works especially well with time series data that show seasonal patterns and linear trends. The autoregressive and moving average components are combined in the ARMA model which has a linear function of previous values as the autoregressive part and past error terms as the moving averages part is assumed to be able to explain the data (Abu Al-Haija et al., 2019). ARIMA extends the capabilities of the ARMA model by introducing different steps to handle non-stationary data (Wang & Guo, 2020). Differencing means subtracting the previous value from the current one to stabilize the data. SARIMA is an extension of ARIMA that can handle recurring seasonal trends, optimizing performance in periodic datasets (Khashei & Hajirahimi, 2017). ARIMAX and SARIMAX models have emerged as valuable extensions of ARIMA and SARIMA, respectively in order to enhance the forecasting capabilities. The ARIMAX model incorporates one or more predictor (exogenous) variables into the ARIMA

framework, enhancing its performance in various contexts such as load forecasting, where it has been shown to yield superior accuracy compared to ARIMA alone (Lee & Cho, 2022). Similarly, SARIMAX has illustrated its efficacy in capturing both seasonal and external influences on the time series (Maku et al., 2023).

Random Forest is a robust ensemble learning method introduced by Breiman (2001), which constructs multiple decision trees during training time and aggregates the outputs through average prediction for regression or majority voting for classification. This method improves prediction performance by avoiding overfitting and increasing generalization through randomization in both future selection and data sampling. GB, initially proposed by Friedman (2001), creates predictive models in a sequential manner where each new model is trained to correct the errors of the combined ensemble. Although it optimizes a loss function using gradient descent, making it effective for complex data, GB can be computationally expensive and vulnerable to overfitting if not properly organized. LightGBM, developed by Ke et al. (2017) at Microsoft Research, is an efficient implementation of GB that introduces histogram-based decision trees. XGBoost introduced by Chen and Guestrin (2016), is a scalable and regularized implementation of gradient boosting machines. LightGBM uses a leaf-wise tree growth strategy, which makes it faster and more memory-efficient, especially for large datasets. XGBoost, on the other hand, employs a depth-wise approach and includes additional regularization techniques to improve accuracy and prevent overfitting. Both are widely used in competitions and real-world applications due to their speed and predictive power.

LSTM networks were proposed by Hochreiter and Schmidhuber (1997) to overcome the limitations of traditional RNNs, especially the problem of vanishing gradients. It uses memory cells and gating mechanisms to retain important

information over time, making it highly effective for tasks like time-series forecasting, text generation, and speech recognition. GRU introduced by Cho et al. (2014), is a streamlined variant of LSTM networks. It uses fewer gates than LSTM, making it computationally faster while still maintaining strong performance in modeling sequential data.

The Transformer architecture, initially proposed by Vaswani et al. (2017), introduced self-attention mechanisms for capturing long-range dependencies in sequential data and allowing for parallelization during training. This capability increased the popularity of Transformer-based models for time series forecasting. Various extensions and innovations have been developed to improve its applicability and performance in diverse tasks. Autoformer (Wu et al., 2021) presents a seasonal decomposition-based Transformer architecture specifically designed for long-term time series forecasting. Crossformer (Zhang & Yan, 2023) advances the Transformer by explicitly modeling cross-time and cross-dimension dependencies in multivariate time series. FiLM (Zhou, Ma, Wang, et al., 2022) proposes frequency-domain processing combined with memory mechanisms to improve forecasting efficiency, particularly in capturing periodic components. iTransformer (Liu et al., 2024) introduces an inverted Transformer architecture tailored for time-series forecasting by reconciling variate-level representations. Informer (Zhou et al., 2021) addresses scalability with a ProbSparse self-attention mechanism and a generative decoder, making it suitable for long-sequence forecasting. FEDFormer (Zhou, Ma, Wen, et al., 2022) combines seasonal-trend decomposition with frequency-domain attention to enhance both accuracy and efficiency. Pyraformer (Liu, Yu, et al., 2022) leverages pyramidal attention to model long-range dependencies with linear time complexity. Reformer (Kitaev et al., 2020) improves scalability by using locality-sensitive

hashing for approximate attention, reducing the complexity to O(L log L). PatchTST (Nie et al., 2023) captures localized temporal patterns by dividing input series into patches and applying self-attention across them. TimesNet (Wu et al., 2023) introduces a 2D variation modeling framework to extract both intra- and inter-period features from multiscale time series. TSMixer (Chen et al., 2023) offers a lightweight design using multilayer perceptron and mixing operations to capture temporal patterns efficiently. TimeXer (Wang et al., 2024) enhances the vanilla Transformer by incorporating exogenous variables via patch-wise and variate-wise attention. DLinear (Zeng, Chen, Zhang, et al., 2023) challenges the efficacy of complex attention mechanisms by showing that simple linear projections often outperform Transformers in long-term forecasting tasks. LightTS (Zhang et al., 2022) further simplifies the forecasting pipeline by using lightweight MLPs over sampled inputs, demonstrating high accuracy with minimal computational cost. Non-Stationary Transformer (Liu, Wu, et al., 2022) dynamically adjusts attention to capture shifting temporal patterns in non-stationary data. Finally, WPMixer (Murad et al., 2025) extends the mixer-based architecture by integrating windowed temporal information, offering strong performance in low-data and high-variance settings.

3.2.1 Evaluation metrics

The models were assessed using the following standard time-series forecasting evaluation metrics:

MAE: Measures the average absolute differences between predicted and actual values, providing an intuitive understanding of prediction accuracy.

MSE: Measures the average squared differences between predicted and actual values, penalizing larger errors more heavily.

These metrics were chosen for their relevance in assessing both the accuracy and consistency of forecasting models. Runtime was also recorded as a secondary evaluation criterion to assess computational efficiency and practical applicability in real-time OTT operations.

3.3 Experimental setup

The design and setup of the extensive experimental framework used to assess the effectiveness of different time series forecasting models are described in this section. The experiments evaluate the model's ability to forecast traffic to OTT platforms and compare it to publicly accessible datasets, such as Wikipedia page views and weather records. Establishing a benchmark for comparison across statistics, machine learning, and deep learning forecasting approaches under various circumstances and data complexity is the aim.

To provide a fair comparison, models were trained and tested under identical experimental conditions, and hyperparameters were fine-tuned based on previous studies and optimization strategies. The simulation framework including data transformation, model training, and performance evaluation was executed on a personal HP Victus Gaming Laptop with an NVIDIA RTX 4070 GPU running Ubuntu 20.04.6 LTS. All experiments were implemented using Python 3.8, PyTorch 1.10, and CUDA 11.2 as the deep learning environment. In this virtual environment, the following packages were installed:

- EinOps (Einstein Operations) 0.8.0
- Local Attention 1.9.14
- Matplotlib 3.7.0
- Numpy 1.23.5

- Pandas 1.5.3
- Sklearn (Scikit-Learn) 1.2.2
- SciPy 1.10.1
- SkTime 0.16.1
- SymPy 1.11.1
- Torch 1.7.1
- Plotly 6.0.1
- StatsModels 0.14.4
- LightGBM 4.6.0
- XGBoost 3.0.0
- TensorFlow 2.19.0

3.3.1 Experimental design

Traditional statistical models (ARIMAX, SARIMAX, ES, etc.) were configured based on time series stationarity checks and autocorrelation plots. Machine learning models (RF, XGBoost, GB, and LightGBM) incorporated lag-based features and were trained using a grid search for hyperparameter tuning. Deep learning models (LSTM, GRU, etc.) were trained with adaptive learning rate scheduling, dropout regularization, and early stopping on the validation set.

Additionally, for implementation consistency and reproducibility, the Time Series Library (TSLib) developed by Tsinghua University Machine Learning Group (THUML) was used as a base framework for several deep learning and Transformer-based models, including TimesNet, Autoformer, PatchTST, MICN, and others[1]. The TSLib repository offers modular implementations, standardized data loaders, and

[1] https://github.com/thuml/Time-Series-Library

benchmarking utilities, which help streamline model training, tuning, and evaluation. Wherever applicable, TSLib's default configurations were adapted to fit the dataset characteristics in this study while ensuring fairness by applying uniform preprocessing, sliding window configurations, and evaluation metrics across all model variants.

A standardized sliding window technique was used for every model. To guarantee consistency and comparability between models, input sequence lengths (seq len) of 168 hours (7 days) and 24 hours (1 day), as well as forecast horizons (pred len) of 12, 24, and 36 hours, were specifically selected.

3.3.2 Hyperparameter tuning

In time series forecasting, hyperparameter adjustment is essential. It ensures generalization across various data sets and uses scenarios, which increases the models' accuracy and predictive potential. Depending on the model classifications, different optimization techniques were used in this work, ranging from Transformer-based designs to statistical models. Table 3 shows the hyperparameter spaces that have been prepared for each forecasting model.

Grid Search was used as a methodical technique for hyperparameter optimization for the OTT dataset in order to optimize each algorithm's parameters. Within a predetermined search space, Grid Search assesses every potential combination of hyperparameters and chooses the one that performs best according to a predetermined evaluation metric (such as MAE or MSE). The hyperparameter tuning procedure for each algorithm utilized in this investigation is explained below.

Traditional statistical models that depend on parameter selection include ARIMAX and SARIMAX. Seasonality is used in the SARIMAX model, which

expands upon simpler linear models. The seasonal orders (P, D, Q, s) and the non-seasonal orders (p, d, q) are the hyperparameters that need to be adjusted. All possible combinations of $p \in \{0, 1, 2\}$, $d \in \{0, 1\}$, $q \in \{0, 1, 2\}$, $P \in \{0, 1\}$, $D \in \{0, 1\}$, $Q \in \{0, 1\}$, and $s = 12$ were searched using a grid. The best combination of (p, d, q) and (P, D, Q) was determined by minimizing the MAE and MSE.

Random Forest is an ensemble-based approach that generates multiple decision trees during the training process. Its primary hyperparameters include the number of trees (n_estimators), the maximum depth allowed for each tree (max_depth), and the minimum number of samples required to split an internal node (min_samples_split). The search space for tuning included n_estimators values of 50, 100, and 200; max_depth values of None, 10, and 20; and min_samples_split values of 2, 5, and 10. LightGBM, on the other hand, is a GB algorithm that utilizes histogram-based learning. The key hyperparameters for this model were num_leaves (set to 31, 50, or 100), learning_rate (with values 0.01, 0.1, or 0.2), and n_estimators (ranging from 100 to 300). XGBoost, a widely used GB framework, was fine-tuned across a set of hyperparameters, including max_depth values of 3, 5, and 7; learning rate values of 0.01, 0.1, and 0.2; and n_estimators values of 100, 200, and 300. GB models, which generate trees in a sequential manner where each new tree corrects the errors of the previous ones, were tuned using the same hyperparameter ranges: $n\_estimators \in \{100, 200, 300\}$, $learning\_rate \in \{0.01, 0.1, 0.2\}$, and $max\_depth \in \{3, 5, 7\}$.

For deep learning models such as LSTM, GRUs, and RNN, optimization was performed using a hyperparameter grid that included $units \in \{50, 100, 200\}$, $learning\_rate \in \{0.001, 0.01, 0.1\}$, and $batch\_size \in \{32, 64, 128\}$.

Table 3. Hyperparameter Spaces for Forecasting Models

Model	Hyperparameter	Search Space
ARIMAX	p (AR order) d (Diff. order) q (MA order)	0, 1, 2 0, 1 0, 1, 2
SARIMAX	p (Non-seasonal AR) d (Non-seasonal diff.) q (Non-seasonal MA) P (Seasonal AR) D (Seasonal diff.) Q (Seasonal MA)	0, 1, 2 0, 1 0, 1, 2 0, 1 0, 1 0, 1
Random Forest	n_estimators max_depth (max_d) min_samples_split	50, 100, 200 0, 10, 20 2, 5, 10
GB	max_depth learning_rate n_estimators	3, 5, 7 0.01, 0.1, 0.2 100. 200, 300
XGBoost	max_depth learning_rate n_estimators	3, 5, 7 0.01, 0.1, 0.2 100. 200, 300
LightGBM	num_leaves learning_rate n_estimators	31, 50, 100 0.01, 0.1, 0.2 100, 200, 300
Deep Learning Methods	units learning_rate batch_size	50, 100, 200 0.001, 0.01, 0.1 32, 64, 128

TSLib developed by the THUML, was utilized to implement Transformer-based models such as Autoformer, Informer, TimesNet, PatchTST, MICN, Crossformer, and others. In line with a consistent experimental approach across datasets, tuning was restricted to configurations with seq_len ∈ {168, 24} and pred_len ∈ {12, 24, 36}, given the high computational demands and architectural complexity of these models.

CHAPTER 4

RESULTS

This section outlines the predictive performance of several time-series forecasting models evaluated on three datasets: a proprietary OTT platform traffic dataset, Wikipedia page views, and a multivariate Weather dataset. The models were assessed using MAE, MSE, and runtime (in seconds) for all three datasets. A uniform sliding window strategy was employed during training, using input sequence lengths of either 24 or 168 and forecast horizons of 12, 24, or 36 time steps.

4.1 Exploratory data analysis

In order to understand the temporal behavior of the OTT traffic data, an exploratory data analysis was performed. This section summarizes the findings of data in terms of distribution, seasonality, and periodicity.

Figure 1 illustrates the hourly distribution over the entire observation period, which shows strong fluctuations with occasional peaks reaching extremely high values. These peaks are mostly related to the sports games or nation-wide events. Figure 2 and Figure 3 show the number of sessions aggregated daily and weekly respectively. Although there is some variability across weeks in Figure 3, the overall traffic volume demonstrates a relatively stable pattern. A finer granularity is shown in Figure 2 with higher frequency variations and visible peaks similar to Figure 1.

Figure 4 displays the total number of sessions by hour. Viewership sharply increases between 20:00 – 22:00 which aligns with the prime viewing hours and overnight activity drops significantly to the lowest levels between 00:00 and 05:00.

The consistent daily cycle is also supported by the autocorrelation and partial autocorrelation plots in Figure 7 and Figure 8 respectively.

Figure 5 compares total session counts across days of the week. Sunday gets the highest volume followed by Tuesday and Wednesday while Friday gets the lowest volume. Figure 6 presents a weekly-hourly heatmap to visualize the joint effect of the day and hour. This heatmap indicates that the user activity is low in the early morning hours and sharply increases between 18:00 and 22:00 every day while the busiest periods occur between 21:00 and 23:00 on weekdays.

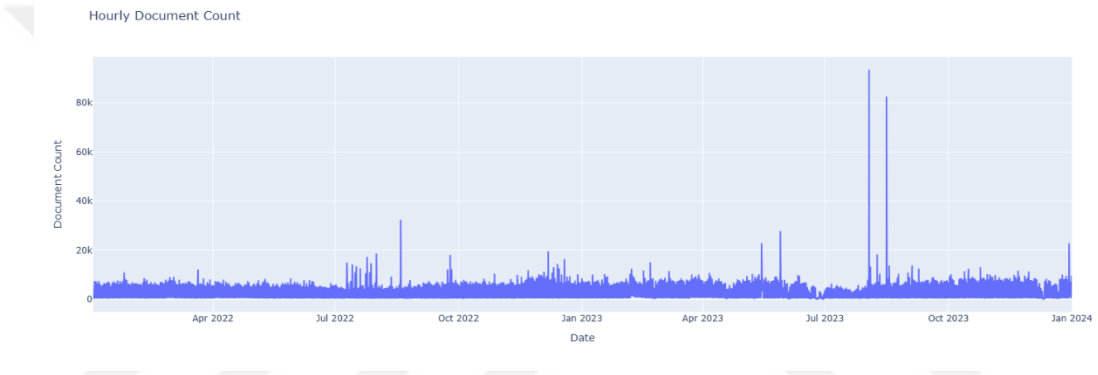

Figure 1. Hourly session distribution

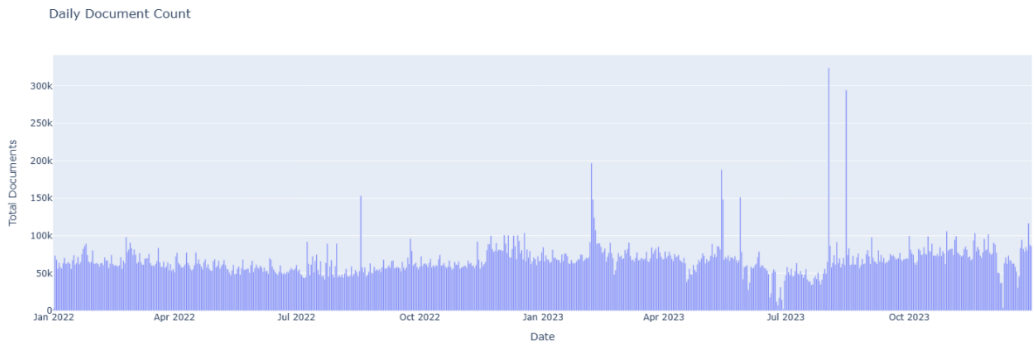

Figure 2. Daily aggregated session distribution

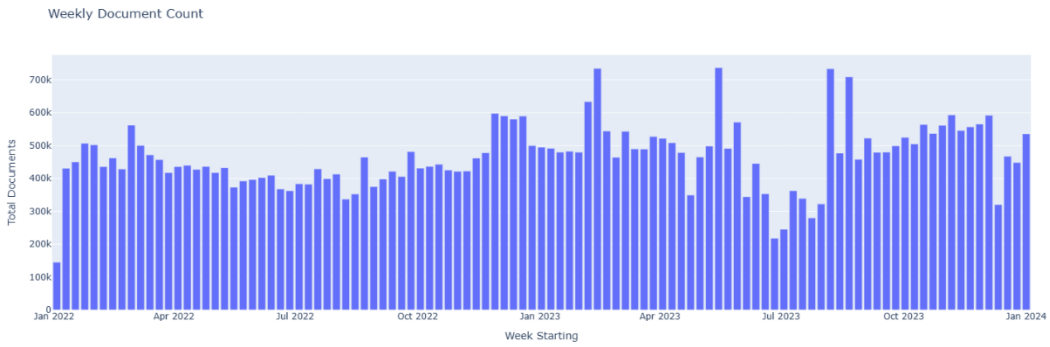

Figure 3. Weekly aggregated session distribution

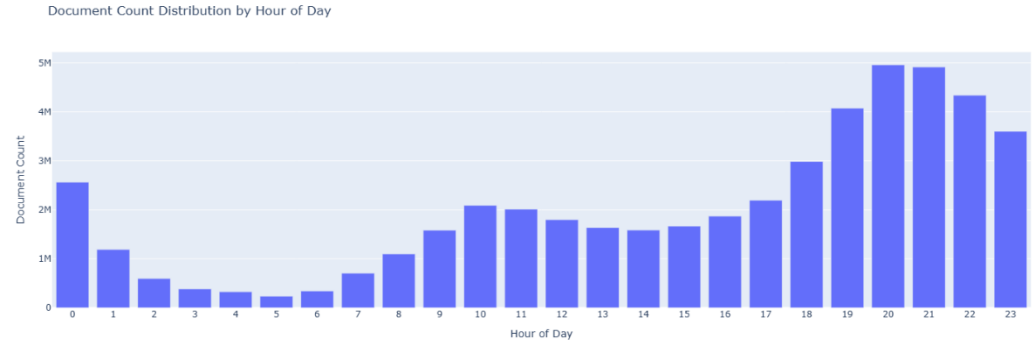

Figure 4. Session distribution by hours of the day

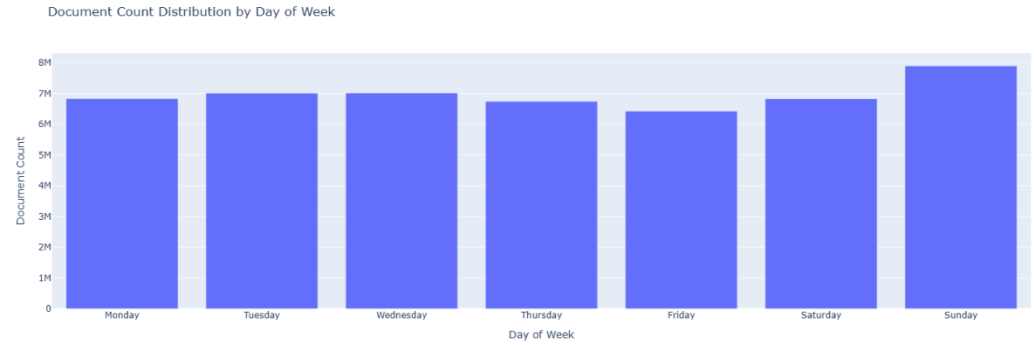

Figure 5. Session distribution by days of the week

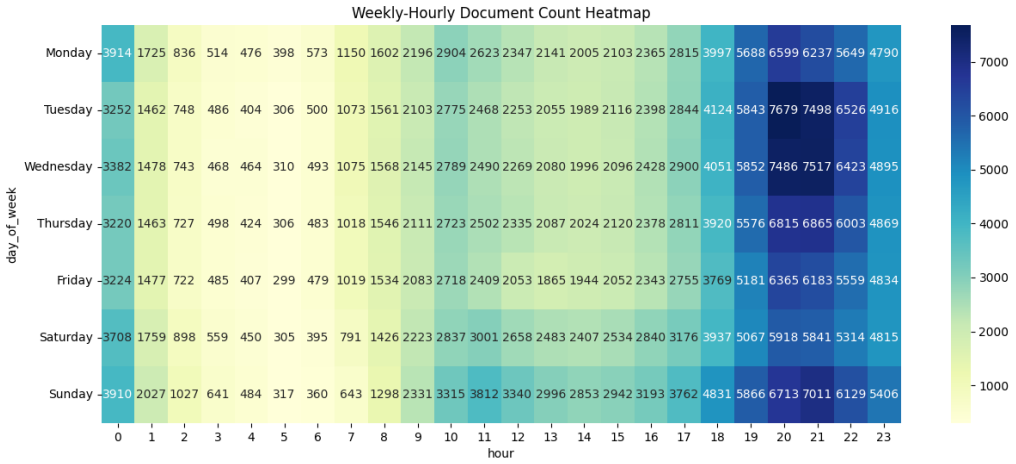

Figure 6. Weekly-Hourly Session Heatmap

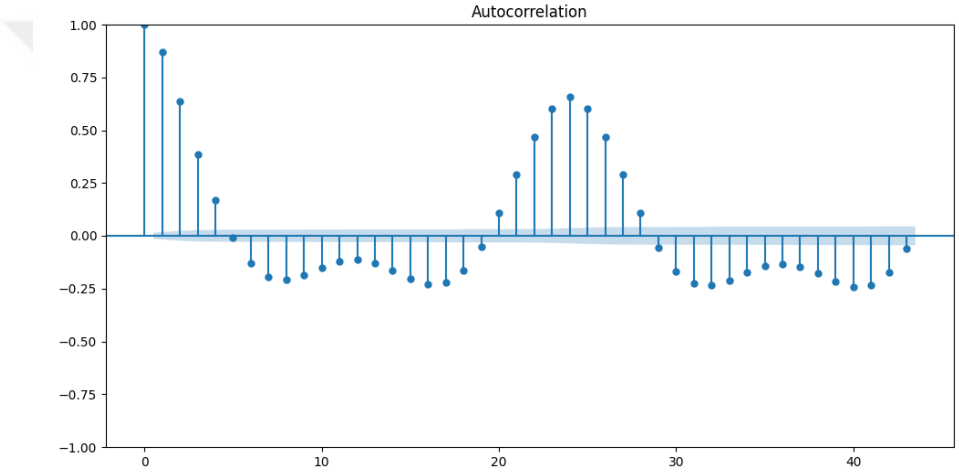

Figure 7. Autocorrelation Function Plot

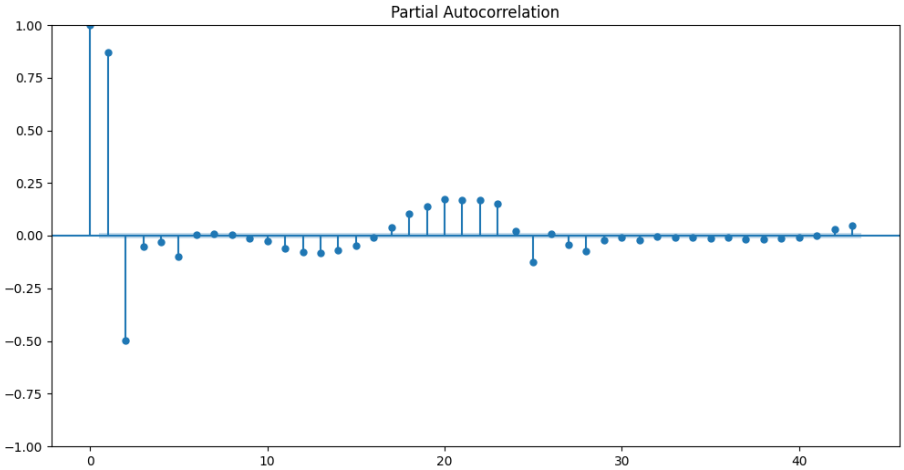

Figure 8. Partial Autocorrelation Function Plot

4.2 Hyperparameter tuning results

The effectiveness of forecasting models is highly dependent on their hyperparameter configurations. A systematic hyperparameter tuning procedure was carried out for each forecasting model to improve prediction performance. The tuning process employed grid search to systematically explore the parameter space while ensuring optimal performance across different model architectures. The tuning aimed to minimize MAE and MSE while observing runtime efficiency.

ARIMAX models showed moderate performance with the best configuration ($p = 0$, $d = 0$, $q = 0$) achieving an MAE of 0.604. Performance typically got worse as the model got more complicated, especially with differencing ($d = 1$), where MAE values went up a lot (from 0.767 to 0.978). The inclusion of moving average components also tended to increase error rates, suggesting that simpler ARIMAX configurations were more suitable for this dataset. The complete ARIMAX parameter tuning results are documented in Appendix B, Table B1. SARIMAX models exhibited extreme variability in performance, ranging from excellent results (MAE as low as 0.233) to very poor performance (MAE exceeding 4.0). The best-performing configurations consistently included seasonal differencing ($D = 1$) and seasonal moving average terms ($Q = 1$), with the optimal configuration being ($p = 0$, $d = 0$, $q = 0$) ($P = 0$, $D = 1$, $Q = 1$) achieving MAE of 0.233. This suggests that SARIMAX requires very careful parameter selection and may not be robust for this dataset. Detailed SARIMAX configuration results and convergence analysis are provided in Appendix B, Table B2.

Random Forest demonstrated remarkable stability across different hyperparameter configurations. The optimal configuration achieved an MAE of 0.152 with parameters (n_estimators = 50, max_depth = 20, min_samples_split = 5).

Notably, Random Forest showed minimal performance variation across all tested configurations, with MAE values ranging only from 0.152 to 0.157, confirming its robustness. The complete Random Forest hyperparameter tuning results are detailed in Appendix B, Table B3. XGBoost achieved the best overall performance among all models, with the lowest MAE of 0.150 using configuration (max_depth = 7, learning_rate = 0.1, n_estimators = 100). The results revealed a clear pattern where learning rates of 0.1 consistently outperformed both 0.01 and 0.2 across all depth settings. GB followed similar patterns to XGBoost but with generally higher error rates. The best configuration (max_depth = 3, learning_rate = 0.1, n_estimators = 300) achieved an MAE of 0.152. The model showed extreme sensitivity to learning rate, with configurations using learning_rate = 0.01 performing very poorly (MAE up to 0.361), demonstrating the critical importance of proper learning rate selection for traditional gradient boosting implementations. The complete parameter tuning analysis for GB and XGBoost are represented in Appendix B, Table B4, and Table B5 respectively. LightGBM achieved competitive performance with an MAE of 0.159 (leaves = 50, learning_rate = 0.1, n_estimators = 100) while maintaining exceptional computational efficiency. All LightGBM configurations were completed in under 1 second, making it the fastest algorithm tested. The full set of LightGBM hyperparameter results can be found in Appendix B, Table B6.

Deep Learning methods (GRU, RNN, and LSTM) showed the best with moderate complexity configurations. The optimal setup (units = 100, learning_rate = 0.001, batch_size = 64) achieved MAE of 0.177. Smaller batch sizes (32) generally outperformed larger ones (128) and learning rates of 0.001 consistently provided better results than 0.01 or 0.1. Very high learning rates (0.1) led to poor convergence, with some configurations achieving MAE values exceeding 0.5. The comprehensive

GRU architecture evaluation and training dynamics are detailed in Appendix B, Table B7.

Across all model categories, optimal hyperparameters contributed to noticeable performance improvements. This validates the role of hyperparameter tuning as a critical component of time-series forecasting pipelines for OTT platforms. These results are consistent with prior work, such as Weerts et al. (2020), which emphasizes the performance gaps between default and tuned configurations.

4.3 Forecasting accuracy

4.3.1 OTT dataset

The OTT dataset, which captures session activity from an internet-based video streaming platform, was used to evaluate a variety of time-series forecasting models. These included classical statistical approaches, machine learning techniques, deep learning architectures such as RNNs, and advanced Transformer-based frameworks. Model performance was measured using MAE, MSE, and runtime (in seconds), with all runtime assessments conducted under standardized computational settings to ensure a fair comparison of training and inference efficiency. Tables 4 through 9 summarize the corresponding MAE, MSE, and runtime performances of these models.

For short-term forecasting tasks with 24 hours sequence length, FEDFormer consistently outperformed other models across all prediction intervals. At a 12-hour forecast horizon, it yielded the most accurate results (MAE: 0.1408, MSE: 0.2675), however, there is a high computational cost of 757 seconds. Autoformer (MAE: 0.2073, MSE: 0.4618) and LightGBM (MAE: 0.2436, MSE: 0.709) also delivered competitive accuracy but LightGBM notably produced results in just 1 second in this

experimental environment. Table 4 displays the results of each model for a 24-hour sequence length with a 12-hour forecast horizon, while Tables 5 and 6 give the results for a 24-hour sequence length with 12-hour and 36-hour forecast scenarios, respectively. When the forecasting horizon is 24 hours, FEDFormer again achieved the best accuracy (MAE: 0.1949, MSE: 0.4505), while GB offered a solid balance between speed and precision (MAE: 0.2434, MSE: 0.7231) with a runtime of 192 seconds. For the 36-hour forecast, FEDFormer retained the top spot in accuracy (MAE: 0.2228, MSE: 0.5773), although iTransformer emerged as a faster alternative (MAE: 0.2682, MSE: 0.7707) with significantly reduced computation time (99 seconds), making it a feasible choice for applications requiring faster output.

In modeling weekly temporal patterns with 168 hours of sequence length, TimesNet demonstrated the highest forecasting accuracy, albeit at a significant computational cost. For the 12-hour forecast, it produced the lowest error rates (MAE: 0.1866, MSE: 0.4376), but required 7,894 seconds to complete. A more efficient alternative was iTransformer, which delivered comparable performance (MAE: 0.2005, MSE: 0.5078) with a much faster runtime of just 87 seconds. Table 7 presents the outcomes of each model for a 168-hour sequence length with a 12-hour forecast horizon, whereas Table 8 and Table 9 provide the results for a 168-hour sequence length with 12-hour and 36-hour forecast scenarios, respectively. When predicting 24 hours ahead, TimesNet continued to outperform other models (MAE: 0.2478, MSE: 0.6424), although the Non-stationary Transformer provided a more runtime-efficient solution (MAE: 0.2407, MSE: 0.6506) in 256 seconds. At the 36-hour prediction interval, TimesNet again yielded the best accuracy (MAE: 0.3030, MSE: 0.7954), while LightGBM offered the quickest inference (16 seconds) with

only a moderate trade-off in accuracy (MAE: 0.2573, MSE: 0.7362), making it particularly suitable for time-sensitive applications.

Transformer-based architectures such as FEDFormer and TimesNet consistently delivered top-tier accuracy but required significantly longer runtimes (ranging from 291 to 9,296 seconds). In contrast, tree-based models like LightGBM and XGBoost demonstrated faster inference times, from a few seconds up to several minutes, while maintaining acceptable accuracy levels.

Among traditional statistical models, ARIMAX executed relatively quickly (39 seconds) but exhibited moderate accuracy (MAE: 0.791), while SARIMAX was both slower (3,692 seconds) and less accurate (MAE: 1.103), making it unsuitable for time-sensitive applications. Exponential Smoothing remained the fastest method (0 seconds), yet its accuracy (MAE: 1.034) lagged significantly behind modern alternatives.

Overall, the results affirm that Transformer-based models are the most suitable for high-accuracy forecasting in complex OTT scenarios, especially for batch processing environments. Tree-based models like LightGBM and XGBoost, however, strike a better balance between performance and speed for real-time use cases. Classical models like ARIMAX and SARIMAX may still offer value for simpler or seasonal datasets, but their effectiveness diminishes in dynamic, non-stationary contexts.

Table 4. OTT Forecasting Results for seq_len = 24 and pred_len = 12

Model	MAE	MSE	Time (s)
ARIMAX	0.791	2.121	39
SARIMAX	1.103	3.709	3692
ES	1.034	2.815	0
RandomForest	0.238	0.714	92
GB	0.234	0.694	96
LightGBM	0.244	0.709	1
XGBoost	0.233	0.699	109
LSTM	0.245	0.674	538
GRU	0.254	0.697	477
RNN	0.727	1.526	147
Autoformer	0.207	0.462	291
Crossformer	0.229	0.594	467
DLinear	0.355	1.077	19
FEDFormer	0.141	0.267	757
FiLM	0.335	1.094	138
Informer	0.218	0.523	203
iTransformer	0.192	0.547	101
LightTS	0.243	0.559	61
Non-stationary Transformer	0.185	0.571	213
PatchTST	0.245	0.717	114
Pyraformer	0.207	0.510	117
Reformer	0.213	0.545	120
TimesNet	0.163	0.503	2157
TimeXer	0.181	0.530	162
Transformer	0.196	0.536	190
TSMixer	0.212	0.329	42
WPMixer	0.245	0.782	86

Table 5. OTT Forecasting Results for seq_len = 24 and pred_len = 24

Model	MAE	MSE	Time (s)
ARIMAX	0.741	2.013	40
SARIMAX	1.179	3.991	3558
ES	1.006	2.828	0
RandomForest	0.249	0.739	239
GB	0.243	0.723	192
LightGBM	0.256	0.738	2
XGBoost	0.242	0.723	188
LSTM	0.256	0.713	1062
GRU	0.261	0.732	942
RNN	0.693	1.421	291
Autoformer	0.264	0.623	332
Crossformer	0.243	0.645	472
DLinear	0.357	1.122	19
FEDFormer	0.195	0.450	706
FiLM	0.328	1.159	241
Informer	0.246	0.625	201
iTransformer	0.240	0.701	100
LightTS	0.276	0.744	62
Non-stationary Transformer	0.237	0.769	214
PatchTST	0.269	0.804	114
Pyraformer	0.233	0.610	114
Reformer	0.243	0.656	109
TimesNet	0.207	0.627	2609
TimeXer	0.227	0.667	165
Transformer	0.228	0.648	188
TSMixer	0.256	0.540	58
WPMixer	0.267	0.845	95

Table 6. OTT Forecasting Results for seq_len = 24 and pred_len = 36

Model	MAE	MSE	Time (s)
ARIMAX	0.754	1.966	40
SARIMAX	1.223	4.079	3696
ES	1.025	2.891	0
RandomForest	0.258	0.755	386
GB	0.253	0.746	288
LightGBM	0.266	0.756	3
XGBoost	0.251	0.742	302
LSTM	0.263	0.733	1616
GRU	0.266	0.745	1431
RNN	0.654	1.362	443
Autoformer	0.279	0.692	535
Crossformer	0.260	0.691	480
DLinear	0.375	1.134	14
FEDFormer	0.223	0.577	992
FiLM	0.364	1.217	142
Informer	0.270	0.717	179
iTransformer	0.268	0.771	99
LightTS	0.293	0.814	56
Non-stationary Transformer	0.274	0.869	172
PatchTST	0.285	0.839	113
Pyraformer	0.262	0.700	107
Reformer	0.274	0.733	116
TimesNet	0.245	0.732	4657
TimeXer	0.253	0.745	159
Transformer	0.275	0.762	154
TSMixer	0.286	0.691	47
WPMixer	0.283	0.858	86

Table 7. OTT Forecasting Results for seq_len = 168 and pred_len = 12

Model	MAE	MSE	Time (s)
ARIMAX	0.501	1.251	38
SARIMAX	0.795	2.050	1741
ES	0.432	15.040	176
RandomForest	0.238	0.708	898
GB	0.247	0.728	662
LightGBM	0.241	0.694	5
XGBoost	0.227	0.689	159
LSTM	0.275	0.706	3110
GRU	0.350	0.812	2608
RNN	0.638	1.309	803
Autoformer	0.240	0.380	950
Crossformer	0.265	0.652	216
DLinear	0.298	0.854	21
FEDFormer	0.222	0.326	1667
FiLM	0.320	0.948	213
Informer	0.255	0.579	299
iTransformer	0.200	0.508	87
LightTS	0.242	0.545	49
Non-stationary Transformer	0.188	0.500	252
PatchTST	0.250	0.679	270
Pyraformer	0.206	0.509	299
Reformer	0.212	0.529	388
TimesNet	0.187	0.438	7894
TimeXer	0.209	0.487	222
Transformer	0.204	0.517	219
TSMixer	0.209	0.347	45
WPMixer	0.249	0.693	602

Table 8. OTT Forecasting Results for seq_len = 168 and pred_len = 24

Model	MAE	MSE	Time (s)
ARIMAX	0.469	1.015	40
SARIMAX	0.776	1.814	1746
ES	0.508	44.480	176
RandomForest	0.247	0.728	1269
GB	0.253	0.748	1331
LightGBM	0.252	0.725	11
XGBoost	0.237	0.710	320
LSTM	0.316	0.810	6071
GRU	0.353	0.838	5192
RNN	0.756	1.564	1672
Autoformer	0.321	0.599	640
Crossformer	0.239	0.721	599
DLinear	0.311	0.892	15
FEDFormer	0.257	0.496	1662
FiLM	0.324	0.941	511
Informer	0.271	0.669	276
iTransformer	0.246	0.693	66
LightTS	0.281	0.709	61
Non-stationary Transformer	0.241	0.651	256
PatchTST	0.263	0.834	270
Pyraformer	0.232	0.610	298
Reformer	0.254	0.653	401
TimesNet	0.248	0.642	5696
TimeXer	0.239	0.629	222
Transformer	0.241	0.631	222
TSMixer	0.267	0.631	51
WPMixer	0.264	0.768	423

Table 9. OTT Forecasting Results for seq_len = 168 and pred_len = 36

Model	MAE	MSE	Time (s)
ARIMAX	0.472	0.816	40
SARIMAX	0.772	1.607	1747
ES	0.588	91.228	177
RandomForest	0.253	0.740	1877
GB	0.258	0.756	1994
LightGBM	0.257	0.736	16
XGBoost	0.242	0.722	497
LSTM	0.287	0.757	9120
GRU	0.323	0.788	7772
RNN	0.738	1.487	2554
Autoformer	0.318	0.702	651
Crossformer	0.238	0.692	628
DLinear	0.329	0.910	16
FEDFormer	0.284	0.629	1652
FiLM	0.330	0.933	403
Informer	0.302	0.740	280
iTransformer	0.261	0.752	107
LightTS	0.293	0.780	44
Non-stationary Transformer	0.269	0.707	244
PatchTST	0.299	0.817	157
Pyraformer	0.259	0.691	299
Reformer	0.282	0.732	401
TimesNet	0.303	0.795	6764
TimeXer	0.258	0.728	157
Transformer	0.274	0.758	234
TSMixer	0.307	0.726	34
WPMixer	0.277	0.803	304

4.3.2 Wikipedia dataset

To assess the generalizability of the forecasting models under evaluation, an extensive benchmarking analysis was performed using the Wikipedia page views dataset. The Wikipedia Dataset serves as a valuable alternative for OTT traffic patterns, as it includes hourly-based web traffic for the same period. Similar to OTT data, Wikipedia page views demonstrate significant fluctuations and irregular patterns which include periodic changes. Wikipedia traffic most closely replicates the complexities encountered in OTT environments, particularly regarding non-stationarity, seasonality, and sharp demand changes among publicly available sources. Consequently, it offers a valuable external validation framework for examining the transferability of models trained on proprietary OTT data to equally dynamic, open-domain scenarios.

All models were trained using a standardized sliding window approach, utilizing input sequence lengths of 24 and 168, along with a fixed prediction horizon of 24 hours. This configuration was selected to assess model robustness in both short-term and long-term forecasting scenarios through all the models. It also allowed for the evaluation of how each model responds to frequent temporal fluctuations and non-linear spikes in content popularity commonly observed in online user engagement data. Transformer-based models consistently outperformed classical statistical and machine learning approaches, especially in 24-hour sequence-length scenarios. Table 10 and Table 11 represent the results of each model for a 24-hour forecast horizon with 24-hour and 1-week sequence lengths respectively. Autoformer achieved the lowest MAE of 0.1967 and an MSE of 0.8909, offering an effective balance between accuracy and computational efficiency, with a runtime of 546 seconds. FEDFormer, while slightly less accurate in terms of MAE (0.1981),

recorded the lowest MSE (0.7119), indicating enhanced effectiveness in handling outliers, although it required a longer training duration of 868 seconds. In comparison, ensemble-based machine learning models such as GB delivered reasonably accurate forecasts (MAE: 0.2270, MSE: 1.5310) with significantly faster inference times of 14 seconds. XGBoost and Random Forest also produced consistent results while maintaining minimal computational overhead, making them appropriate choices for latency-sensitive applications.

When the input sequence length was increased to 168 hours, representing an entire week of temporal context, the models exhibited an improved capacity to learn long-term periodic patterns. This adjustment particularly benefited deep learning and attention-based architectures. TimesNet delivered the highest forecasting accuracy under this setting, achieving the lowest MAE of 0.2089 and MSE of 1.0591. However, this performance came at the cost of significant computational demands, requiring over 9,296 seconds for training, which limits its practicality for real-time or resource-constrained applications.

In contrast, TSMixer emerged as a computationally efficient alternative. It achieved an MAE of 0.2855 and an MSE of 0.9726 with a runtime of only 57 seconds. This represents an approximately 160-fold improvement in speed compared to TimesNet, with only a modest reduction in forecast accuracy. These findings underscore the potential of lightweight architectures in operational environments where frequent retraining or rapid inference is essential.

Traditional statistical models such as SARIMAX performed poorly in this high-variance context. Despite their longstanding use in time-series analysis, SARIMAX recorded a relatively high MAE of 1.103 and MSE of 3.709, highlighting its limited ability to adapt to structural changes and regime shifts commonly

observed in Wikipedia traffic. ARIMAX performed better with a lower MAE of 0.791 and shorter runtime but still lagged more modern approaches in terms of accuracy. These outcomes underscore the limitations of classical models when applied to non-stationary, multi-scale, and user-driven time-series data, where deep learning and Transformer-based methods exhibit superior adaptability and forecasting performance.

In summary, the results obtained from the Wikipedia benchmark aligned with the patterns observed in the OTT dataset. Transformer-based models, especially TimesNet and FEDFormer, consistently achieved the highest forecasting accuracy. However, their time-consuming computational requirements highlight the need for careful consideration in practical deployments. Machine learning approaches such as GB and efficient architectures like TSMixer offer a decent trade-off between predictive performance and resource efficiency which makes them particularly suitable for production settings. These findings highlight the importance of selecting forecasting models based not solely on accuracy but also on operational restrictions like latency tolerance, scalability, and retraining frequency.

Table 10. Wikipedia Forecasting Results for seq_len = 24 and pred_len = 24

Model	MAE	MSE	Time (s)
ARIMAX	0.813	2.528	39
SARIMAX	0.801	2.573	2808
ES	0.907	3.016	0
RandomForest	0.318	1.147	201
GB	0.326	1.156	246
LightGBM	0.330	1.159	2
XGBoost	0.324	1.163	592
LSTM	0.346	1.204	1090
GRU	0.332	1.158	953
RNN	0.911	2.332	303
Autoformer	0.197	0.891	546
Crossformer	0.284	1.397	476
DLinear	0.379	1.736	22
FEDFormer	0.153	0.712	868
FiLM	0.359	1.724	253
Informer	0.299	1.373	254
iTransformer	0.238	1.319	101
LightTS	0.295	1.374	55
Non-stationary Transformer	0.233	1.313	212
PatchTST	0.298	1.416	76
Pyraformer	0.276	1.309	117
Reformer	0.314	1.448	155
TimesNet	0.209	1.209	2854
TimeXer	0.237	1.302	147
Transformer	0.296	1.410	188
TSMixer	0.291	1.202	57
WPMixer	0.309	1.432	54

Table 11. Wikipedia Forecasting Results for seq_len = 168 and pred_len = 24

Model	MAE	MSE	Time (s)
ARIMAX	0.709	2.435	36
SARIMAX	0.720	2.473	1101
ES	0.537	170.846	173
RandomForest	0.336	1.193	1283
GB	0.390	1.415	1556
LightGBM	0.326	1.159	10
XGBoost	0.324	1.168	36
LSTM	0.418	1.364	6087
GRU	0.401	1.260	1505
RNN	1.003	2.596	1673
Autoformer	0.380	1.164	1069
Crossformer	0.276	1.374	554
DLinear	0.346	1.543	29
FEDFormer	0.279	0.806	1661
FiLM	0.352	1.577	542
Informer	0.334	1.455	301
iTransformer	0.265	1.236	106
LightTS	0.323	1.418	48
Non-stationary Transformer	0.272	1.265	253
PatchTST	0.315	1.431	130
Pyraformer	0.277	1.306	298
Reformer	0.336	1.506	399
TimesNet	0.209	1.059	9296
TimeXer	0.279	1.338	138
Transformer	0.374	1.570	156
TSMixer	0.286	0.973	57
WPMixer	0.307	1.424	305

4.3.3 Weather dataset

The Weather dataset is a large-scale multivariate time series data that includes meteorological variables such as temperature, humidity, wind speed, and solar radiation. It also provides relatively stable and more seasonally consistent data than the more volatile OTT and Wikipedia datasets. Accordingly, this data is a useful real-world benchmark for measuring the performance of the models on highly structured data with few changes or outliers. Table 12 reveals all the forecasting models' results for the Weather dataset. Traditional statistical models performed well in both accuracy and computational efficiency. Exponential Smoothing (ES) delivered highly accurate forecasts with an MAE of 1.034 and MSE of 2.815, completing inference in 0 seconds, making it extremely efficient. ARIMAX yielded slightly better accuracy (MAE: 0.791, MSE: 2.121) than ES, with a modest runtime of 39 seconds. In contrast, SARIMAX showed the lowest accuracy among the three (MAE: 1.103, MSE: 3.709) and required the longest runtime at 3,692 seconds, which limits its practicality for large-scale or time-sensitive forecasting applications. These results suggest that while traditional models can remain competitive on stable datasets, their scalability and robustness may vary significantly depending on the model configuration and the nature of the input data.

Machine learning models such as Random Forest and LightGBM demonstrated relatively higher forecasting errors on the Weather dataset, with MAE values of 0.0474 and 0.0863, and MSE values of 0.0956 and 0.2058, respectively. However, these models benefited from notably shorter runtimes, ranging from 1 to 335 seconds. Deep learning models, including LSTM (MAE: 0.0670, MSE: 0.0128) and GRU (MAE: 0.1216, MSE: 0.0296), achieved modest improvements in accuracy

compared to machine learning approaches, but their training required significantly more time, with runtimes between 2,831 and 3,231 seconds.

Transformer-based models such as Autoformer (MAE: 0.2044, MSE: 0.1343) and FEDFormer (MAE: 0.1933, MSE: 0.1273) underperformed in this domain, revealing their limitations when applied to well-structured and seasonal data. This highlights the inefficiency of complex attention-based architectures in scenarios characterized by smooth, predictable temporal patterns.

Traditional statistical models were outstanding in terms of both performance and computational power needed. ES produced almost instantaneous results with minimal error. Deep learning and Transformer-based approaches such as TimesNet with runtimes exceeding 8,000 seconds. They are impractical for real-time deployment in this type of use cases was acknowledged with the consideration of marginal accuracy compared to computational expenses incurred.

Table 12. Weather Forecasting Results for seq_len = 24 and pred_len = 24

Model	MAE	MSE	Time (s)
ARIMAX	0.017	0.001	106
SARIMAX	0.020	0.001	1024
ES	0.015	0.001	0
RandomForest	0.047	0.096	335
GB	0.074	0.111	289
LightGBM	0.086	0.206	3
XGBoost	0.065	0.127	581
LSTM	0.067	0.013	3231
GRU	0.122	0.030	2831
RNN	0.159	0.061	875
Autoformer	0.204	0.134	152
Crossformer	0.141	0.094	182
DLinear	0.204	0.132	13
FEDFormer	0.193	0.127	194
FiLM	0.148	0.124	136
Informer	0.242	0.174	109
iTransformer	0.136	0.116	51
LightTS	0.175	0.119	22
Non-stationary Transformer	0.135	0.108	89
PatchTST	0.135	0.112	263
Pyraformer	0.205	0.132	41
Reformer	0.343	0.319	125
TimesNet	0.144	0.113	8223
TimeXer	0.152	0.124	455
Transformer	0.275	0.200	85
TSMixer	0.200	0.126	18
WPMixer	0.125	0.103	229

Overall, the results derived from the Weather dataset reinforce the continued relevance of classical time series models in structured and seasonally consistent settings. ES and ARIMAX provided outstanding forecasting accuracy with minimal computational cost, making them highly effective for such data types. Although machine learning and deep learning models offer greater flexibility, their increased complexity did not translate into meaningful improvements in this context.

These findings underscore the importance of selecting forecasting models based on the temporal and statistical nature of the data. Transformer-based architectures have shown excellent performance in handling highly variable and user-driven datasets such as OTT traffic and Wikipedia page views. However, in stationary and periodic environments like meteorology, simpler linear models retain a clear advantage in terms of both accuracy and efficiency.

CHAPTER 5

DISCUSSION

The results of this large-scale evaluation over the OTT, Wikipedia, and Weather datasets provide important insights into model selection, the necessity of computational necessity, and hyperparameter tuning in time-series forecasting. The following discussion goes further on these findings, addressing each research question and describing their impact on academic research and industry implementation.

RQ1: Which forecasting models including statistical, machine learning, deep learning models, or Transformer-based architectures are most effective in capturing the dynamic and non-linear nature of OTT user behavior for hourly usage prediction?

Transformer-based models, especially FEDFormer, TimesNet, and PatchTST demonstrated the most accurate predictions in the OTT dataset. These models outperformed statistical, machine learning, and deep learning approaches. FEDFormer achieved the lowest MAE on the 24-hour configuration of the OTT dataset. Their success can be attributed to advanced mechanisms like self-attention and multi-scale processing. In contrast, classical statistical methods like SARIMAX and ES struggled with OTT data. This performance gap highlights the fundamental mismatch between the linear assumptions of classical approaches and the highly dynamic nature of user-driven traffic patterns. Machine learning alternatives showed more adaptability achieving MAEs around 0.227. However, their predictions lacked the granularity to capture peak times. Deep learning approaches showed intermediate performance while their performance is better than statistical methods but outperformed by machine learning and Transformer-based methods. This suggests

that deep learning approaches can model temporal dependencies but the self-attention mechanisms of Transformer-based methods provide superior capacity for handling complex patterns.

RQ2: How does hyperparameter tuning impact the performance of forecasting models in predicting OTT platform data?

Hyperparameter tuning revealed that statistical models like ARIMAX and SARIMAX are highly sensitive to parameter configurations. For ARIMAX, the optimal setup was (p = 0, d = 0, q = 0), which achieved an MAE of 0.604. Introducing differencing (d = 1) or complex moving average terms led to significant performance drops, with MAE values increasing to 0.978 in some configurations, indicating that simpler ARIMAX variants perform better in this context. SARIMAX exhibited extreme variability. While the best configuration (p = 0, d = 0, q = 0) × (P = 0, D = 1, Q = 1) achieved an impressive MAE of 0.233, other setups resulted in extremely poor performance with MAE values like 4.0. This underlines SARIMAX's potential when tuned correctly, but also its vulnerability to misconfiguration. The machine learning models exhibited more robust performance across parameter variations, though tuning still provided meaningful improvements. The tuning process of Random Forest revealed that increasing the number of estimators beyond 100 provided diminishing returns while deeper trees (max_depth = 20) slightly improved accuracy. GB model exhibited consistent improvement with deeper trees and more iterations. The configuration (max_depth = 3, learning_rate = 0.1, n_estimators = 300) reached the lowest MAE of 0.152 and MSE of 0.420 in 55.47s, a substantial drop from the 0.361 MAE obtained with just 100 estimators and learning rate 0.01. These results highlight how finely tuned configurations offer sharp accuracy gains. XGBoost behaved similarly to GB and benefited from tree depth and

learning rate optimizations. The optimal configuration reached an MAE of 0.150 and MSE of 0.411. The tuning experiments revealed that LightGBM's performance was less sensitive to the number of leaves beyond 31 which suggests that as a good default value for similar forecasting tasks. For deep learning models, the LSTM tuning process demonstrated the critical importance of learning rate selection. While larger networks like 200 units could achieve slightly better accuracy (MAE: 0.210) and the optimal configuration (50 units, learning_rate = 0.001, batch_size = 64) provided the best prediction accuracy (MAE: 0.180). The experiments clearly showed that learning rates above 0.01 led to significant performance degradation, with MAE values increasing by 150% in some cases.

RQ3: How do statistical, machine learning, and deep learning forecasting models perform across datasets with different characteristics such as seasonality, and trend complexity?

Classical forecasting methods such as ARIMAX, SARIMAX, and ES demonstrated strong performance in more structural environments. ARIMAX and ES achieved highly accurate results in the Weather dataset which exhibits high regularity and seasonality with MAEs as low as 0.0147 and MSEs below 0.0006. Their success is due in part to structural simplicity, allowing efficient modeling of consistent temporal trends with low variance. However, their effectiveness declined considerably in high-variance, user-driven contexts such as OTT and Wikipedia traffic.

Machine learning models which include GB, Random Forest, LightGBM, and XGBoost offered a practical middle ground between traditional statistical approaches and deep learning architectures. GB and XGBoost consistently achieved competitive performance on the OTT dataset while LightGBM provided accurate results while

maintaining extremely low training times, often under one second. These models generalized well to the Wikipedia dataset. For example, GB achieved an MAE of 0.2270 under the 24-hour configuration. Although they did not consistently achieve the lowest error rates, their ability to provide stable and efficient performance across various datasets makes them well-suited for real-time inference and large-scale operational deployments. Their results on the Weather dataset demonstrate their ability to model smooth and changeable data alike, at times being just under as accurate as classical models in highly seasonal environments at smoothing.

Deep learning models with recurrent neural network architectures, like LSTM and GRU, showed vast improvements over the existing models, especially in more variable environments as seen in the OTT and Wikipedia data. However, these architectures typically did not surpass tree-based models in terms of forecasting accuracy and required significantly longer training times. For instance, LSTM achieved an MAE of only 0.0669 on the Weather dataset. This was still worse than ES. On the OTT and Wikipedia datasets, LSTM and GRU models achieved acceptable results but ranged from several hundred to over a thousand seconds. This indicates that although RNN-based models might learn sequential dependencies better, they have less favorable efficiency and performance trade-offs in cases where a simpler and much faster alternative is possible.

Transformer-based models consistently demonstrated the highest forecasting accuracy for all three datasets. FEDFormer, TimesNet, iTransformer, and PatchTST consistently ranked highest or top across all the configurations. FEDFormer achieved the lowest MAE on the 24-hour configuration of the OTT dataset. TimesNet particularly performed in long-sequence scenarios on both the OTT and Wikipedia benchmarks.

In the Wikipedia dataset, TimesNet and FEDFormer established novel benchmarks in both accuracy and robustness. These models effectively handled irregular user behavior and periodic revisit patterns through the application of attention mechanisms and Fourier-based representations. Their computational needs were massive. TimesNet requires over 9,000 seconds to train with a weekly input configuration, demonstrating major scalability issues for production settings.

Recent architectures such as TSMixer and WPMixer have performed even better as lightweight alternatives. These models are still effectively modeled for temporal dependencies with a considerable decrease in the time taken for training. For example, TSMixer achieved a 160-times improvement in runtime compared to TimesNet on the Wikipedia dataset with only a marginal increase in forecasting error. These findings indicate maturity in the field from a research perspective and an increasing focus on creating efficient Transformer variants that perform optimally in resource-constrained environments.

In Shelatkar et al. (2020), a hybrid model consisting of ARIMA and LSTM was tested on web traffic forecasting using Wikipedia data. Their research emphasized the importance of breaking down signals into linear and non-linear components and found that the combination of ARIMA with an LSTM layer offered improved accuracy compared to either model individually. The findings presented in this thesis support the observed trend, ARIMA and LSTM provided moderate accuracy, but were consistently outperformed by Transformer-based architectures such as PatchTST, TSMixer, and iTransformer, particularly on volatile datasets like OTT and Wikipedia. This suggests that while hybrid models can enhance traditional methods, Transformer-based models are more scalable and accurate in capturing both short- and long-range dependencies. MV et al. (2024) employed a multi-model

ensemble to forecast Wikipedia traffic. Their top-performing models combined SARIMA with LSTM but required manual decomposition and ensemble stacking. In contrast, the benchmark results in this thesis indicate that Transformer-based architectures not only achieve better accuracy but also offer a more modular and efficient alternative. Studies such as Nunnagoppula et al. (2023) and Casado-Vara et al. (2021) focused on short-term traffic forecasting and video-on-demand (VoD) consumption trends. In both cases, LSTM and CNNs were noted for their ability to adapt to temporal shifts, especially under high variance and fluctuated user behavior. The results presented in this thesis reveal that while LSTM continues to be a reliable baseline, it was consistently outperformed by PatchTST and TimesNet, where the complexity of user behavior demands more robust representations of temporal structure like on OTT and Wikipedia datasets. Similarly, Nogueira et al. (2017) applied regression-based techniques to OTT catch-up TV traffic and emphasized runtime efficiency. The runtime evaluations included in this thesis match this concern and extend the analysis by benchmarking Transformer models alongside statistical and machine learning models. Notably, XGBoost and LightGBM remained the best choices for low-latency and resource-limited environments while Transformer models like PatchTST and TSMixer offered the best accuracy with runtime trade-off where predictive performance is crucial.

One of the key insights from this study is the consistent superiority of Transformer-based models across highly variables like OTT and Wikipedia datasets and more structured such as Weather datasets. These results extend the conclusions from the findings from Jiang et al. (2021) and Bi et al. (2021) which showed that Transformers like Autoformer and Informer can dominate at long-sequence forecasting and generalization across tasks. In the present benchmarks, PatchTST,

TSMixer, and TimesNet consistently achieved top-tier performance with minimal tuning while offering a unified solution across diverse domains.

Furthermore, experimental results align with the findings of Kougioumtzidis et al. (2025) that models exploiting multi-scale temporal decomposition with Non-Stationary Transformer. Also, their results provide meaningful improvements for datasets with seasonal variation, such as Weather. In our evaluation, these models ranked closely behind PatchTST and TimesNet, especially in MSE, suggesting that multi-scale modeling continues to be a valuable architectural principle in Transformer forecasting.

Overall, Transformer-based models have set a new state-of-the-art in forecasting accuracy, especially on complicated and highly variable datasets. Due to their substantial resource demands, it is essential to persist in investigating methods to optimize model design, including model compression, reduction techniques because of their substantial resource demands, it is essential to persist in investigating methods to optimize model design, including model compression, and reduction techniques. Ultimately, the choice of the best forecasting model is still case-by-case and should be matched to the data characteristics, desired use, and deployment environment availability constraints.

CHAPTER 6

CONCLUSION

This study evaluates the applicability of various time-series forecasting models and approaches including statistical methods, machine learning techniques, deep learning methods, and Transformer-based infrastructure to predict hourly usage patterns of OTT platforms with a specific focus on complex, non-stationary, and volatile behavior of end users. On the other hand, these methods are evaluated with publicly available datasets like Wikipedia Page Views and Weather alongside the OTT dataset. Through this evaluation, a comprehensive empirical basis was established to support model selection in contemporary time-series forecasting tasks.

The results indicated that forecasting accuracy is highly dependent on the characteristics of the underlying dataset. Models that produce accurate predictions on smooth, structured, and frequently seasonal data tend to lose effectiveness when applied to non-linear, disorganized, and irregular user-driven data. On the other hand, models that perform well in dynamic and high-variance settings could perform well in periodic and low-noise settings as well. These findings support the conclusion that there is no single forecasting model that can be optimal in every situation. The selection of a model should be driven by the nature of complexity, temporal dynamics, and applicability of operational requirements.

Among the approaches evaluated, Transformer-based models demonstrated superior forecasting accuracy, particularly when assessed on the OTT and Wikipedia datasets. Models such as FEDFormer, TimesNet, and iTransformer were found to be highly effective in learning long-term dependencies, accommodating non-stationary patterns, and capturing complex temporal structures. These models consistently

outperformed both classical statistical methods and deep recurrent architectures based on metrics such as MAE and MSE. However, this superior performance was associated with significant computational demands, including extended training durations and sensitivity to hyperparameter settings. For instance, TimesNet achieved state-of-the-art results on the Wikipedia dataset, but its training time exceeded 9,000 seconds which limits its applicability in time-sensitive or resource-limited environments.

Tree-based ensemble models such as GB and XGBoost were shown to provide an ideal balance between predictive performance and computational efficiency. These models achieved competitive accuracy while requiring significantly less training time. These models reached accuracy competitively with larger state-of-the-art models while needing orders of magnitude less training time. LightGBM is specifically designed for rapid predictions and is optimally utilized in applications necessitating real-time processing on hardware with limited computational resources. These strong generalization capabilities were seen on all datasets, adding to the confidence in the reliability of these methods even when deployed at a large scale.

Traditional forecasting models like ARIMAX, SARIMAX, and ES proved to be extremely effective, especially when used on the Weather dataset. These approaches found success in modeling seasonality and produced consistently stable, low-error forecasts at lower computational costs. In scenarios where clear temporal trends are the norm, the increased complexity involved in using deep learning or Transformer models did not result in substantive gains and in some instances produced inefficiencies.

The study emphasized the growing significance of lightweight Transformer-inspired products such as TSMixer and WPMixer. These simplified models achieved

forecasting accuracy comparable to comprehensive Transformer topologies but incurred significantly lower computational costs. For example, TSMixer required only a fraction of the training duration compared to TimesNet on the wiki dataset, with only a negligible compromise on accuracy. Models of this nature are particularly adept for implementation in operational settings that necessitate daily retraining or deployment at the edge.

A standardized evaluation approach was employed to facilitate comparability across various experiments. Standardized input lengths, prediction intervals, and assessment metrics (MAE, MSE, and runtime) facilitated reproducible and equitable comparisons among model types and within a dataset. The consistency of application enabled valid conclusions regarding the models' behavior and overall applicability.

While this study provides numerous contributions to the field by applying and comparing a wide range of forecasting models on OTT data, certain limitations should be acknowledged. First, although the used OTT dataset is proprietary and reflects real user behavior, the platform data represents only Turkish people with specific demographic segment. Its availability limits external reproducibility and broader generalization. Second, the hyperparameter space was not exhaustive for all model classes while a quite extensive parameter set was used. This situation is possibly leaving some performance gains unexplored.

The findings of this study indicate several potential directions for future investigation. Hybrid models that integrate the interpretability of tree-based methods with the temporal modeling capabilities of Transformer topologies may provide a means to achieve both accuracy and efficiency. Additionally, further efforts aimed at optimizing the computational efficiency of Transformer models through architectural

simplification, adaptive attention mechanisms, or dynamic pruning techniques may increase their accessibility for broader use beyond large-scale cloud infrastructure.

In summary, this research offers a practical methodology for model selection in time-series forecasting based on application-specific requirements, dataset characteristics, and operational limitations. While Transformer models clearly dominate in accuracy for forecasting, classical and tree-based models continue to offer distinct advantages in efficiency, simplicity, and deployment. Intentional decision-making that acknowledges these trade-offs will allow researchers and practitioners to make model choices in the service of technical and operational goals.

APPENDIX A

OTT FORECASTING SAMPLE RESULTS

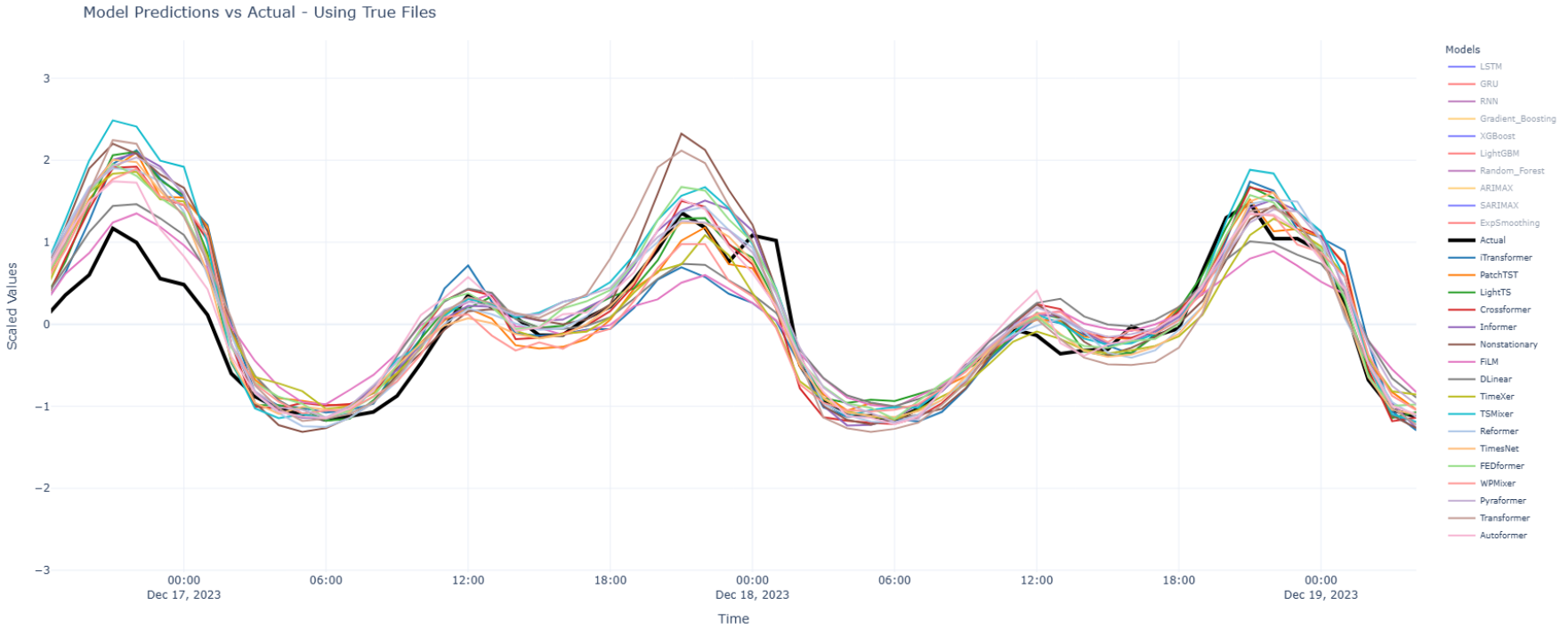

Figure A1. Transformer-based models forecasting results between Dec 17, 2023 and Dec 19, 2023 for seq_len = 24 and pred_len = 12

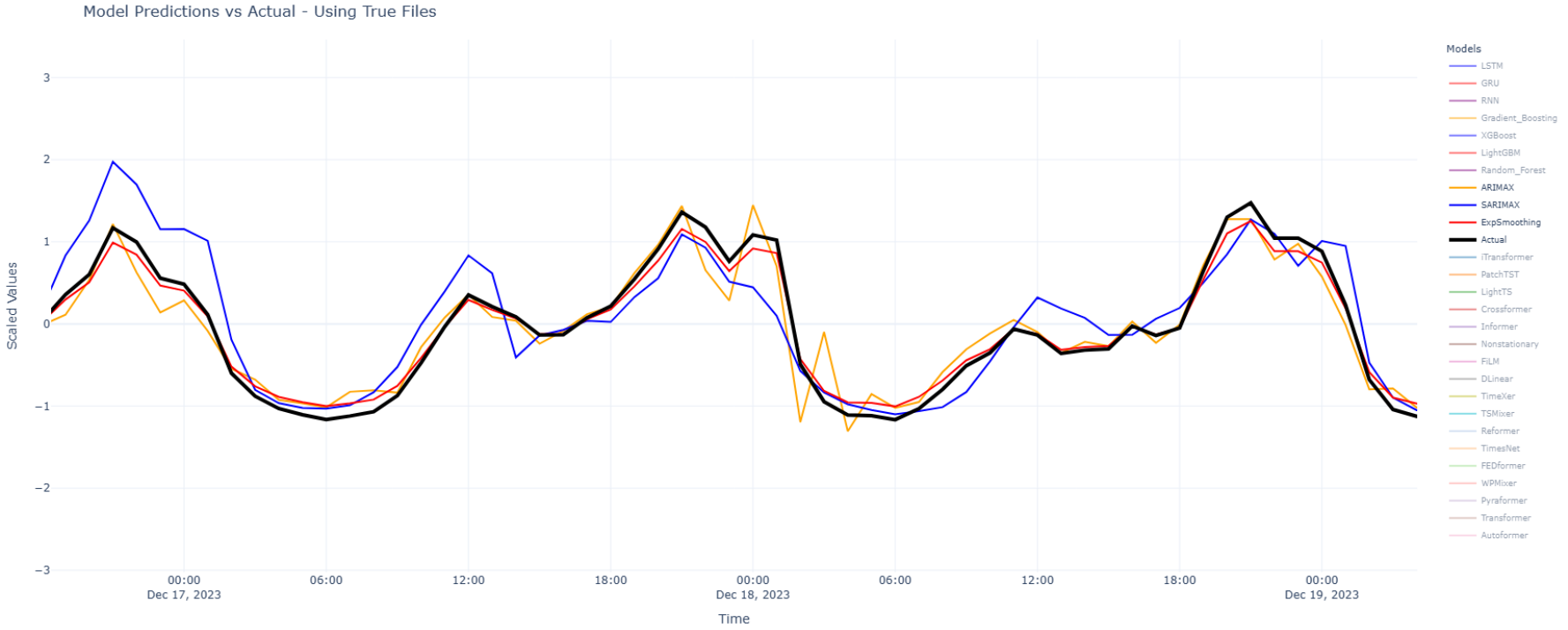

Figure A2. Statistical methods forecasting results between Dec 17, 2023 and Dec 19, 2023 for seq_len = 24 and pred_len = 12

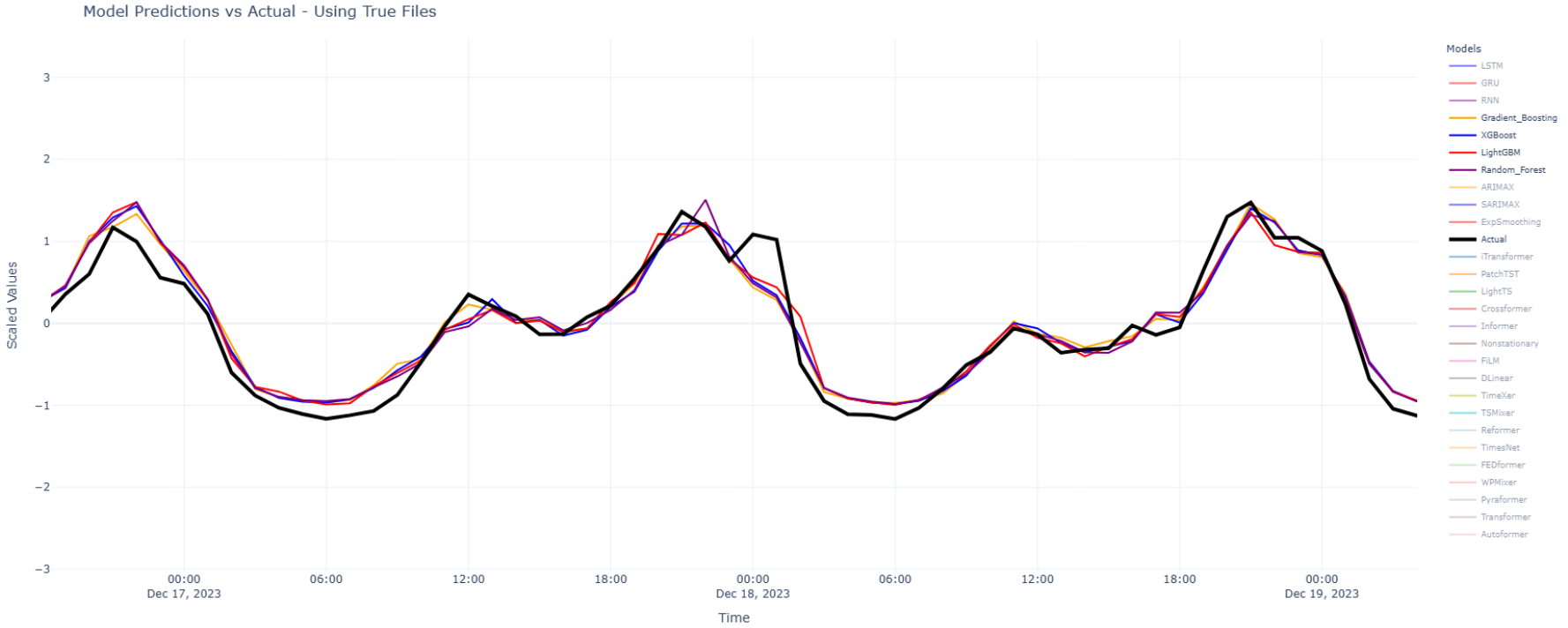

Figure A3. Machine learning methods forecasting results between Dec 17, 2023 and Dec 19, 2023 for seq_len = 24 and pred_len = 12

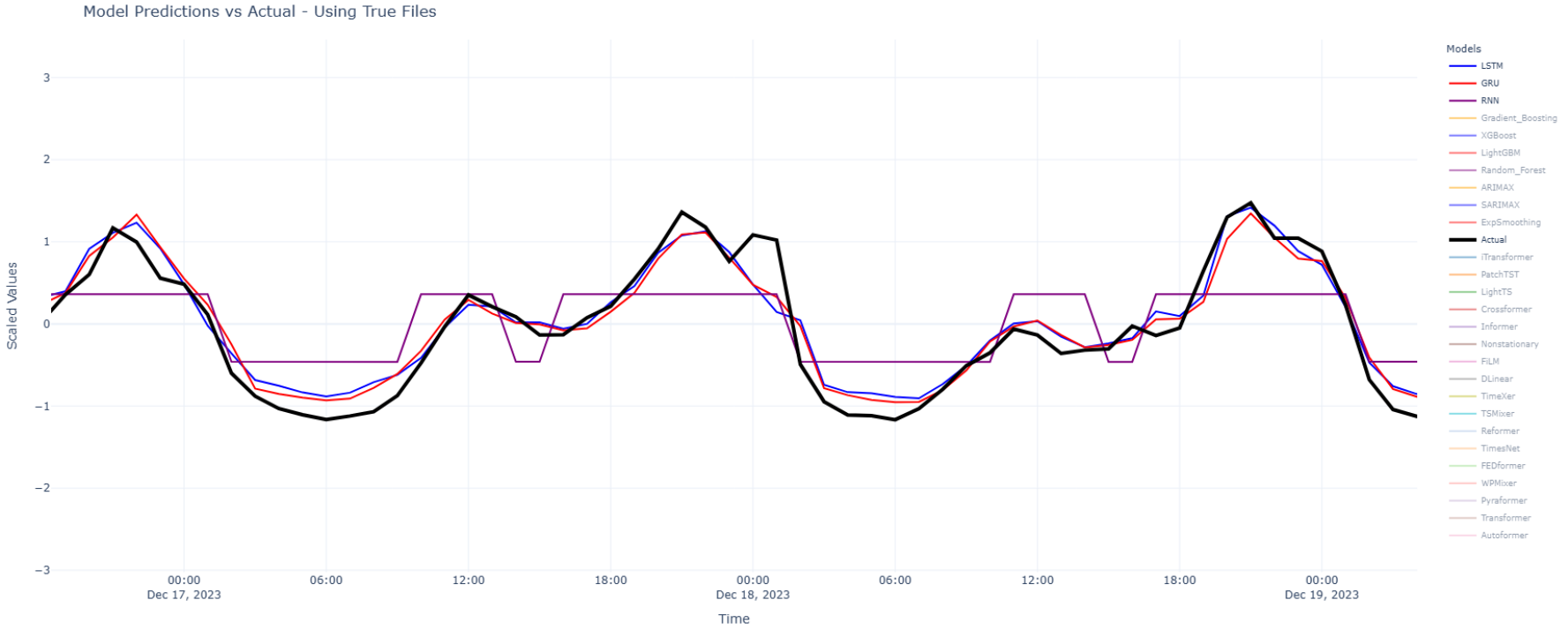

Figure A4. Deep learning methods forecasting results between Dec 17, 2023 and Dec 19, 2023 for seq_len = 24 and pred_len = 12

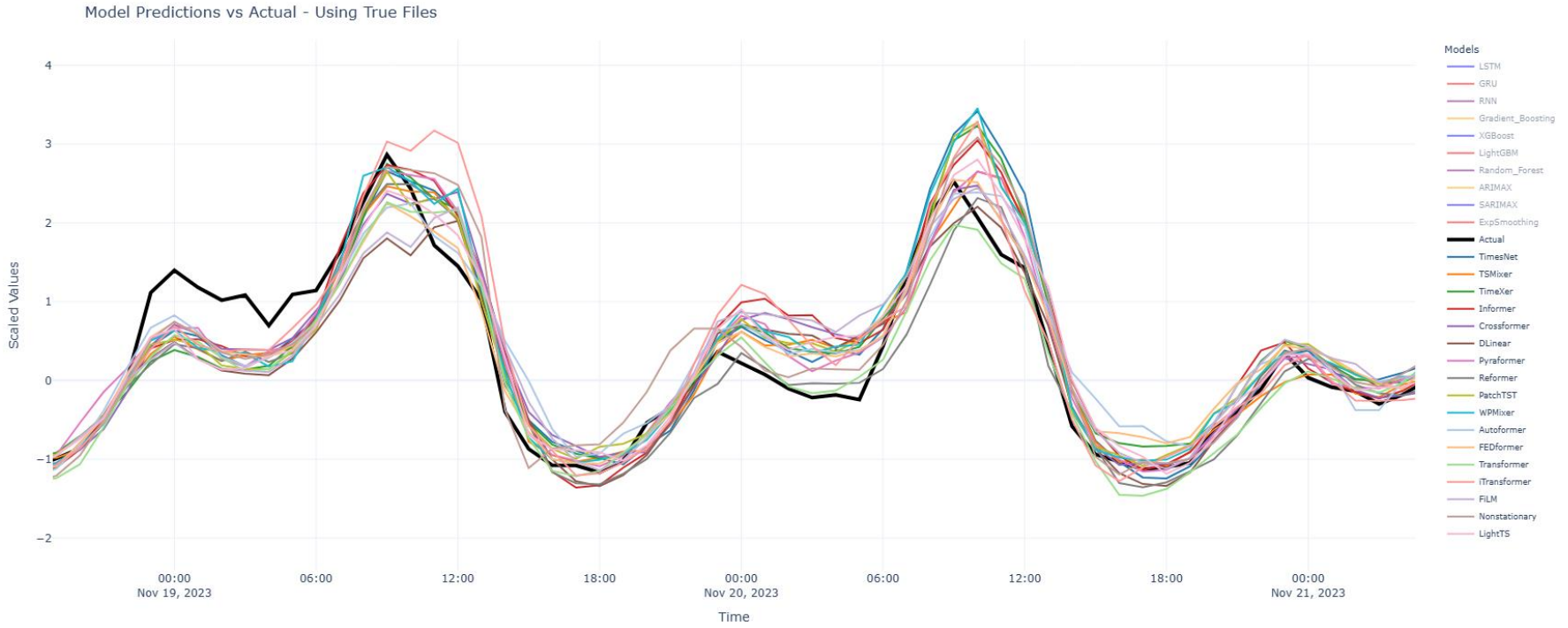

Figure A5. Transformer-based models forecasting results between Nov 19, 2023 and Nov 21, 2023 for seq_len = 24 and pred_len = 24

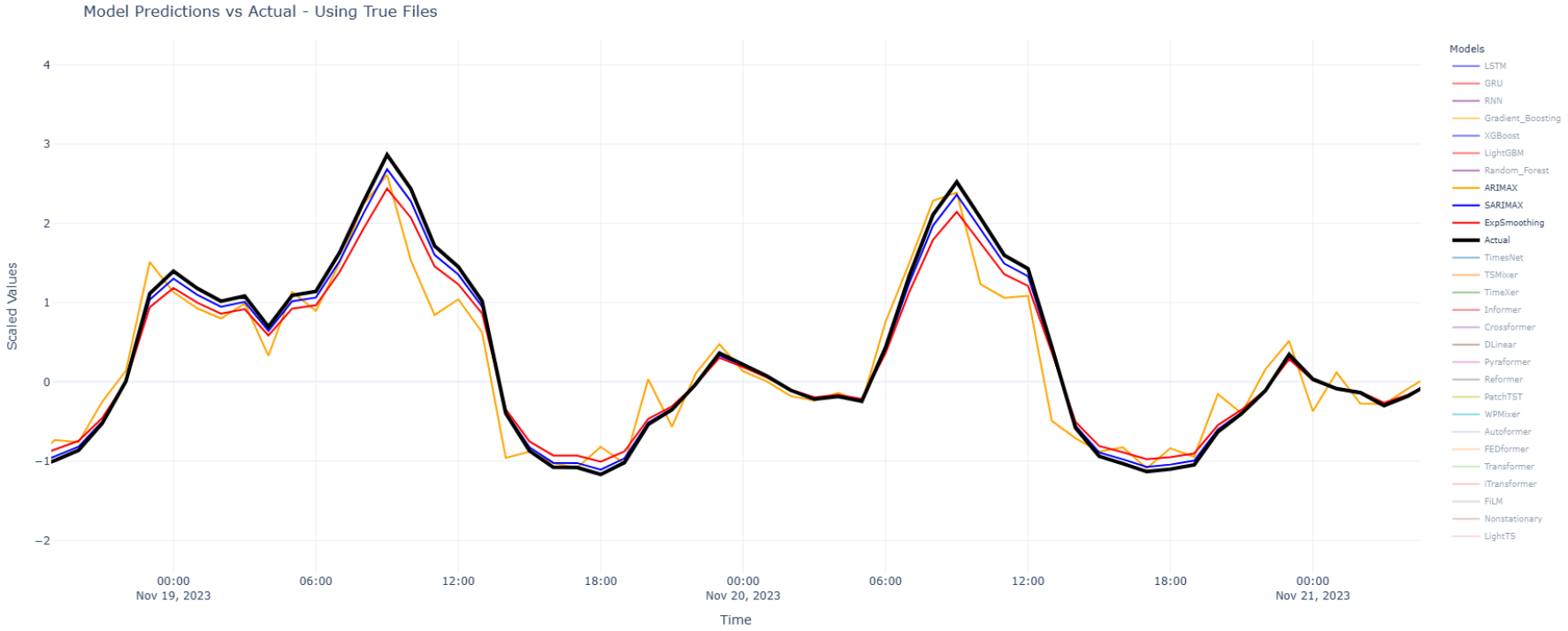

Figure A6. Statistical methods forecasting results between Nov 19, 2023 and Nov 21, 2023 for seq_len = 24 and pred_len = 24

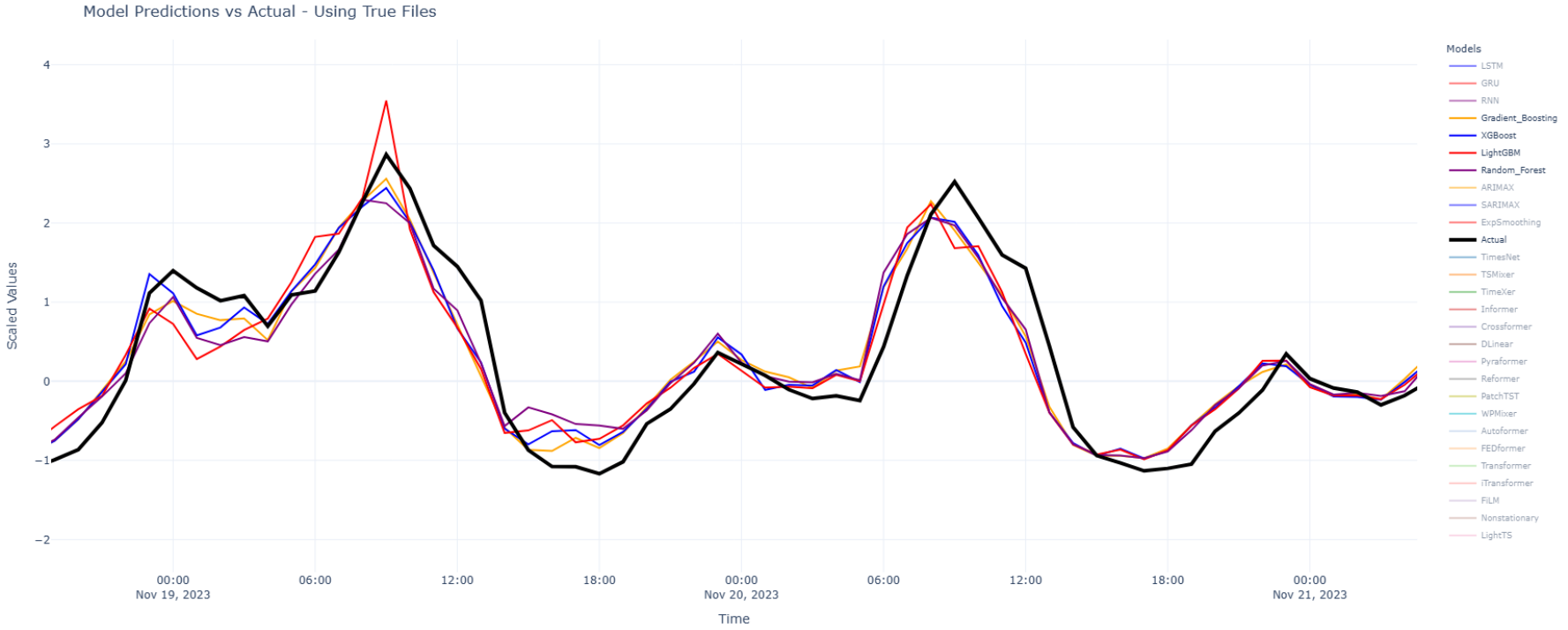

Figure A7. Machine learning methods forecasting results between Nov 19, 2023 and Nov 21, 2023 for seq_len = 24 and pred_len = 24

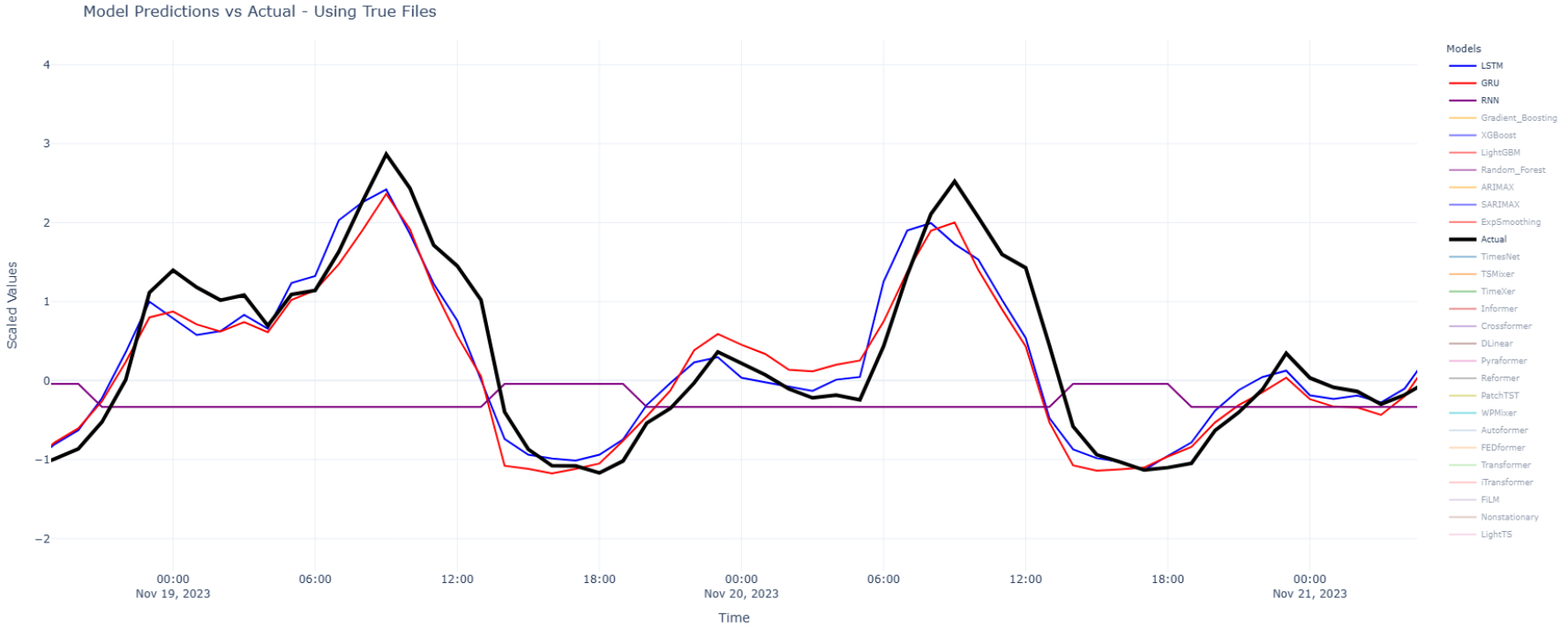

Figure A8. Deep learning methods forecasting results of between Nov 19, 2023 and Nov 21, 2023 for seq_len = 24 and pred_len = 24

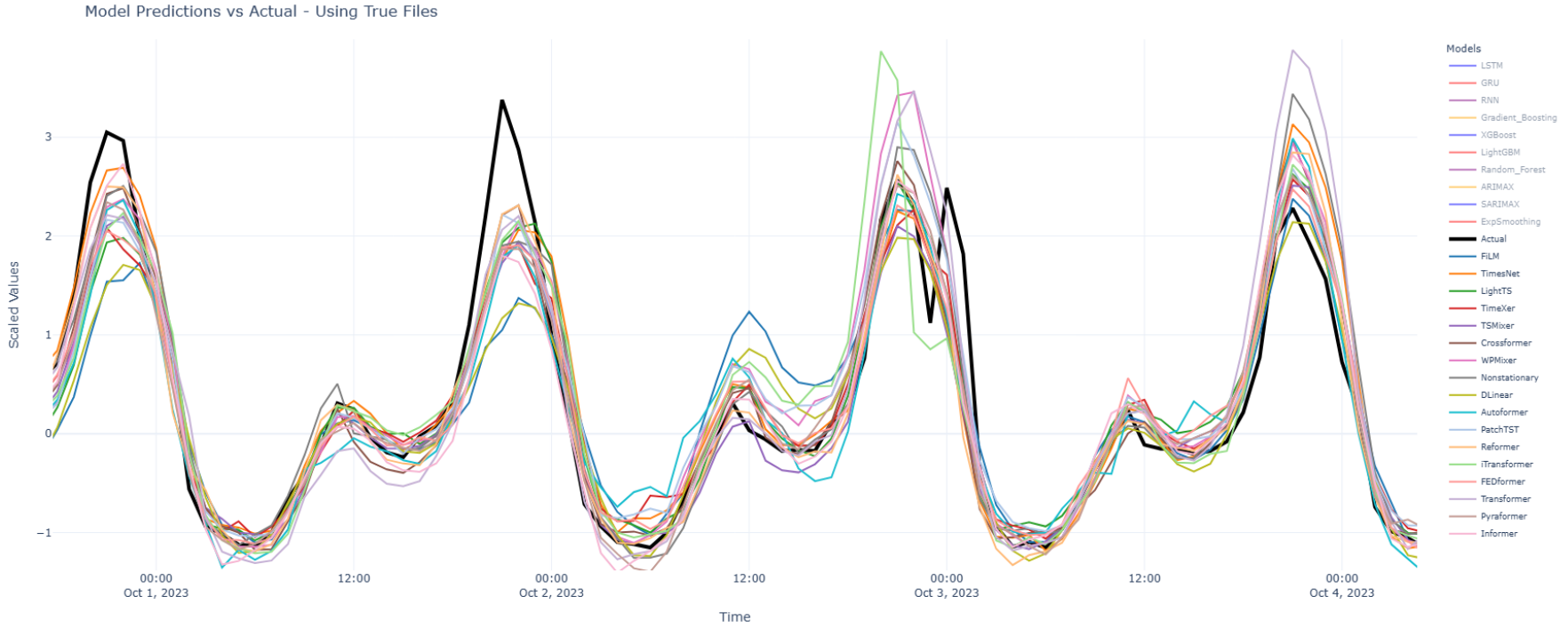

Figure A 9. Transformer-based models forecasting results of Oct 1, 2023 and Oct 4, 2023 for seq_len = 24 and pred_len = 36

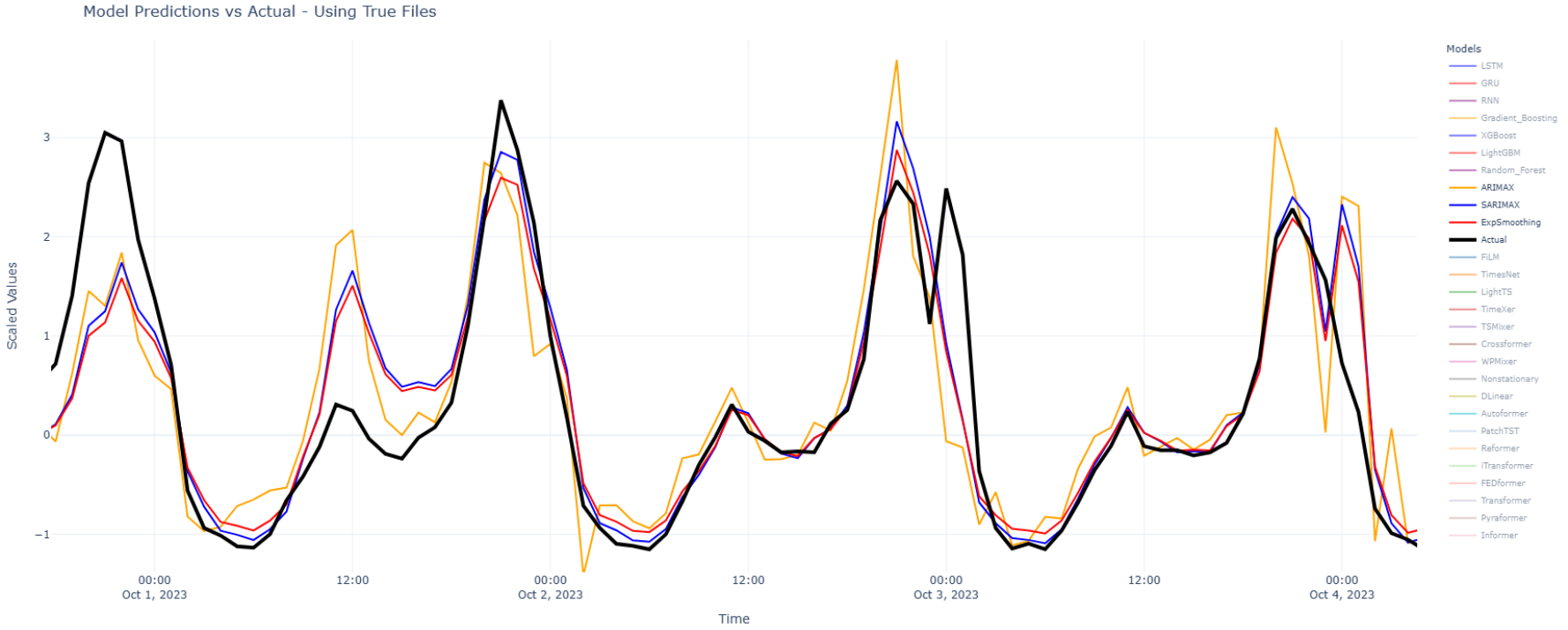

Figure A10. Statistical methods forecasting results between Oct 1, 2023 and Oct 4, 2023 for seq_len = 24 and pred_len = 36

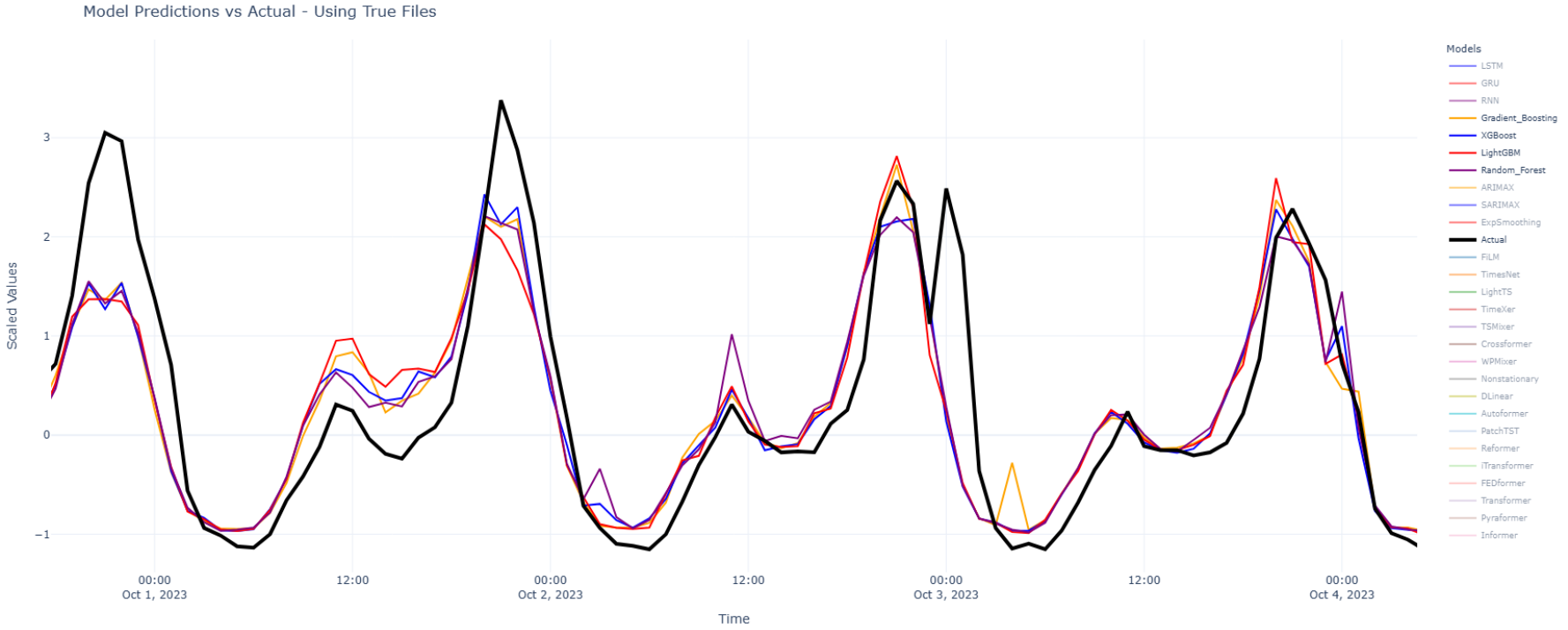

Figure A11. Machine learning methods forecasting results between Oct 1, 2023 and Oct 4, 2023 for seq_len = 24 and pred_len = 36

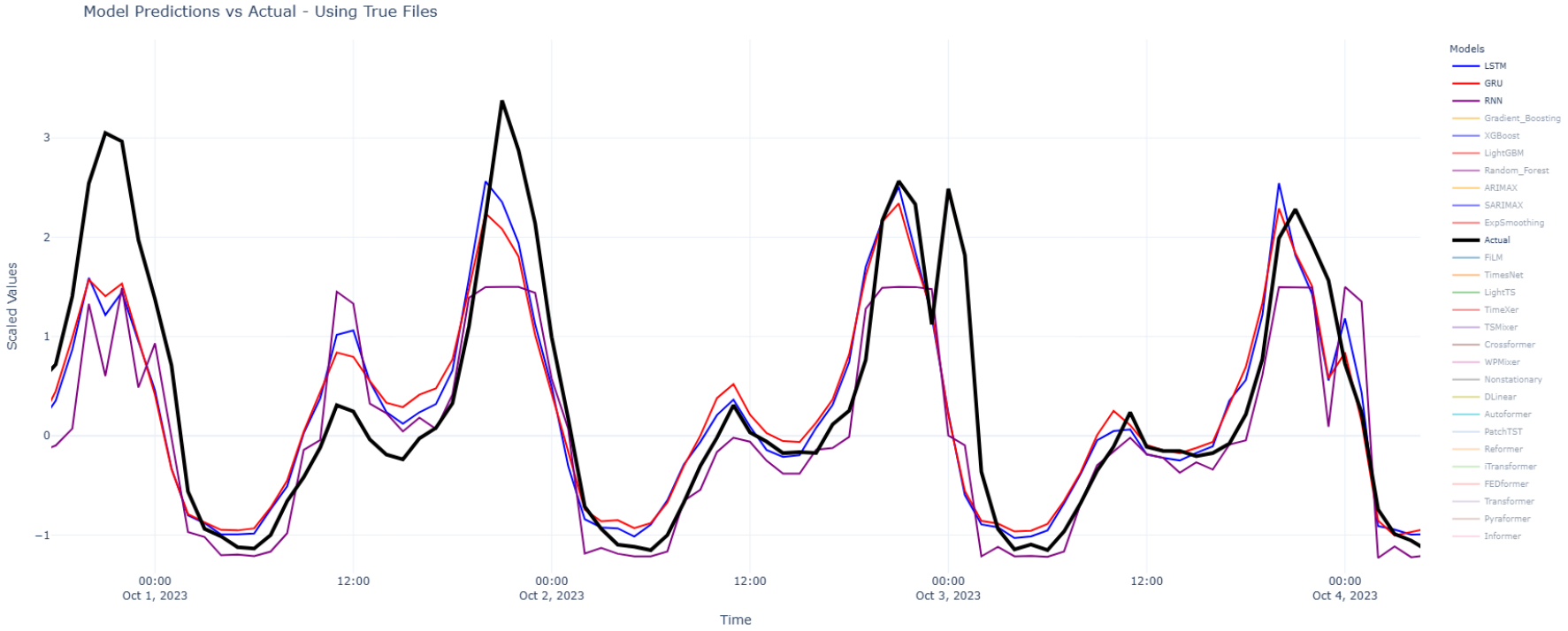

Figure A12. Deep learning methods forecasting results between Oct 1, 2023 and Oct 4, 2023 for seq_len = 24 and pred_len = 36

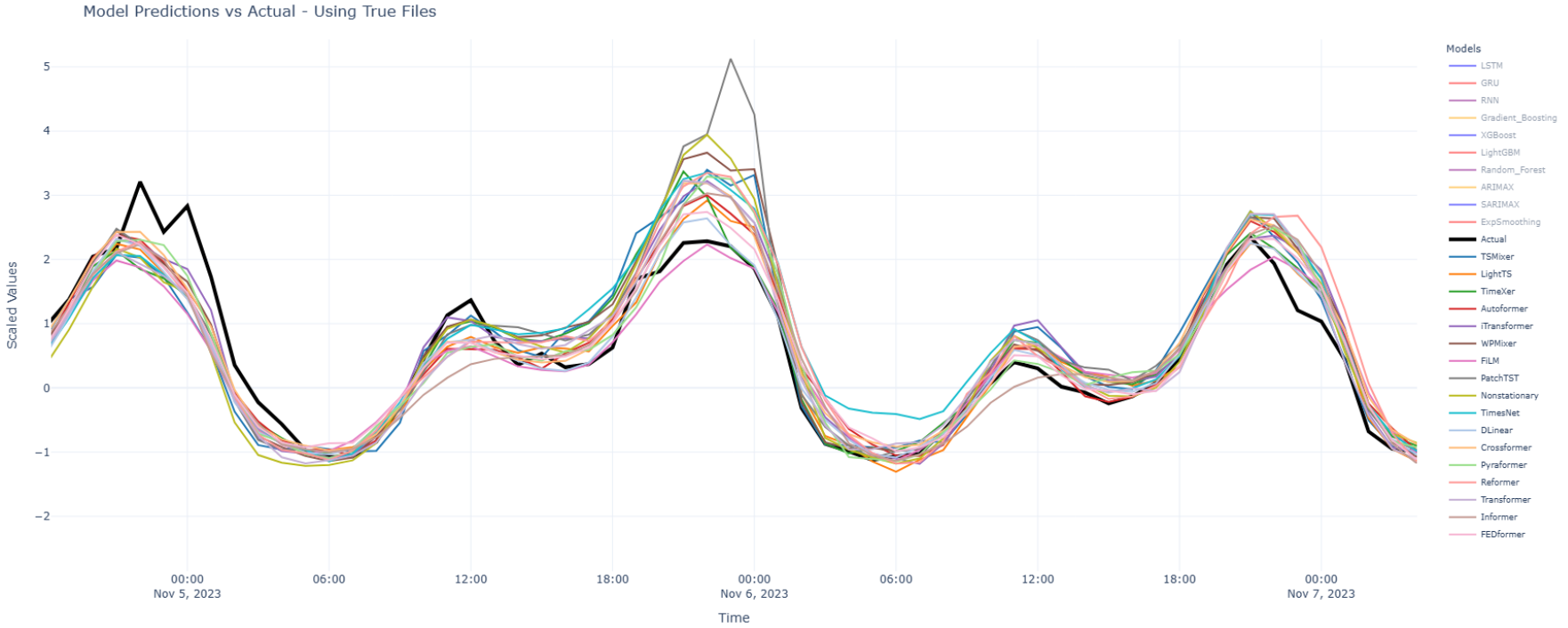

Figure A13. Transformer-based models forecasting results between Nov 5, 2023 and Nov 7, 2023 for seq_len = 168 and pred_len = 12

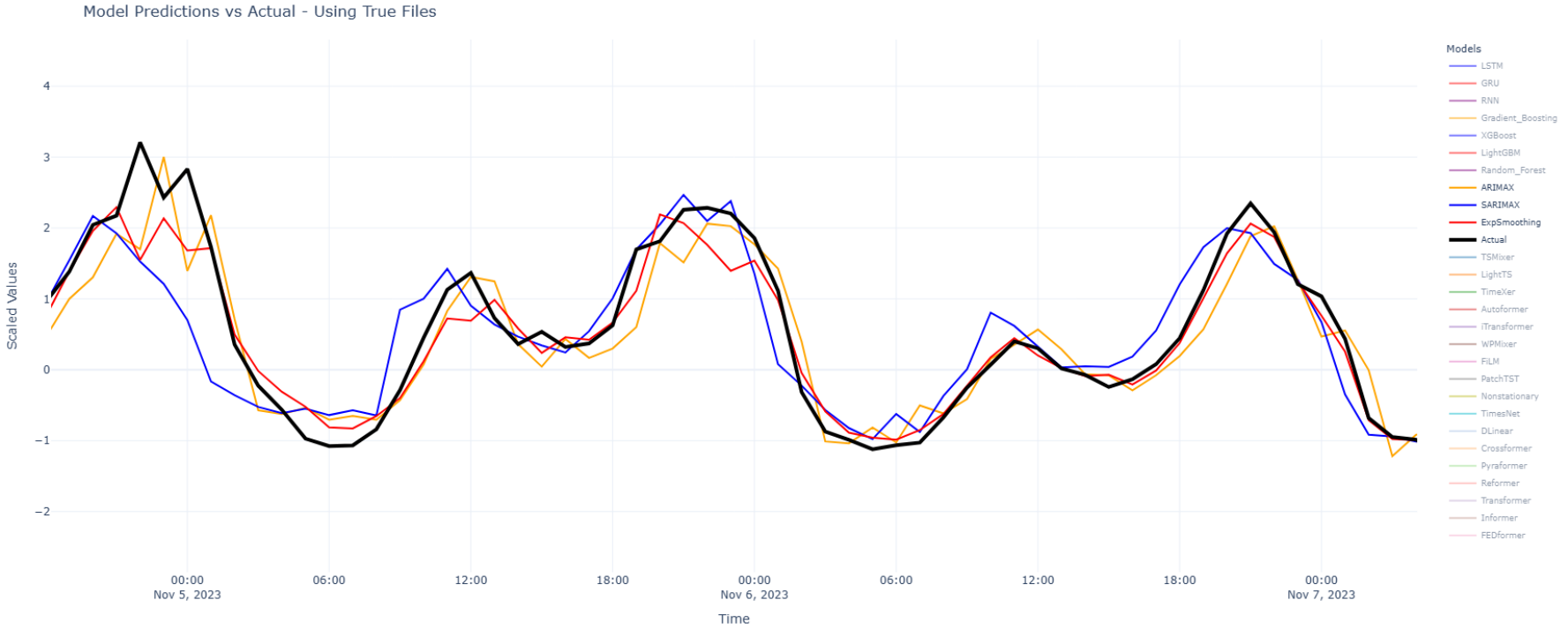

Figure A14. Statistical methods forecasting results between Nov 5, 2023 and Nov 7, 2023 for seq_len = 168 and pred_len = 12

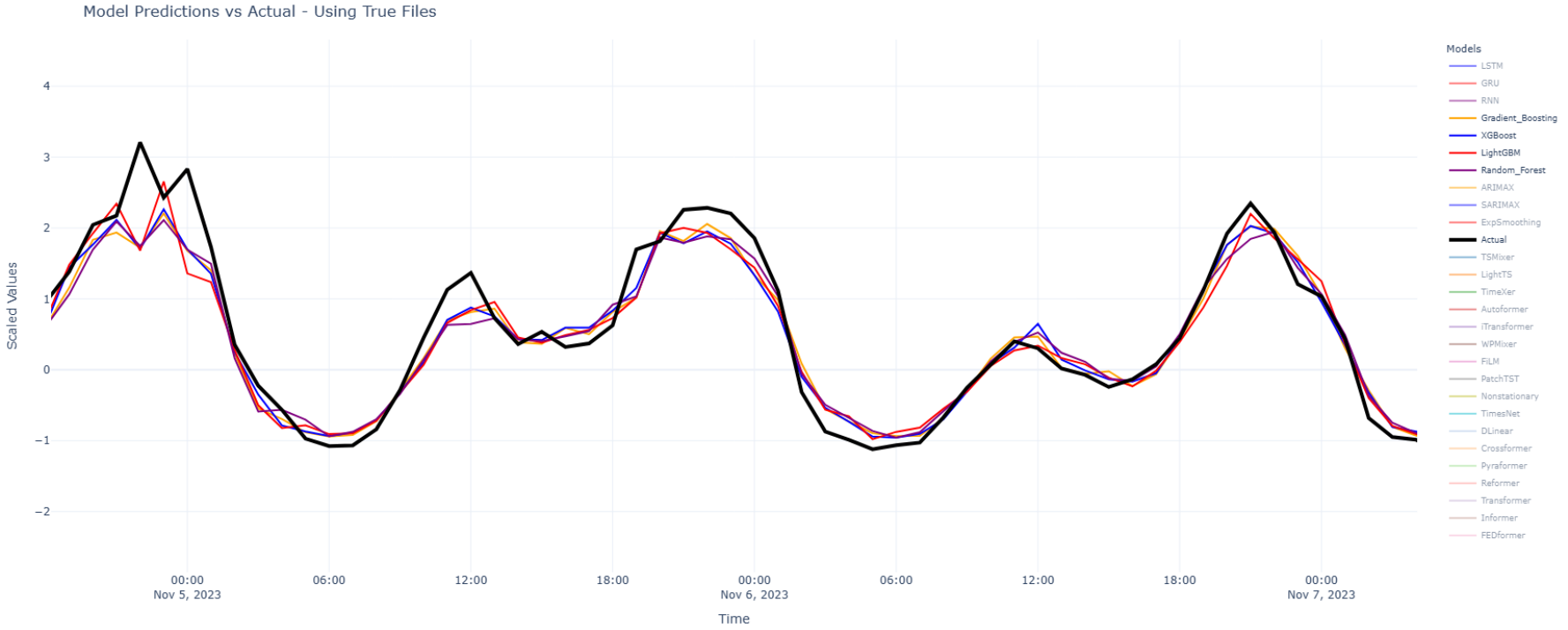

Figure A15. Machine learning methods forecasting results between Nov 5, 2023 and Nov 7, 2023 for seq_len = 168 and pred_len = 12

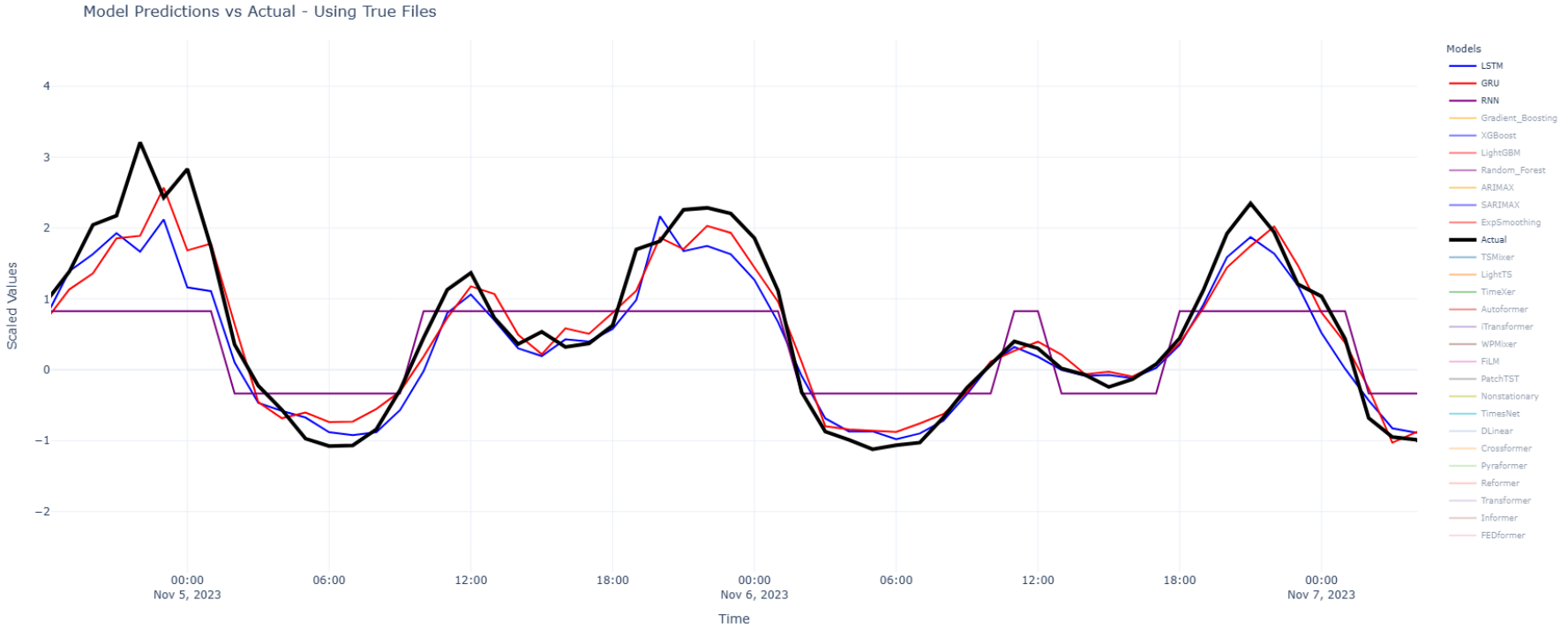

Figure A16. Deep learning methods forecasting results between Nov 5, 2023 and Nov 7, 2023 for seq_len = 168 and pred_len = 12

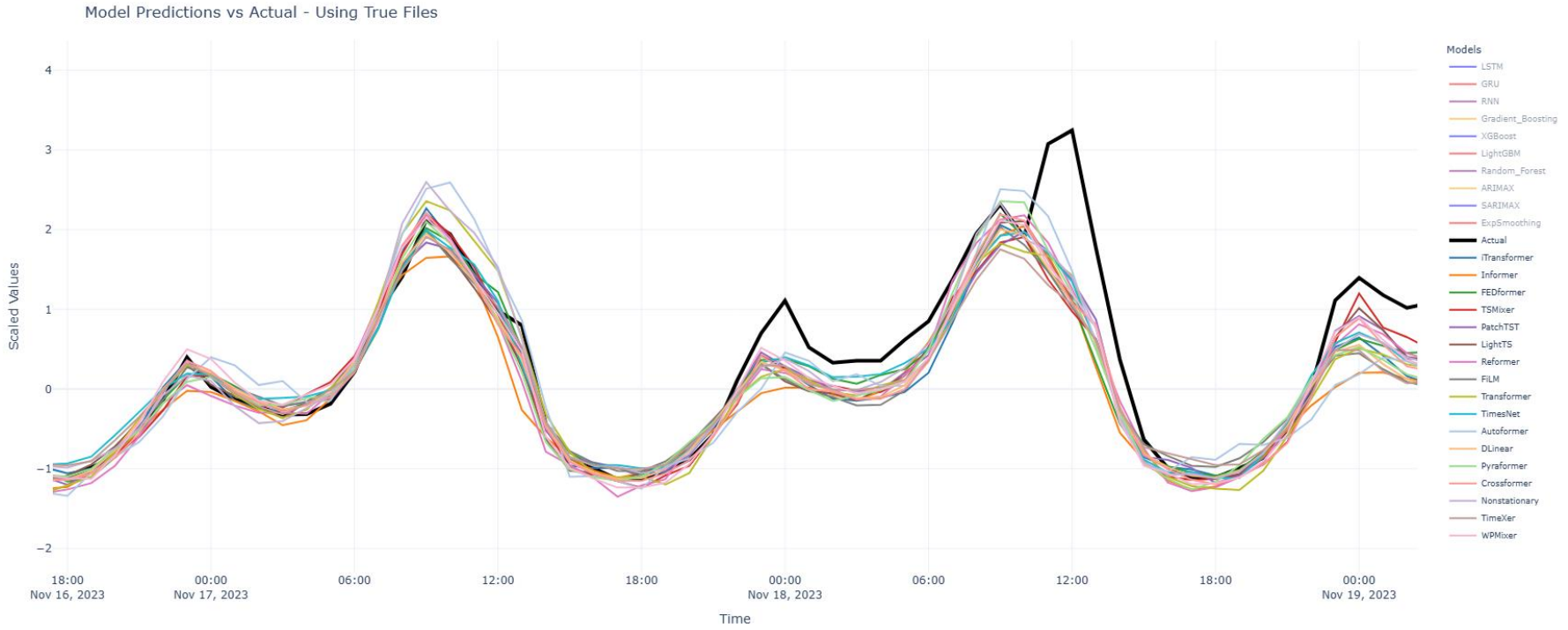

Figure A17. Transformer-based models forecasting results between Nov 17, 2023 and Nov 19, 2023 for seq_len = 168 and pred_len = 24

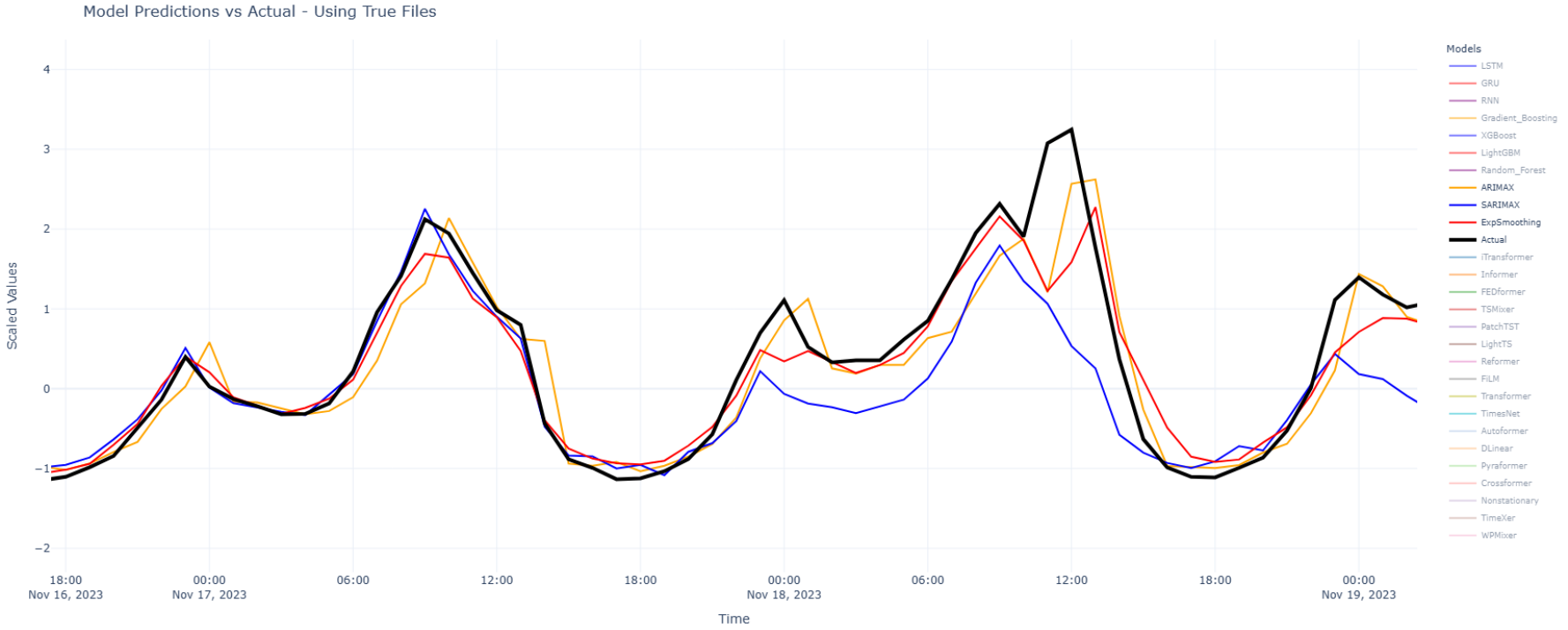

Figure A18. Statistical methods forecasting results between Nov 17, 2023 and Nov 19, 2023 for seq_len = 168 and pred_len = 24

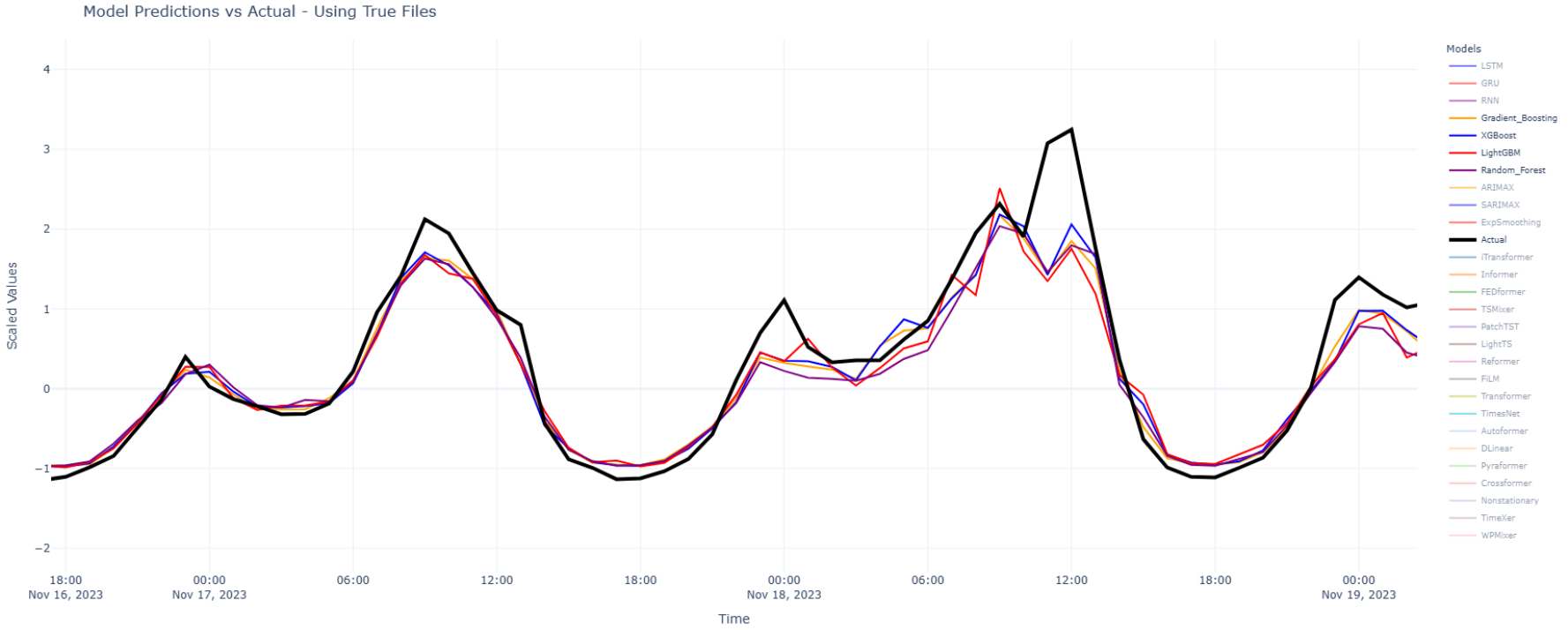

Figure A19. Machine learning methods forecasting results between Nov 17, 2023 and Nov 19, 2023 for seq_len = 168 and pred_len = 24

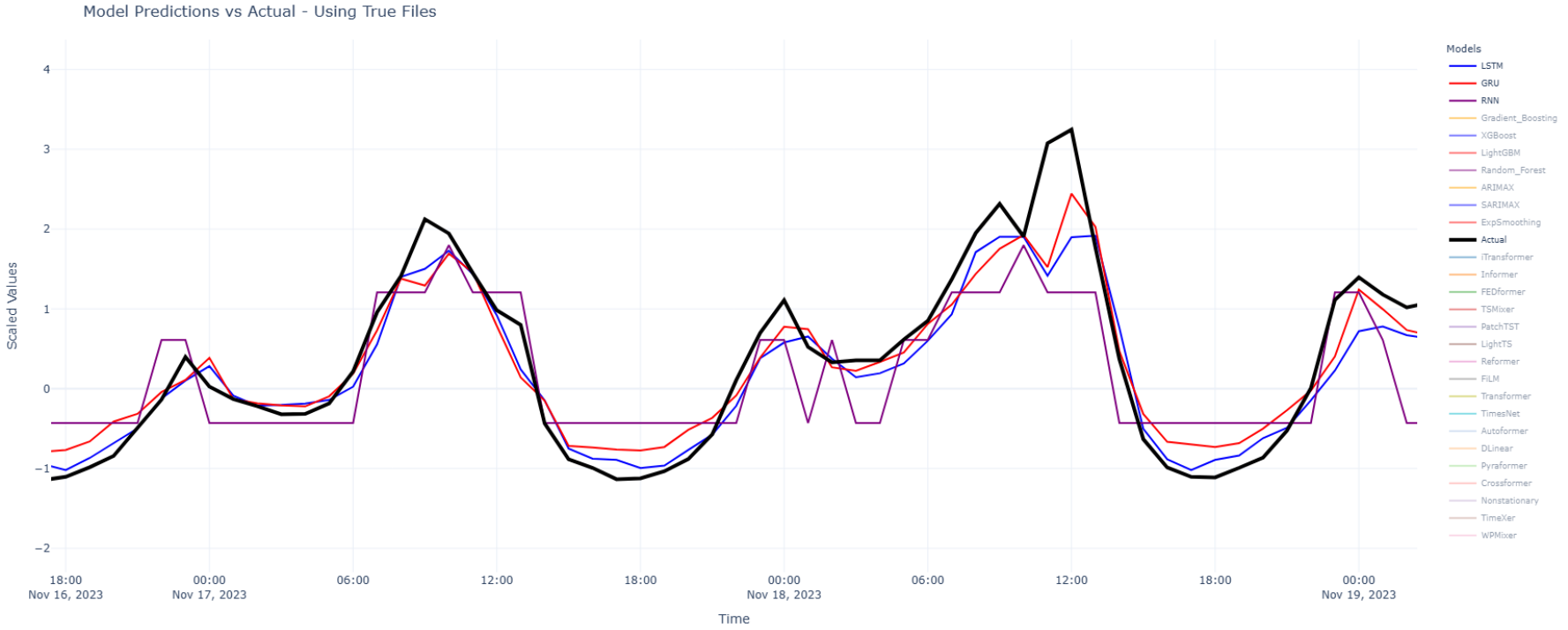

Figure A20. Deep learning methods forecasting results between Nov 17, 2023 and Nov 19, 2023 for seq_len = 168 and pred_len = 24

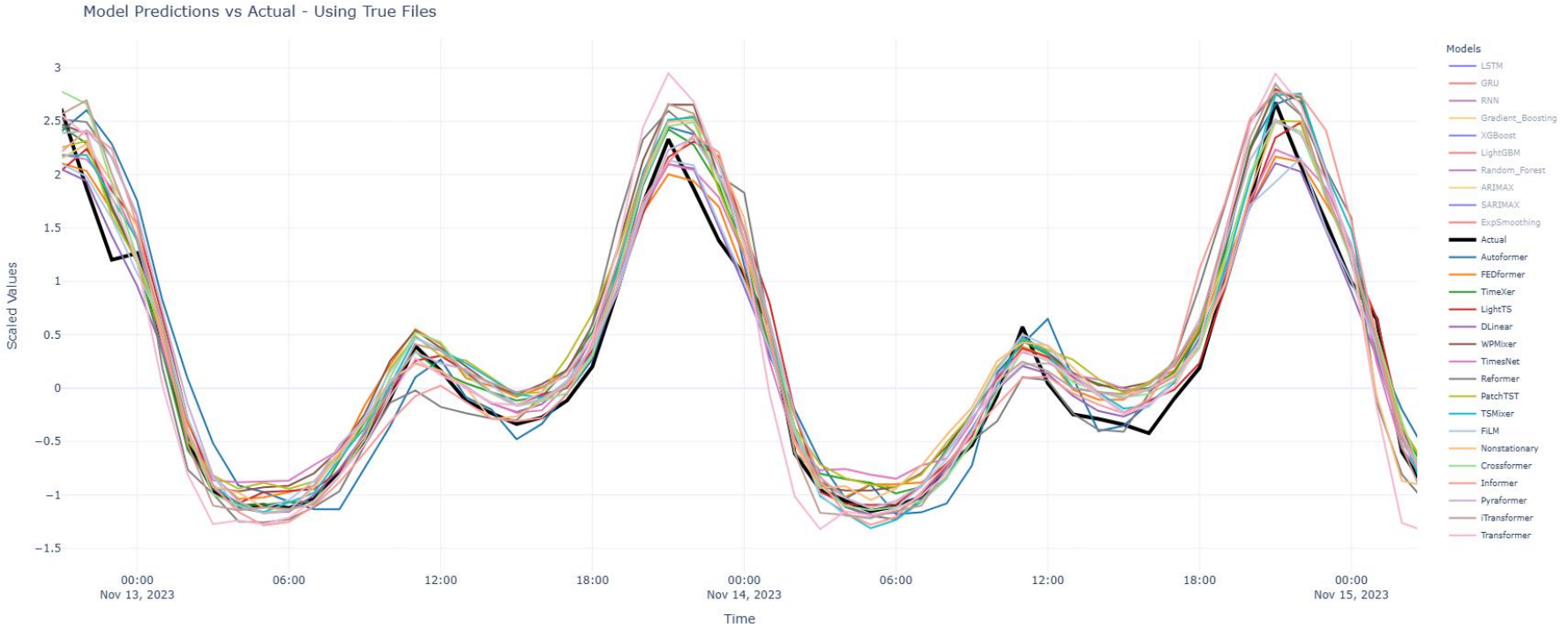

Figure A21. Transformer-based models forecasting results between Nov 13, 2023 and Nov 15, 2023 for seq_len = 168 and pred_len = 36

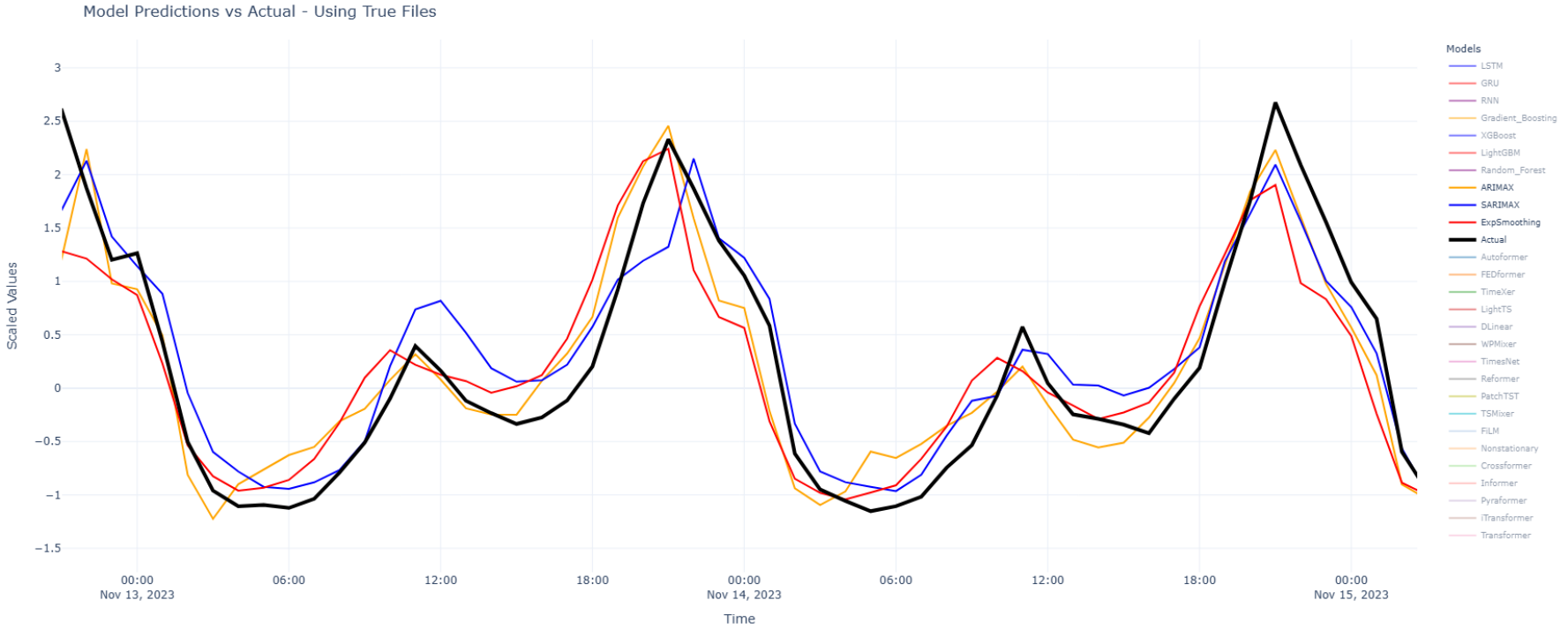

Figure A 22. Statistical methods forecasting results between Nov 13, 2023 and Nov 15, 2023 for seq_len = 168 and pred_len = 36

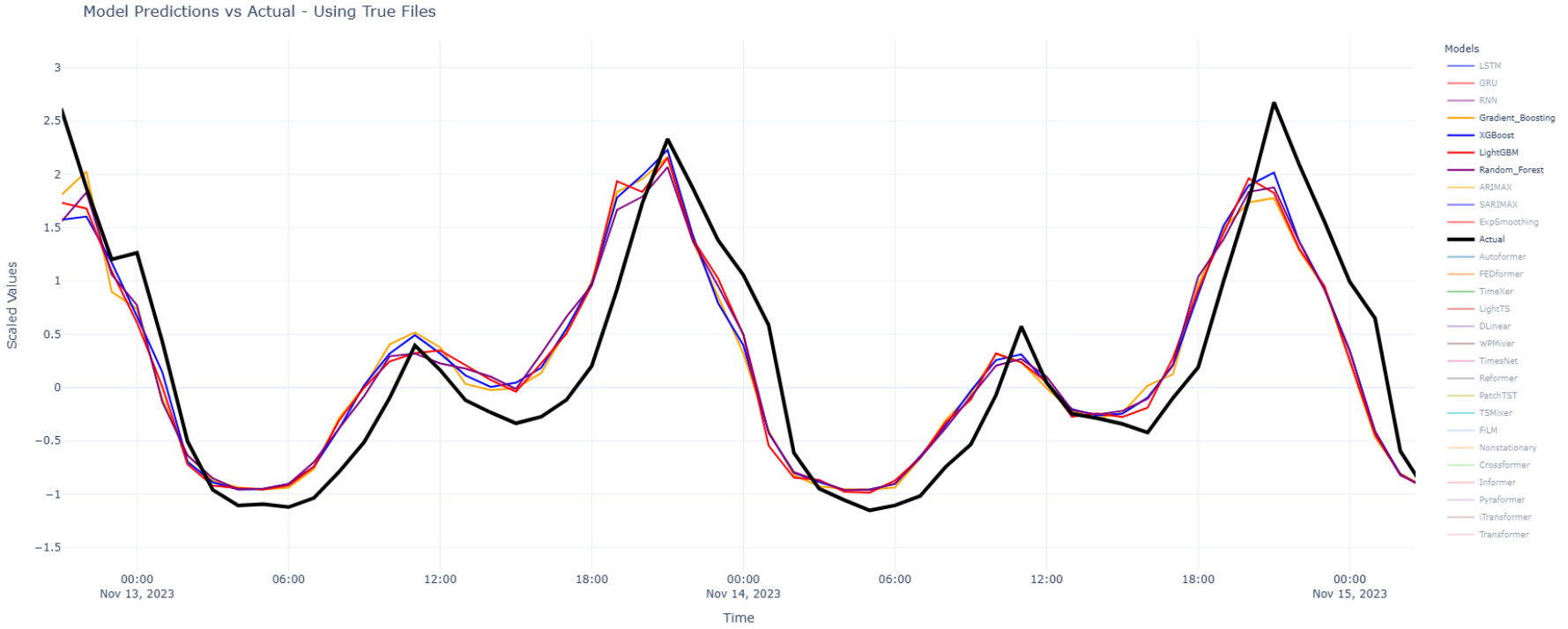

Figure A23. Machine learning methods forecasting results between Nov 13, 2023 and Nov 15, 2023 for seq_len = 168 and pred_len = 36

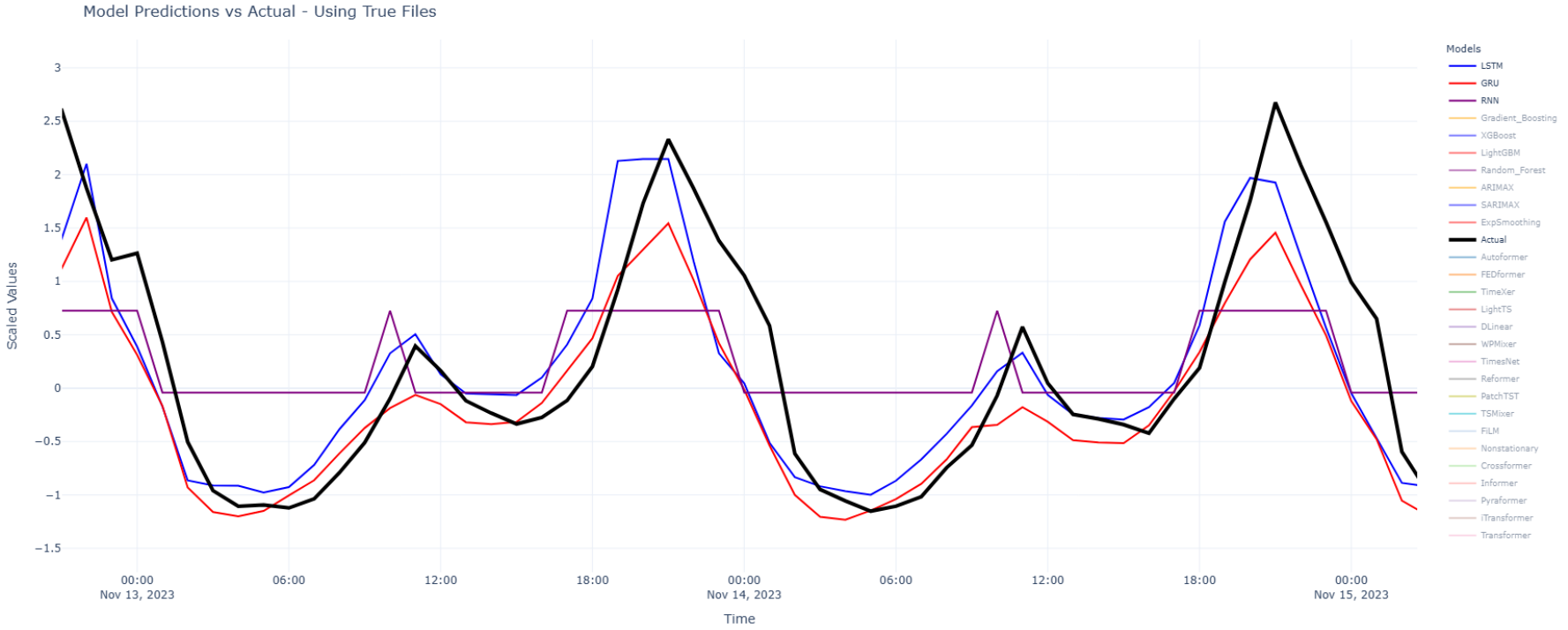

Figure A24. Deep learning methods forecasting results between Nov 13, 2023 and Nov 15, 2023 for seq_len = 168 and pred_len = 36

APPENDIX B

HYPERPARAMETER TUNING RESULTS

Table B1. ARIMAX Hyperparameter Tuning Results

Parameters	MAE	MSE	Runtime (s)
p = 0, d = 0, q = 0	0.604	1.400	9.92
p = 0, d = 0, q = 1	0.640	1.466	14.25
p = 0, d = 0, q = 2	0.689	1.564	18.35
p = 0, d = 1, q = 0	0.767	1.838	4.80
p = 0, d = 1, q = 1	0.862	2.087	9.16
p = 0, d = 1, q = 2	0.916	2.253	9.61
p = 1, d = 0, q = 0	0.671	1.530	10.63
p = 1, d = 0, q = 1	0.722	1.632	15.87
p = 1, d = 0, q = 2	0.733	1.655	19.62
p = 1, d = 1, q = 0	0.978	2.439	4.92
p = 1, d = 1, q = 1	0.926	2.281	8.34
p = 1, d = 1, q = 2	0.905	2.219	20.41
p = 2, d = 0, q = 0	0.744	1.681	14.24
p = 2, d = 0, q = 1	0.743	1.683	17.00
p = 2, d = 0, q = 2	0.744	1.684	18.39
p = 2, d = 1, q = 0	0.896	2.190	5.87
p = 2, d = 1, q = 1	0.787	1.632	20.01
p = 2, d = 1, q = 2	0.750	1.637	23.26

Table B2. SARIMAX Hyperparameter Tuning Results

Parameters	MAE	MSE	Runtime (s)
p = 0, d = 0, q = 0, P = 0, D = 0, Q = 0	0.459	1.218	21.88
p = 0, d = 0, q = 0, P = 0, D = 0, Q = 1	0.464	1.172	459.59
p = 0, d = 0, q = 0, P = 0, D = 1, Q = 0	0.313	0.984	112.51
p = 0, d = 0, q = 0, P = 0, D = 1, Q = 1	0.233	0.801	1052.12
p = 0, d = 0, q = 0, P = 1, D = 0, Q = 0	0.812	1.623	347.18
p = 0, d = 0, q = 0, P = 1, D = 0, Q = 1	0.236	0.802	438.78
p = 0, d = 0, q = 0, P = 1, D = 1, Q = 0	0.270	0.840	668.85
p = 0, d = 0, q = 0, P = 1, D = 1, Q = 1	0.235	0.802	1093.69
p = 0, d = 0, q = 1, P = 0, D = 0, Q = 0	0.553	2.389	122.93
p = 0, d = 0, q = 1, P = 0, D = 0, Q = 1	0.548	2.217	1014.77
p = 0, d = 0, q = 1, P = 0, D = 1, Q = 0	0.432	1.799	221.01
p = 0, d = 0, q = 1, P = 0, D = 1, Q = 1	0.238	0.827	1045.27
p = 0, d = 0, q = 1, P = 1, D = 0, Q = 0	0.790	2.375	546.52
p = 0, d = 0, q = 1, P = 1, D = 0, Q = 1	0.261	0.937	371.63
p = 0, d = 0, q = 1, P = 1, D = 1, Q = 0	0.275	0.882	760.82
p = 0, d = 0, q = 1, P = 1, D = 1, Q = 1	0.240	0.828	1067.23
p = 0, d = 0, q = 2, P = 0, D = 0, Q = 0	0.579	2.806	158.94
p = 0, d = 0, q = 2, P = 0, D = 0, Q = 1	0.610	3.080	1000.44
p = 0, d = 0, q = 2, P = 0, D = 1, Q = 0	0.524	2.567	315.27
p = 0, d = 0, q = 2, P = 0, D = 1, Q = 1	0.241	0.841	1214.88
p = 0, d = 0, q = 2, P = 1, D = 0, Q = 0	0.831	3.232	655.42
p = 0, d = 0, q = 2, P = 1, D = 0, Q = 1	0.416	1.932	621.13
p = 0, d = 0, q = 2, P = 1, D = 1, Q = 0	0.471	2.276	1794.10
p = 0, d = 0, q = 2, P = 1, D = 1, Q = 1	0.243	0.838	1154.49
p = 0, d = 1, q = 0, P = 0, D = 0, Q = 0	1.936	15.767	17.74
p = 0, d = 1, q = 0, P = 0, D = 0, Q = 1	2.811	23.557	895.93
p = 0, d = 1, q = 0, P = 0, D = 1, Q = 0	4.449	34.435	92.82
p = 0, d = 1, q = 0, P = 0, D = 1, Q = 1	2.881	20.033	851.43
p = 0, d = 1, q = 0, P = 1, D = 0, Q = 0	2.048	17.170	211.62
p = 0, d = 1, q = 0, P = 1, D = 0, Q = 1	3.000	24.890	1186.54
p = 0, d = 1, q = 0, P = 1, D = 1, Q = 0	4.861	40.997	471.82
p = 0, d = 1, q = 0, P = 1, D = 1, Q = 1	2.942	21.481	809.13
p = 0, d = 1, q = 1, P = 0, D = 0, Q = 0	1.935	15.746	6.91
p = 0, d = 1, q = 1, P = 0, D = 0, Q = 1	2.779	22.408	735.94
p = 0, d = 1, q = 1, P = 0, D = 1, Q = 0	4.871	40.985	112.77
p = 0, d = 1, q = 1, P = 0, D = 1, Q = 1	2.785	18.647	911.95
p = 0, d = 1, q = 1, P = 1, D = 0, Q = 0	2.405	23.368	460.92
p = 0, d = 1, q = 1, P = 1, D = 0, Q = 1	3.096	26.068	921.62
p = 0, d = 1, q = 1, P = 1, D = 1, Q = 0	4.877	41.251	488.21
p = 0, d = 1, q = 1, P = 1, D = 1, Q = 1	3.416	30.663	1775.90

p = 0, d = 1, q = 2, P = 0, D = 0, Q = 0	1.202	6.640	69.89
p = 0, d = 1, q = 2, P = 0, D = 0, Q = 1	2.689	21.102	925.40
p = 0, d = 1, q = 2, P = 0, D = 1, Q = 0	2.015	8.784	219.68
p = 0, d = 1, q = 2, P = 0, D = 1, Q = 1	2.559	15.857	710.15
p = 0, d = 1, q = 2, P = 1, D = 0, Q = 0	1.973	16.060	352.90
p = 0, d = 1, q = 2, P = 1, D = 0, Q = 1	2.971	24.214	871.91
p = 0, d = 1, q = 2, P = 1, D = 1, Q = 0	1.839	6.808	1317.12
p = 0, d = 1, q = 2, P = 1, D = 1, Q = 1	4.857	59.908	1987.48
p = 1, d = 0, q = 0, P = 0, D = 0, Q = 0	0.547	2.597	62.06
p = 1, d = 0, q = 0, P = 0, D = 0, Q = 1	0.627	4.509	795.87
p = 1, d = 0, q = 0, P = 0, D = 1, Q = 0	0.613	3.125	168.18
p = 1, d = 0, q = 0, P = 0, D = 1, Q = 1	0.238	0.831	1010.49
p = 1, d = 0, q = 0, P = 1, D = 0, Q = 0	0.576	3.234	1296.82
p = 1, d = 0, q = 0, P = 1, D = 0, Q = 1	0.265	0.938	329.56
p = 1, d = 0, q = 0, P = 1, D = 1, Q = 0	0.555	2.754	1937.60
p = 1, d = 0, q = 0, P = 1, D = 1, Q = 1	0.241	0.832	1042.56
p = 1, d = 0, q = 1, P = 0, D = 0, Q = 0	0.490	1.749	118.89
p = 1, d = 0, q = 1, P = 0, D = 0, Q = 1	0.628	4.518	1128.14
p = 1, d = 0, q = 1, P = 0, D = 1, Q = 0	0.616	3.313	432.12
p = 1, d = 0, q = 1, P = 0, D = 1, Q = 1	0.246	0.844	1010.39
p = 1, d = 0, q = 1, P = 1, D = 0, Q = 0	0.578	3.262	1277.29
p = 1, d = 0, q = 1, P = 1, D = 0, Q = 1	0.512	1.715	530.78
p = 1, d = 0, q = 1, P = 1, D = 1, Q = 0	0.580	3.133	2276.71
p = 1, d = 0, q = 1, P = 1, D = 1, Q = 1	0.250	0.841	956.09
p = 1, d = 0, q = 2, P = 0, D = 0, Q = 0	0.517	2.190	368.41
p = 1, d = 0, q = 2, P = 0, D = 0, Q = 1	0.626	4.588	2171.35
p = 1, d = 0, q = 2, P = 0, D = 1, Q = 0	0.610	3.257	472.83
p = 1, d = 0, q = 2, P = 0, D = 1, Q = 1	0.299	1.035	2331.90
p = 1, d = 0, q = 2, P = 1, D = 0, Q = 0	0.561	3.007	2258.71
p = 1, d = 0, q = 2, P = 1, D = 0, Q = 1	0.525	1.697	771.81
p = 1, d = 0, q = 2, P = 1, D = 1, Q = 0	0.565	2.982	2687.26
p = 1, d = 0, q = 2, P = 1, D = 1, Q = 1	0.246	0.838	1112.31
p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0	1.935	15.747	6.37
p = 1, d = 1, q = 0, P = 0, D = 0, Q = 1	2.772	22.356	650.58
p = 1, d = 1, q = 0, P = 0, D = 1, Q = 0	4.684	38.103	76.98
p = 1, d = 1, q = 0, P = 0, D = 1, Q = 1	2.794	18.766	799.17
p = 1, d = 1, q = 0, P = 1, D = 0, Q = 0	2.435	23.923	472.24
p = 1, d = 1, q = 0, P = 1, D = 0, Q = 1	3.123	26.541	838.66
p = 1, d = 1, q = 0, P = 1, D = 1, Q = 0	4.862	41.035	490.97
p = 1, d = 1, q = 0, P = 1, D = 1, Q = 1	3.713	36.273	1574.36
p = 1, d = 1, q = 1, P = 0, D = 0, Q = 0	1.934	15.738	13.60
p = 1, d = 1, q = 1, P = 0, D = 0, Q = 1	2.217	14.750	2499.55
p = 1, d = 1, q = 1, P = 0, D = 1, Q = 0	4.410	33.613	117.25

p = 1, d = 1, q = 1, P = 0, D = 1, Q = 1	2.635	16.737	759.31
p = 1, d = 1, q = 1, P = 1, D = 0, Q = 0	2.346	22.267	647.65
p = 1, d = 1, q = 1, P = 1, D = 0, Q = 1	3.946	41.688	1537.10
p = 1, d = 1, q = 1, P = 1, D = 1, Q = 0	4.749	38.771	518.66
p = 1, d = 1, q = 1, P = 1, D = 1, Q = 1	3.317	27.446	1456.22
p = 1, d = 1, q = 2, P = 0, D = 0, Q = 0	1.929	15.635	14.28
p = 1, d = 1, q = 2, P = 0, D = 0, Q = 1	2.799	22.678	1033.81
p = 1, d = 1, q = 2, P = 0, D = 1, Q = 0	0.627	3.176	237.44
p = 1, d = 1, q = 2, P = 0, D = 1, Q = 1	2.275	12.777	652.52
p = 1, d = 1, q = 2, P = 1, D = 0, Q = 0	1.571	10.721	772.71
p = 1, d = 1, q = 2, P = 1, D = 0, Q = 1	3.028	25.155	1149.78
p = 1, d = 1, q = 2, P = 1, D = 1, Q = 0	0.473	2.173	735.61
p = 1, d = 1, q = 2, P = 1, D = 1, Q = 1	0.568	2.968	1465.90
p = 2, d = 0, q = 0, P = 0, D = 0, Q = 0	0.519	2.137	213.26
p = 2, d = 0, q = 0, P = 0, D = 0, Q = 1	0.629	4.544	1099.24
p = 2, d = 0, q = 0, P = 0, D = 1, Q = 0	0.620	3.323	426.63
p = 2, d = 0, q = 0, P = 0, D = 1, Q = 1	0.243	0.847	1040.96
p = 2, d = 0, q = 0, P = 1, D = 0, Q = 0	0.577	3.245	914.07
p = 2, d = 0, q = 0, P = 1, D = 0, Q = 1	0.712	2.668	591.82
p = 2, d = 0, q = 0, P = 1, D = 1, Q = 0	0.748	4.725	2591.28
p = 2, d = 0, q = 0, P = 1, D = 1, Q = 1	0.245	0.843	1013.22
p = 2, d = 0, q = 1, P = 0, D = 0, Q = 0	0.516	2.170	346.62
p = 2, d = 0, q = 1, P = 0, D = 0, Q = 1	0.629	4.564	1889.13
p = 2, d = 0, q = 1, P = 0, D = 1, Q = 0	0.618	3.332	391.49
p = 2, d = 0, q = 1, P = 0, D = 1, Q = 1	0.373	1.177	1969.17
p = 2, d = 0, q = 1, P = 1, D = 0, Q = 0	0.569	3.097	1428.90
p = 2, d = 0, q = 1, P = 1, D = 0, Q = 1	0.483	1.602	611.46
p = 2, d = 0, q = 1, P = 1, D = 1, Q = 0	0.586	3.174	2471.28
p = 2, d = 0, q = 1, P = 1, D = 1, Q = 1	0.248	0.845	978.83
p = 2, d = 0, q = 2, P = 0, D = 0, Q = 0	0.486	1.699	181.15
p = 2, d = 0, q = 2, P = 0, D = 0, Q = 1	0.614	3.958	1931.44
p = 2, d = 0, q = 2, P = 0, D = 1, Q = 0	0.504	2.275	774.57
p = 2, d = 0, q = 2, P = 0, D = 1, Q = 1	0.274	0.930	2193.48
p = 2, d = 0, q = 2, P = 1, D = 0, Q = 0	0.567	2.956	1400.06
p = 2, d = 0, q = 2, P = 1, D = 0, Q = 1	0.561	2.156	667.90
p = 2, d = 0, q = 2, P = 1, D = 1, Q = 0	0.433	1.773	2713.66
p = 2, d = 0, q = 2, P = 1, D = 1, Q = 1	0.246	0.840	1137.36
p = 2, d = 1, q = 0, P = 0, D = 0, Q = 0	1.613	11.196	55.29
p = 2, d = 1, q = 0, P = 0, D = 0, Q = 1	2.730	21.770	775.74
p = 2, d = 1, q = 0, P = 0, D = 1, Q = 0	4.191	30.537	116.21
p = 2, d = 1, q = 0, P = 0, D = 1, Q = 1	2.612	16.508	664.93
p = 2, d = 1, q = 0, P = 1, D = 0, Q = 0	1.986	16.261	374.67
p = 2, d = 1, q = 0, P = 1, D = 0, Q = 1	2.912	23.337	1295.19

p = 2, d = 1, q = 0, P = 1, D = 1, Q = 0	4.810	39.387	486.47
p = 2, d = 1, q = 0, P = 1, D = 1, Q = 1	3.579	33.381	1549.05
p = 2, d = 1, q = 1, P = 0, D = 0, Q = 0	1.149	6.182	134.07
p = 2, d = 1, q = 1, P = 0, D = 0, Q = 1	3.840	44.225	2209.31
p = 2, d = 1, q = 1, P = 0, D = 1, Q = 0	0.657	3.438	479.95
p = 2, d = 1, q = 1, P = 0, D = 1, Q = 1	2.276	12.839	693.74
p = 2, d = 1, q = 1, P = 1, D = 0, Q = 0	1.976	16.098	391.74
p = 2, d = 1, q = 1, P = 1, D = 0, Q = 1	3.163	27.611	1109.12
p = 2, d = 1, q = 1, P = 1, D = 1, Q = 0	0.496	2.401	1773.67
p = 2, d = 1, q = 1, P = 1, D = 1, Q = 1	2.233	12.072	838.80
p = 2, d = 1, q = 2, P = 0, D = 0, Q = 0	1.893	15.153	48.34
p = 2, d = 1, q = 2, P = 0, D = 0, Q = 1	2.608	20.172	1947.61
p = 2, d = 1, q = 2, P = 0, D = 1, Q = 0	0.566	2.656	331.04
p = 2, d = 1, q = 2, P = 0, D = 1, Q = 1	2.376	13.897	666.91
p = 2, d = 1, q = 2, P = 1, D = 0, Q = 0	2.021	16.758	345.39
p = 2, d = 1, q = 2, P = 1, D = 1, Q = 0	2.159	14.077	2679.63
p = 2, d = 1, q = 2, P = 1, D = 1, Q = 1	0.474	2.321	1534.20

Table B3. Random Forest Hyperparameter Tuning Results

Parameters	MAE	MSE	Runtime (s)
n_est = 50, max_d = 0, min_split = 2	0.153	0.419	26.97
n_est = 50, max_d = 0, min_split = 5	0.152	0.427	25.65
n_est = 50, max_d = 0, min_split = 10	0.153	0.425	24.48
n_est = 50, max_d = 10, min_split = 2	0.156	0.426	17.42
n_est = 50, max_d = 10, min_split = 5	0.155	0.427	18.51
n_est = 50, max_d = 10, min_split = 10	0.156	0.432	17.46
n_est = 50, max_d = 20, min_split = 2	0.153	0.417	26.37
n_est = 50, max_d = 20, min_split = 5	0.152	0.424	25.23
n_est = 50, max_d = 20, min_split = 10	0.153	0.430	24.26
n_est = 100, max_d = 0, min_split = 2	0.153	0.434	54.05
n_est = 100, max_d = 0, min_split = 5	0.152	0.435	51.24
n_est = 100, max_d = 0, min_split = 10	0.153	0.439	48.85
n_est = 100, max_d = 10, min_split = 2	0.156	0.436	34.82
n_est = 100, max_d = 10, min_split = 5	0.156	0.437	34.79
n_est = 100, max_d = 10, min_split = 10	0.157	0.448	34.59
n_est = 100, max_d = 20, min_split = 2	0.153	0.436	52.43
n_est = 100, max_d = 20, min_split = 5	0.152	0.439	50.44
n_est = 100, max_d = 20, min_split = 10	0.153	0.442	48.45
n_est = 200, max_d = 0, min_split = 2	0.152	0.441	108.46
n_est = 200, max_d = 0, min_split = 5	0.152	0.442	103.01
n_est = 200, max_d = 0, min_split = 10	0.152	0.444	98.09
n_est = 200, max_d = 10, min_split = 2	0.155	0.439	69.54
n_est = 200, max_d = 10, min_split = 5	0.155	0.441	69.17
n_est = 200, max_d = 10, min_split = 10	0.156	0.454	69.01
n_est = 200, max_d = 20, min_split = 2	0.152	0.442	105.27
n_est = 200, max_d = 20, min_split = 5	0.152	0.446	101.16
n_est = 200, max_d = 20, min_split = 10	0.152	0.446	97.04

Table B4. GB Hyperparameter Tuning Results

Parameters	MAE	MSE	Runtime (s)
max_d = 3, lr = 0.01, n_est = 100	0.361	0.746	18.33
max_d = 3, lr = 0.01, n_est = 200	0.232	0.532	36.84
max_d = 3, lr = 0.01, n_est = 300	0.189	0.470	55.53
max_d = 3, lr = 0.1, n_est = 100	0.158	0.427	18.33
max_d = 3, lr = 0.1, n_est = 200	0.153	0.421	36.71
max_d = 3, lr = 0.1, n_est = 300	0.152	0.420	55.47
max_d = 3, lr = 0.2, n_est = 100	0.160	0.438	18.33
max_d = 3, lr = 0.2, n_est = 200	0.158	0.435	36.86
max_d = 3, lr = 0.2, n_est = 300	0.159	0.437	55.51
max_d = 5, lr = 0.01, n_est = 100	0.331	0.702	30.25
max_d = 5, lr = 0.01, n_est = 200	0.200	0.522	60.55
max_d = 5, lr = 0.01, n_est = 300	0.163	0.471	90.57
max_d = 5, lr = 0.1, n_est = 100	0.155	0.445	30.40
max_d = 5, lr = 0.1, n_est = 200	0.156	0.449	60.77
max_d = 5, lr = 0.1, n_est = 300	0.157	0.448	91.15
max_d = 5, lr = 0.2, n_est = 100	0.163	0.478	30.51
max_d = 5, lr = 0.2, n_est = 200	0.162	0.479	60.83
max_d = 5, lr = 0.2, n_est = 300	0.164	0.479	91.19
max_d = 7, lr = 0.01, n_est = 100	0.319	0.710	41.47
max_d = 7, lr = 0.01, n_est = 200	0.192	0.569	82.69
max_d = 7, lr = 0.01, n_est = 300	0.159	0.535	124.28
max_d = 7, lr = 0.1, n_est = 100	0.158	0.524	42.01
max_d = 7, lr = 0.1, n_est = 200	0.160	0.527	84.39
max_d = 7, lr = 0.1, n_est = 300	0.160	0.527	126.70
max_d = 7, lr = 0.2, n_est = 100	0.161	0.530	42.16
max_d = 7, lr = 0.2, n_est = 200	0.163	0.531	84.67
max_d = 7, lr = 0.2, n_est = 300	0.163	0.532	126.96

Table B5. XGBoost Hyperparameter Tuning Results

Parameters	MAE	MSE	Runtime (s)
max_d = 3, lr = 0.01, n_est = 100	0.359	0.771	4.97
max_d = 3, lr = 0.01, n_est = 200	0.232	0.597	12.26
max_d = 3, lr = 0.01, n_est = 300	0.190	0.550	16.50
max_d = 3, lr = 0.1, n_est = 100	0.159	0.481	5.89
max_d = 3, lr = 0.1, n_est = 200	0.154	0.459	8.41
max_d = 3, lr = 0.1, n_est = 300	0.153	0.455	17.57
max_d = 3, lr = 0.2, n_est = 100	0.160	0.490	4.06
max_d = 3, lr = 0.2, n_est = 200	0.158	0.480	11.97
max_d = 3, lr = 0.2, n_est = 300	0.159	0.475	21.92
max_d = 5, lr = 0.01, n_est = 100	0.328	0.672	5.90
max_d = 5, lr = 0.01, n_est = 200	0.197	0.519	20.79
max_d = 5, lr = 0.01, n_est = 300	0.162	0.483	30.47
max_d = 5, lr = 0.1, n_est = 100	0.153	0.456	11.55
max_d = 5, lr = 0.1, n_est = 200	0.153	0.454	22.24
max_d = 5, lr = 0.1, n_est = 300	0.153	0.453	28.77
max_d = 5, lr = 0.2, n_est = 100	0.156	0.461	7.87
max_d = 5, lr = 0.2, n_est = 200	0.160	0.462	20.03
max_d = 5, lr = 0.2, n_est = 300	0.161	0.462	29.71
max_d = 7, lr = 0.01, n_est = 100	0.320	0.664	16.15
max_d = 7, lr = 0.01, n_est = 200	0.189	0.472	28.39
max_d = 7, lr = 0.01, n_est = 300	0.156	0.425	40.39
max_d = 7, lr = 0.1, n_est = 100	0.150	0.411	15.01
max_d = 7, lr = 0.1, n_est = 200	0.151	0.413	24.38
max_d = 7, lr = 0.1, n_est = 300	0.153	0.413	39.90
max_d = 7, lr = 0.2, n_est = 100	0.156	0.446	19.91
max_d = 7, lr = 0.2, n_est = 200	0.158	0.446	26.83
max_d = 7, lr = 0.2, n_est = 300	0.159	0.446	42.99

Table B6. LightGBM Hyperparameter Tuning Results

Parameters	MAE	MSE	Runtime (s)
leaves = 31, lr = 0.01, n_est = 100	0.322	0.614	0.30
leaves = 31, lr = 0.01, n_est = 200	0.194	0.486	0.28
leaves = 31, lr = 0.01, n_est = 300	0.165	0.471	0.39
leaves = 31, lr = 0.1, n_est = 100	0.161	0.460	0.13
leaves = 31, lr = 0.1, n_est = 200	0.162	0.471	0.21
leaves = 31, lr = 0.1, n_est = 300	0.164	0.472	0.29
leaves = 31, lr = 0.2, n_est = 100	0.166	0.482	0.13
leaves = 31, lr = 0.2, n_est = 200	0.170	0.487	0.20
leaves = 31, lr = 0.2, n_est = 300	0.172	0.490	0.29
leaves = 50, lr = 0.01, n_est = 100	0.316	0.615	0.23
leaves = 50, lr = 0.01, n_est = 200	0.189	0.479	0.39
leaves = 50, lr = 0.01, n_est = 300	0.162	0.463	0.54
leaves = 50, lr = 0.1, n_est = 100	0.159	0.463	0.17
leaves = 50, lr = 0.1, n_est = 200	0.161	0.474	0.29
leaves = 50, lr = 0.1, n_est = 300	0.164	0.476	0.40
leaves = 50, lr = 0.2, n_est = 100	0.168	0.489	0.16
leaves = 50, lr = 0.2, n_est = 200	0.172	0.489	0.30
leaves = 50, lr = 0.2, n_est = 300	0.174	0.489	0.40
leaves = 100, lr = 0.01, n_est = 100	0.311	0.611	0.35
leaves = 100, lr = 0.01, n_est = 200	0.187	0.483	0.74
leaves = 100, lr = 0.01, n_est = 300	0.161	0.466	0.94
leaves = 100, lr = 0.1, n_est = 100	0.163	0.457	0.29
leaves = 100, lr = 0.1, n_est = 200	0.166	0.468	0.53
leaves = 100, lr = 0.1, n_est = 300	0.168	0.472	0.78
leaves = 100, lr = 0.2, n_est = 100	0.169	0.466	0.29
leaves = 100, lr = 0.2, n_est = 200	0.172	0.472	0.51
leaves = 100, lr = 0.2, n_est = 300	0.173	0.475	0.76

Table B7. Deep learning methods Hyperparameter Tuning Results

Parameters	MAE	MSE	Runtime (s)
units = 50, lr = 0.001, batch = 32	0.224	0.540	13.52
units = 50, lr = 0.001, batch = 64	0.180	0.565	7.77
units = 50, lr = 0.001, batch = 128	0.196	0.572	5.03
units = 50, lr = 0.01, batch = 32	0.280	0.803	13.46
units = 50, lr = 0.01, batch = 64	0.278	0.725	7.47
units = 50, lr = 0.01, batch = 128	0.336	0.781	5.02
units = 50, lr = 0.1, batch = 32	0.438	0.866	13.35
units = 50, lr = 0.1, batch = 64	0.489	1.057	8.00
units = 50, lr = 0.1, batch = 128	0.457	0.933	5.05
units = 100, lr = 0.001, batch = 32	0.198	0.546	16.64
units = 100, lr = 0.001, batch = 64	0.203	0.579	11.36
units = 100, lr = 0.001, batch = 128	0.278	0.601	9.44
units = 100, lr = 0.01, batch = 32	0.270	0.777	16.74
units = 100, lr = 0.01, batch = 64	0.284	0.758	11.91
units = 100, lr = 0.01, batch = 128	0.551	0.944	9.37
units = 100, lr = 0.1, batch = 32	0.358	0.902	16.96
units = 100, lr = 0.1, batch = 64	0.484	1.019	11.29
units = 100, lr = 0.1, batch = 128	0.338	0.858	9.29
units = 200, lr = 0.001, batch = 32	0.278	0.650	29.83
units = 200, lr = 0.001, batch = 64	0.246	0.637	25.89
units = 200, lr = 0.001, batch = 128	0.210	0.545	24.06
units = 200, lr = 0.01, batch = 32	0.312	0.829	29.34
units = 200, lr = 0.01, batch = 64	0.285	0.817	26.00
units = 200, lr = 0.01, batch = 128	0.309	0.805	24.22
units = 200, lr = 0.1, batch = 32	0.828	1.568	28.57
units = 200, lr = 0.1, batch = 64	1.119	2.245	25.88
units = 200, lr = 0.1, batch = 128	0.491	0.949	24.18

REFERENCES

Abu Al-Haija, Q., Mao, Q., & Al Nasr, K. (2019). Forecasting the number of monthly active Facebook and Twitter worldwide users using ARMA model. *Journal of Computer Science*, 15(4), 499–510. https://doi.org/10.3844/jcssp.2019.499.510

Afzal, S., Prodan, R., & Timmerer, C. (2024). Green video streaming: Challenges and opportunities. *ACM SIGMultimedia Records*, 15(1), 1–1. https://doi.org/10.1145/3699843.3699846

Alharbi, F. R., & Csala, D. (2022). A seasonal autoregressive integrated moving average with exogenous factors (SARIMAX) forecasting model-based time series approach. *Inventions*, *7*(4), 94. https://doi.org/10.3390/inventions7040094

Almeida, M. M. S. C. de, Pereira, T. E., & Morais, F. (2022). A case study of proactive auto-scaling for an ecommerce workload. *arXiv preprint arXiv:2211.11928*. https://doi.org/10.48550/arXiv.2211.11928

Arkenberg, C., Casey, M., & Wigginton, C. (2019, December 9). *Coming to a CDN near you: Videos, games, and much, much more*. Deloitte Insights. https://www2.deloitte.com/us/en/insights/industry/technology/technology-media-and-telecom-predictions/2020/content-delivery-networks-video-streaming.html

Baym, G. (2024, December 18). Can Netflix handle live event buffering issues ahead of NFL Christmas Day games? (B. Baum, Interviewer) [Interview]. In *Temple Now | Temple University*. https://news.temple.edu/news/2024-12-17/can-netflix-handle-live-event-buffering-issues-ahead-nfl-christmas-day-games

Behnam, N., & Fedorov, S. (2023, December 14). *Using Traffic Modeling to Load-Balance Netflix Traffic at Global Scale*. InfoQ. https://www.infoq.com/presentations/load-balancing-netflix

Bi, J., Zhang, X., Yuan, H., Zhang, J., & Zhou, M. (2021). A hybrid prediction method for realistic network traffic with temporal convolutional network and LSTM. *IEEE Transactions on Automation Science and Engineering*, *19*(3), 1869–1879. https://doi.org/10.1109/tase.2021.3077537

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32. https://doi.org/10.1023/a:1010933404324

Casado-Vara, R., Martin del Rey, A., Pérez-Palau, D., de-la-Fuente-Valentín, L., & Corchado, J. M. (2021). Web traffic time series forecasting using LSTM neural networks with distributed asynchronous training. *Mathematics*, *9*(4), 421. https://doi.org/10.3390/math9040421

Chakraborty, T., Chattopadhyay, S., & Ghosh, I. (2019). Forecasting dengue epidemics using a hybrid methodology. *Physica A: Statistical Mechanics and Its Applications*, *527*, 121266. https://doi.org/10.1016/j.physa.2019.121266

Chen, S.-A., Li, C.-L., Arik, S. O., Yoder, N. C., & Pfister, T. (2023). TSMixer: An all-MLP architecture for time series forecasting. *arXiv preprint arXiv:2303.06053.* https://doi.org/10.48550/arXiv.2303.06053

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. https://doi.org/10.48550/arXiv.1406.1078

Eskandari, H., Imani, M., & Moghaddam, M. P. (2021). Convolutional and recurrent neural network based model for short-term load forecasting. *Electric Power Systems Research*, *195*, 107173. https://doi.org/10.1016/j.epsr.2021.107173

Faloutsos, C., Gasthaus, J., Januschowski, T., & Wang, Y. (2018). Forecasting big time series: Old and new. *Proceedings of the VLDB Endowment*, *11*(12), 2102–2105. https://doi.org/10.14778/3229863.3229878

Famaey, J., Iterbeke, F., Wauters, T., & De Turck, F. (2013). Towards a predictive cache replacement strategy for multimedia content. *Journal of Network and Computer Applications*, *36*(1), 219–227. https://doi.org/10.1016/j.jnca.2012.08.014

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Guruge, P. B., & Priyadarshana, Y. (2025). Time series forecasting-based Kubernetes autoscaling using Facebook Prophet and Long Short-Term Memory. *Frontiers in Computer Science*, *7*:1509165. https://doi.org/10.3389/fcomp.2025.1509165

He, J., Yuan, K., Zhong, Z., & Sun, Y. (2024). Enhancing short-term power load forecasting with a TimesNet-Crossformer-LSTM Approach. *IEEE Access*, *12*, 56774–56788. https://doi.org/10.1109/access.2024.3383912

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hoque, K. E., & Aljamaan, H. (2021). Impact of hyperparameter tuning on machine learning models in stock price forecasting. *IEEE Access*, *9*, 163815–163830. https://doi.org/10.1109/ACCESS.2021.3134138

Jiang, W. (2021). Internet traffic prediction with deep neural networks. *Internet Technology Letters*, *5*(2), e314. https://doi.org/10.1002/itl2.314

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*. https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Khan, S., & Alghulaiakh, H. (2020). ARIMA model for accurate time series stocks forecasting. *International Journal of Advanced Computer Science and Applications*, *11*(7). https://doi.org/10.14569/ijacsa.2020.0110765

Khashei, M., & Hajirahimi, Z. (2017). Performance evaluation of series and parallel strategies for financial time series forecasting. *Financial Innovation*, *3*, 24. https://doi.org/10.1186/s40854-017-0074-9

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2019, December 20). Reformer: The efficient Transformer. *arXiv preprint arXiv:2001.04451*. https://doi.org/10.48550/arXiv.2001.04451

Kougioumtzidis, G., Poulkov, V. K., Lazaridis, P. I., & Zaharis, Z. D. (2025). Mobile network traffic prediction using Temporal Fusion Transformer. *IEEE Transactions on Artificial Intelligence*, 1–15. https://doi.org/10.1109/tai.2025.3556627

Lee, J., & Cho, Y. (2022). National-scale electricity peak load forecasting: Traditional, machine learning, or hybrid model? *Energy*, *239*, 122366. https://doi.org/10.1016/j.energy.2021.122366

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., & Dustdar, S. (2022, January 29). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. *International Conference on Learning Representations*. https://openreview.net/forum?id=0EXmFzUn5I

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., & Long, M. (2024). iTransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*. https://doi.org/10.48550/arXiv.2310.06625

Liu, Y., Wu, H., Wang, J., & Long, M. (2022). Non-stationary Transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35, 9881–9893. https://proceedings.neurips.cc/paper_files/paper/2022/hash/4054556fcaa934b0bf76da52cf4f92cb-Abstract-Conference.html

Maku, T. O., Adehi, M. U., & Adenomon, M. O. (2023). Modeling and forecasting electricity consumption in Nigeria using ARIMA and ARIMAX time series models. *Science World Journal*, *18*(3), 414–421. https://dx.doi.org/10.4314/swj.v18i3.14

Masini, R. P., Medeiros, M. C., & Mendes, E. F. (2021). Machine learning advances for time series forecasting. *Journal of Economic Surveys*, *37*(1), 76–111. https://doi.org/10.1111/joes.12429

Mettu, N. R., & Sasikala, T. (2018). Prediction analysis on web traffic data using time series modeling, RNN and ensembling techniques. In *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*, 611–618. https://doi.org/10.1007/978-3-030-03146-6_67

Murad, M. M. N., Aktukmak, M., & Yilmaz, Y. (2025). WPMixer: Efficient multi-resolution mixing for long-term time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, *39*(18), 19581–19588. https://doi.org/10.1609/aaai.v39i18.34156

MV, M. K., BH, P., BS, P., HR, S., NS, P., & BJ, R. (2024). Wavelet-Based hybrid ensemble forecasting for accurate web traffic prediction. *2024 First International Conference on Innovations in Communications, Electrical and Computer Engineering (ICICEC)*, 1–10. https://doi.org/10.1109/icicec62498.2024.10808683

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*. https://doi.org/10.48550/arXiv.2211.14730

Nogueira, J., Guardalben, L., Cardoso, B., & Sargento, S. (2017). Catch-up TV forecasting: enabling next-generation over-the-top multimedia TV services. *Multimedia Tools and Applications*, *77*, 14527–14555. https://doi.org/10.1007/s11042-017-5043-9

Nunnagoppula, H., Katragadda, K., & Ramesh, M. (2023). Website traffic forecasting using deep learning techniques. *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*, 531–536. https://doi.org/10.1109/aisc56616.2023.10085005

Puri, R. (2022, April 26). *Ingesting live video streams at global scale*. Twitch Blog. https://blog.twitch.tv/en/2022/04/26/ingesting-live-video-streams-at-global-scale/

Sá, S. L. de, Rocha, A. A. de A., & Paes, A. (2021). Predicting popularity of video streaming services with representation learning: A survey and a real-world case study. *Sensors*, *21*(21), 7328. https://doi.org/10.3390/s21217328

Saayman, A., & Botha, I. (2015). Non-linear models for tourism demand forecasting. *Tourism Economics*, *23*(3), 594–613. https://doi.org/10.5367/te.2015.0532

Shelatkar, T., Tondale, S., Yadav, S., & Ahir, S. (2020). Web traffic time series forecasting using ARIMA and LSTM RNN. *ITM Web of Conferences*, *32*, 03017. https://doi.org/10.1051/itmconf/20203203017

Singh, M. (2024). AI-driven predictive auto-Scaling for cloud-native Systems with real-time anomaly detection. *International Journal of Intelligent Systems and Applications in Engineering*, *12*(22s), 2004–2018. https://ijisae.org/index.php/IJISAE/article/view/7420

Statista Market Insights. (2024, November 12). *Number of users in the TV & video market for different segments worldwide 2019 to 2029*. Statista; Statista. https://www.statista.com/forecasts/1442792/number-of-users-tv-video-market-for-different-segments-worldwide

Taylor, S. J., & Letham, B. (2017). Forecasting at scale. *PeerJ Preprints 5*, e3190v2. https://doi.org/10.7287/peerj.preprints.3190v2

Trzciński, T., Andruszkiewicz, P., Bocheński, T., & Rokita, P. (2017). Recurrent neural networks for online video popularity prediction. In *Foundations of Intelligent Systems: 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings 23* (pp. 146–153). Springer International Publishing. https://doi.org/10.1007/978-3-319-60438-1_15

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*. https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Wang, P., Zheng, X., Ai, G., Liu, D., & Zhu, B. (2020). Time series prediction for the epidemic trends of COVID-19 using the improved LSTM deep learning method: Case studies in Russia, Peru and Iran. *Chaos, Solitons & Fractals*, *140*, 110214. https://doi.org/10.1016/j.chaos.2020.110214

Wang, Y., & Guo, Y. (2020). Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost. *China Communications*, *17*(3), 205–221. https://doi.org/10.23919/jcc.2020.03.017

Wang, Y., Wu, H., Dong, J., Qin, G., Zhang, H., Liu, Y., Qiu, Y., Wang, J., & Long, M. (2024). TimeXer: Empowering transformers for time series forecasting with exogenous variables. *arXiv preprint arXiv:2402.19072*. https://doi.org/10.48550/arXiv.2402.19072

Weerts, H. J., Mueller, A. C., & Vanschoren, J. (2020). Importance of tuning hyperparameters of machine learning algorithms. *arXiv preprint arXiv:2007.07588*. https://doi.org/10.48550/arXiv.2007.07588

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., & Long, M. (2023, February 1). TimesNet: Temporal 2D-variation modeling for general time series analysis. *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=ju_Uqw384Oq

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, *34*, 22419–22430. https://papers.nips.cc/paper_files/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(9), 11121–11128. https://doi.org/10.1609/aaai.v37i9.26317

Zeng, Z., Kaur, R., Siddagangappa, S., Rahimi, S., Balch, T., & Veloso, M. (2023). Financial time series forecasting using CNN and Transformer. *arXiv preprint arXiv:2304.04912*. https://doi.org/10.48550/arXiv.2304.04912

Zhang, T., Zhang, Y., Cao, W., Bian, J., Yi, X., Zheng, S., & Li, J. (2022). Less is more: Fast multivariate time series forecasting with light sampling-oriented MLP structures. *arXiv preprint arXiv:2207.01186*. https://doi.org/10.48550/arXiv.2207.01186

Zhang, Y., & Yan, J. (2023, February 1). Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=vSVLM2j9eie

Zhang, Y., Zhu, C., & Wang, Q. (2020). LightGBM-based model for metro passenger volume forecasting. *IET Intelligent Transport Systems*, *14*(13), 1815–1823. https://doi.org/10.1049/iet-its.2020.0396

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12), 11106–11115. https://doi.org/10.1609/aaai.v35i12.17325

Zhou, T., Ma, Z., Wang, X., Wen, Q., Sun, L., Yao, T., Yin, W., & Jin, R. (2022). FiLM: Frequency improved legendre memory model for long-term time series forecasting. *Advances in neural information processing systems*, *35*, 12677–12690. https://papers.nips.cc/paper_files/paper/2022/hash/524ef58c2bd075775861234266e5e020-Abstract-Conference.html

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). FEDformer: Frequency enhanced decomposed Transformer for long-term series forecasting. *Proceedings of the 39th International Conference on Machine Learning*, *162*, 27268–27286. https://proceedings.mlr.press/v162/zhou22g.html

Zuo, C., Wang, J., Liu, M., Deng, S., & Wang, Q. (2023). An ensemble framework for short-term load forecasting based on TimesNet and TCN. *Energies*, *16*(14), 5330–5330. https://doi.org/10.3390/en16145330