INTERNATIONAL CONFERENCE ON
ARTIFICIAL INTELLIGENCE IN MULTIDISCIPLINARY APPLICATIONS
13th -15th February 2026 | gapHQ, Colombo 07, Sri lanka

HYBRID DEEP LEARNING MODELS FOR TIME SERIES FORECASTING OF USER ENGAGEMENT IN PYTHON SOFTWARE APPLICATIONS

V Janarthan[1, 4*], JDT Erandi [2, 4], YMS Tharaka [1, 4], M Vasavan [3, 4], K Luxshi [3, 4]
[1]*Department of Software Engineering, Faculty of Computing*
[2]*Department of Data Science, Faculty of Computing*
[3] *Department of Physical Sciences and Technology*
[4]*Sabaragmuwa University of Sri Lanka*
**Correspondence E-mail: janarthanvijayakumar@gmail.com, TP: +94 712076399*

Abstract: Understanding users' engagement in python-based software applications is important for improving usability, maintaining user satisfaction, and strategic future planning. Python usage logs contain important but rarely used variables like timestamps, functional interaction, session and download counts to analyze user engagement. However existing traditional analysis mainly supports statistical analysis rather than capturing temporal dependencies, irregular usage patterns, and domain specific variables like feature updates or version releases and these studies mainly focused on single applications like Instagram, Facebook and TikTok but not considered as python-based applications. So, this research addresses this gap by developing a time series forecasting framework with use of Python application usage logs. This research study analyze and compare three types of forecasting model like traditional statistical models (ARIMA, SARIMA and Prophet), deep learning models (LSTM and GRU), and hybrid models (Prophet-LSTM, ARIMA-GRU and Prophet-ElasticNet), guided by research questions assessing by evaluating their prediction accuracy and testing under different use cases, such as sparse logs and bursty activity, to examine their robustness and practical suitability for Python application forecasting. The study started with literature reviewing of limits of existing forecasting models, limits of features and application types, and less studies on python applications. Continuing with accurate methodology including data analysis preprocessing, feature engineering, models development, evaluation (MAE, RMSE, and MAPE) and testing and comparison of models. This study involves a Python native forecasting tool developed with Django, React, REST-API, TensorFlow, Prophet and python libraries with developed model artifacts. Findings explain that hybrid models are highly accurate (93.20% and 91.47%) other than both statistical and deep learning models particularly under the sparse logs and bursty conditions. So, the research concludes that irregular features and hybrid residual prediction methods provide high accuracy and offer valuable insights for python developers and product teams.

Keywords: Time Series; Engagement Forecasting; Prophet-LSTM; Usage Logs; Python Software Applications.

1. Introduction

The knowledge of user-software interaction is now considered as one of the major areas of concern by both developers and consumers, especially in this time when the success of a product relies heavily on the data-driven decision-making process. For applications that are used in the Python ecosystem, which may be web-based platforms, data science tools, academic software, or even open-source libraries, user engagement is a major factor for determining usability, value, and future prospects. The user metrics like daily active users, session frequency, and feature adoption in addition to error occurrences, give the product team an idea of how users are interacting with the application's features over time. Usually, these data are stored in the system logs that capture timestamped interactions, system events, and contextual information. Despite the fact that such logs are a great source of temporal data, it is still a very hard-to-do task to convert them into actionable foresight.

The majority of analytics solutions currently in use for Python application development are mainly retrospective. Featuring a special focus on descriptive or diagnostic analytics, these analytics services will in brief summarize historical usage patterns, trends, and past bottlenecks. These methods are good for understanding what has happened already but are inadequate for telling developers and stakeholders how user behavior will change in the future. This situation becomes especially challenging when the user activity is very dynamic, irregular, or event-driven, e.g., academic applications that go through semester-based usage spikes, developer tools whose popularity is determined by version releases or Python libraries that get surges in usage due to external adopters or updates. Consequently, teams are frequently placed in the position of having to react to shifts in user engagement instead of having the opportunity to plan for them.

The task of predicting user interaction is very difficult because of the complex and non-linear character of software usage behavior. User interaction is hardly ever consistent or solely dependent on trends; it is affected by various factors interacting with one another, like better features, updates, and learning curves along with the seasons, holidays, and outside technology or educational events. For instance, a Python library that is helpful in learning situations may have rapid usage during the testing period, and then the usage may quickly drop afterward. Commonly used analytical techniques, like simple regression models or cohort analysis, typically presume a constant state or overlook the time relationships among the data causing their predictions to be quite inaccurate in situations where inconsistencies like those in real life occur. These limitations point to the necessity for forecasting methods that will explicitly cater for time-dependent structures, seasonality, and non-linear behavior, which are characteristics of Python application usage logs.

Time series forecasting has come up with a very good solution to this problem by allowing to model the temporal relations and future user engagement according to past data. Classical statistics models like ARIMA and its seasonal variations have for a long time been the go-to models for identifying the linear trend and seasonality in time series data. They work best during the periods of stable seasonal cycles, but The usage of Python applications, on the other hand, tends to not always behave linearly—they can sometimes change suddenly and have long-term dependencies which are difficult to manage traditionally. It is here where deep learning, particularly recurrent neural networks like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, has proved to be very advantageous by their ability to learn the intricate temporal relationships of the most complex nonlinear types in a sequential way.

The unique advantage of Python for such forecasting methods through its robust ecosystem of data science and machine learning libraries is an important factor for both their implementation and evaluation. Statistical modeling tools like Statsmodels allow for the execution of ARIMA and SARIMA models, whereas Prophet helps with automated forecasting to a great extent by taking care of trends and seasonality as its built-in feature. Meanwhile, libraries for deep learning such as TensorFlow and PyTorch give support to the creation of sophisticated neural network architectures for time series prediction. The combination of Python with its excellent data preprocessing, visualization, and logging capabilities makes the transformation of raw system logs into forecast analysis-ready structured time series a smooth process.

Even though there are technical advantages, the current research and industrial applications have mainly been centered on the engagement forecasting for very big consumer platforms like social media or e-commerce systems. Python-specific applications have been neglected, and that is so far mainly because they are sparse, bursty, or context-dependent in their usage patterns. The libraries that are open-source, the tools that are produced in academia, and the applications aimed at developers all have difficulties that are peculiar to their types, the problems being dependency on release cycles, version updates, and external adoption trends. In most cases, these elements are neglected in the common engagement models, which results in the disconnection between the theoretical forecasting methods and their practical application to Python ecosystems.

This research fills a gap in the literature and is thus motivated by this issue. The main goal of the present work is to develop and evaluate time series forecasting models that will be used in the prediction of user activities from the logs of Python application usage, and thus to move forward from reactive analytics. This research strategically weighs the classical statistical techniques, the deep learning methods, and the hybrid models which make the most of the advantages of both, the aim being to pinpoint the forecasting techniques that cohere in terms of accuracy, interpretability, and practical usability. In addition, the research points out the necessity to include contextual elements like the release of the software, updates of the features, changes in the version, and calendar effects in the forecasting model to make its applicability to the real world better and to increase the reliability of the predictions.

The most important contribution of this study is that it thoroughly investigates the forecasting of user engagement in Python applications, an area of research that has been largely neglected by the writers of existing literature. Instead of suggesting a model that is only theoretical, this research intends to introduce a practical forecasting framework that Python programmers and product teams will be able to adopt. System log data analysis and testing different modeling approaches in real usage situations provide the study with practical suggestions concerning feature prioritization, retention planning, and resource allocation.

The remaining parts of this document are organized accordingly. The following part unveils the characteristic features of the logs generated by Python application usage and elaborates on the data preprocessing steps that are needed to make it suitable for time series analysis. Then, a methodology section explaining the different statistical, deep learning, and hybrid forecasting models that were taken into consideration for the research is included. The later parts include the description of the experimental framework and the assessment method. The last part wraps up with the implications for practice, limitations, and suggestions for future research in user engagement forecasting in the context of Python applications divided into discussion.

2. Related Works / Literture Review

The studies dealing with foreseeing user engagement in Python applications point out the growth of time series forecasting, machine learning, and hybrid analytical methods. The most recent research is witnessing a trend of forecasting user behavior as a means of improving decision-making, feature planning, and resource allocation throughout different Python-powered software systems.

2.1 Why User Engagement Prediction is Important?

The prediction of user engagement is gaining more and more importance in software engineering, particularly in the case of mobile and web applications developed in Python. The ability to foresee the interactions of users enables the developers to make feature plans strategically and to increase the satisfaction of users. Liu et al. mention that the financial sector is one such industry that heavily relies on AI-based predictive analytics for determining risk (Moumane et al., 2024), which is a good indication of the general importance of the prediction of future behavior patterns. To be precise, Python applications produce very large volumes of logs; however, the process of transforming these logs into precise forecasts is still a difficulty area because of the irregularity of usage patterns, low user participation, and the constantly changing needs of users.

2.2 Research Gap and Problem Context

Despite the fact that the forecasting process is extensively studied in such areas as meteorology and economics, the case of Python software applications presents some specific difficulties. The usage usually follows a sort of a non-regular pattern, and the demand reflects certain features, external factors, or all-release cycles. Liu et al. have demonstrated that the interactions with applications are frequently complicated and cannot easily be represented through standard time series models (Liu et al., 2019). Even though there are generic forecasting tools available, the issue of the peculiar characteristics of Python application logs is rarely addressed in the literature, thus leaving a gap in the development of specialized forecasting methodologies that are in tune with software usage behaviors.

2.3 Key Constructs

Engagement with a user is real metrics calculated by passive events such as the length of a session and others. The engagement of users is described by researchers as the sum of attention, interactivity, and control that the end users are experiencing (Kong et al., 2021). The term forecasting is used to indicate the process of extracting insights from historical data in order to predict the future values (Makridakis et al., 1984), while time series analysis carries with it the meaning of a statistical method applied to data that is indexed in a sequence (Tunnicliffe Wilson, 2016). Python's ecosystem consisting of libraries for modeling, preprocessing, and visualization not only supports the analytical processes but also does so very effectively (McKinney, 2010).

2.4 Classical Statistical Methods

Time series forecasting is built on the classical statistical models, notably ARIMA. ARIMA employs the use of autoregression, differencing, and moving averages for the exposure of the data's linear and seasonal structures (Bora, 2021). Hyndman and Athanasopoulos state that it is necessary to transform the data into a stationary form using seasonal differencing, which is the main step before ARIMA modeling (Hyndman and Athanasopoulos, 2018). Brownlee, in turn, facilitated the out-of-sample predictions in Python with his tutorial (Brownlee, 2020). Such techniques work best with datasets that have stable seasonal patterns usually found in situations like academic software usage which goes up and down in a predictable way.

When evaluating the baseline performance for forecasting social media time series, it was found out that statistical models could still be a good choice in stable conditions (Ng et al., 2023). The use of the statsmodels package in Python makes it easier to work with ARIMA (Seabold and Perktold, 2010), thus allowing even those with little expertise to attempt it. Yet, the conventional methods have the drawbacks of not being able to depict the underlying non-linearities or the sudden alternation of the engagement data and therefore they are in a way outdone by the more powerful methods of machine learning (Zemkoho, 2022, Zhang, 2003).

2.5 Methods of Machine Learning

In an effort to go beyond the restrictions of classical models, the use of machine learning (ML) and deep learning techniques is becoming more and more common in recent research. LSTM networks which are capable of processing, by managing and understanding, the complex temporal relations over long sequences are incredibly effective for this particular task (Liu and Wang, 2024). Their design not only allows for the capture of both linear and nonlinear trends in engagement patterns but also leads to a better performance than the classical models in the case of dynamic scenarios.

On the other hand, Graph Neural Networks (GNNs), which are also being used in a similar way, open up new forecasting possibilities by combining objective metrics with their interdependence, thus enabling a number of tasks like forecasting, anomaly detection and imputation (Jin et al., 2024). This is especially relevant for user engagement forecasting as it is the case when multiple behavioral indicators affect one another. Besides, transformer-based architectures—famed for their long-range dependency modeling—have been serviced time series forecasting with good performance (Miller et al., 2024). Very often, however, transformers do not come up to the performance level of simpler models, depending on the characteristics of the datasets (Kim et al., 2025).

The application of machine learning methods typically necessitates vast amounts of training data, high-performance computing, and less support with interpretable systems (Liu and Wang, 2024). These

drawbacks propel the administration of both statistic and deep learning together, resulting in hybrid models that gain the benefits of both sides.

2.6 Hybrid Models Forecasting

Hybrid models have been a source of the well-balanced solution to the problem of linearity and nonlinearity in the realms of statistical and ML models. Kong and co-workers proposed the hybrid Prophet-LSTM model which at first performs seasonality extraction via Prophet and later on trains the LSTM on the rest of the signal that is non-linear, coming up with accuracy improvements that are quite significant (Kong et al., 2021). In the same fashion Necula et al. showed that the combination of SARIMA with BiLSTM delivers a strong performance in difficult forecasting tasks (Necula et al., 2025).

2.7 Engagement Forecasting Contextual Factors

Stromer and co-workers talk about the necessity of forecasting systems that will consider the knowledge of the industry, the systems' non-expert-friendly explanation, and an incrementally interpretable non-expert sub-system (Stromer et al., 2024). These systems are perfect for Python applications where understanding by the stakeholders and practical interpretability are very important.

Liu and co-authors visualize users' actions in Snapchat using action graphs, achieving the predictive accuracy of the time series methods that are considerably decreased (Liu et al., 2019). As for Tang et al., their approach called FATE, GRAPHICAL, which consists of friendship relations, temporal factors, and behavioral signals using Graph Neural Networks and Long Short-Term Memory, produces engagement forecasts that are easy to interpret (Tang et al., 2020).

Li et al. describe the DIGMN model which adapts to the changing user intent dynamically, thus forecasting accuracy in LinkedIn and similar professional platforms is significantly increased (Li et al., 2023). The context of the factors like holidays and special events also plays a role in the engagement levels, and some studies have found that the incorporation of such factors improves the prediction accuracy (Kong et al., 2021).

2.8 Tools and Frameworks in Python

Python is to good for a whole range of time series forecasting tools and methods. Among the most favorite ones is Prophet due to its simplicity in dealing with seasonality and tolerance to lack of data (Gnanasekaran et al., 2023, Source, 2025). The more advanced ones are for example Temporal-GCN-LSTM, which infers user activity from action graphs (Liu et al., 2019). Developers, on the other hand, can make use of TimeGym that provides debugging utilities for forecasting pipelines, thereby assisting in spotting modeling issues pre-deployment (Seca, 2021). Streaming frameworks for instance that merge n-gram models with LSTMs can offer real-time systems with uninterrupted predictions (Bollig et al., 2024).

The Python in terms of supporting the scalable deployment architecture of Docker, BentoML, FastAPI, and Kubernetes, which allows for the creation of production-ready forecasting tools (Doroshenko et al., 2024). These stacks do not only make the forecasting solutions reusable and of a scale but also user-friendly.

2.9 Summary of Key Findings

Literature yields conventional models like ARIMA to provide useful baselines but at the same time, they are the ones who lose the fight with non-linear or very irregular engagement data. Deep learning models such as LSTMs, GNNs, and transformers show excellent performance but at the same time, they are pretty resource and data hungry. Hybrid frameworks come close to the ideal by giving more accuracy, interpretability, and robustness.

In addition, contextual features like holidays, intent, social relationships, and demographic variations, among others, contribute immensely to the accuracy of predictions. Contrary to the unscientific idea of history, the elephant in the room is that nothing like a super power [Python] forecast library exists, except for the drummed-up kind of self-incriminating propaganda of IPO hype on the digital market. Table 1 shows Literature review summary.

Table 1: LITERATURE REVIEW SUMMARY TABLE

Concept / Study	Key Finding	Study Limitations	Relevance to This Study
ARIMA (Classical Statistical Model)	The relationship between linear trends and seasonal effects in time series data (Bora, 2021, Hyndman and Athanasopoulos, 2018).	Find the behavior of applications from the nonlinear and bursty behaviors (Zemkoho, 2022, Zhang, 2003).	To be utilized as the baseline for comparison with the other sophisticated machine learning methodologies.
LSTM (Deep Learning Model)	Capable of learning the nonlinear and long-term interaction behaviors. (Liu and Wang, 2024).	High computational power and extra large datasets required under the stars. Black box behavior (Liu and Wang, 2024).	This design or analysis software is applicable that exhibits bursty, inconsistent user activity.
Hybrid Prophet–LSTM (Kong et al., 2021)	Prophets take care of seasonality; LSTM captures nonlinear residuals (Kong et al., 2021).	Two step training depends on cleaning process and data.	A robust option for the hybrid model suggested in this research
Hybrid SARIMA–BiLSTM (Necula et al., 2025)	The accuracy of the statistical + deep learning combination is greater than that of single models (Necula et al., 2025).	The experiments were mainly conducted on datasets that are not related to software.	The approach of hybrid models for Python engagement forecasting is supported.
Action-Graph Modeling (Liu et al., 2019)	The model depicts user behavior through sequence, and better results in than simply time series (Liu et al., 2019).	It requires detailed logging for each action; thus, it is more difficult to implement.	It is the reason for the inclusion of context variables like feature usage and event timelines.
Hybrid TSF Surveys (Miller et al., 2024)	Hybrid models considerably more often than single-model forecasting (Miller et al., 2024).	Based on surveys, limited validation & testing on usage logs.	Justifies the selection of a hybrid forecasting design.
Contextual Features (Holidays, Releases, Demographics)	The Introduction of Events in Addition to User Information Increases Precision (van Acker et al., 2023).	Additional metadata is necessary, which is not always accessible.	Allows for holiday indicators, publication schedules, and periods for the academic terms.
Prophet Forecasting Tool (Meta)	User friendly and very efficient for dealing with multiple seasonality and missing values (Gnanasekaran et al., 2023, Source, 2025).	It is difficult to manage nonlinear patterns of usage by oneself.	Considered as the trend/seasonality extraction phase of your hybrid model.
TimeGym (Debugging Framework)	Forecasting pipelines can be tested and verified by this system before available to the public (Seca, 2021).	Rather than to be deemed as a forecaster to software engineer, it, rather, serves as a troubleshooting tool.	Utilized to guarantee the quality and reliability of the pipeline.
Streaming Forecasting Frameworks	Real-time prediction is possible with n-grams together with LSTM networks (LSTMs) (Bollig et al., 2024).	Demanding logging of data in real-time; high cost of implementation.	Something which will provide the possibility of the realm of an extension for the real time foretelling capacity of a system in future.
Python Deployment Tools (Docker, FastAPI, BentoML)	The deployment of this model has now made it more reliable and more scalable (Doroshenko et al., 2024).	This requires competency in configuration and DevOps.	Besides relishing the privilege of Python-friendly collaborative prediction API assistances.

3. Research Objectives and Goals

Main Objective
Main Objective is through the usage logs; this study intends to build an accurate prediction model that forecasts user interaction with future Python-based software applications. This model will assist developers prepare for user interactions more strategically.

Specific Objectives
RO1: To explore, evaluate and experiment on some of the conventional time series forecasting models (ARIMA, SARIMA models) on data sets of usage logs in Python Applications.

RO2: At that point tried to compare how a more modern deep learning configuration (LSTM or GRU) could work on the same data, and that data is usually irregular and quite thin.

RO3: To start experimenting and streamlining hybrid adaptations of such models, Prophet-LSTM, ARIMA-GRU, combine statistical and machine-learning concepts to up accuracy levels.

RO4: Then, to incorporate some more low-hanging, specific to the topic features such as software version updates, academic calendars, or (public) holidays to understand to what degree these will be of any significance.

RO5: Lastly, composed a basic forecasting app that allowed developers to reuse their logs and receive not only predictions of relations but also certain visualizations. This was done entirely with FastAPI, TensorFlow, Prophet, and a few other Python packages.

These outcomes will provide the necessary resources and guidance needed from the research by Python developers facing challenges of engaging users in their software. Figure 1 shows how the Research Questions Mapping with Research Objectives and Goals.

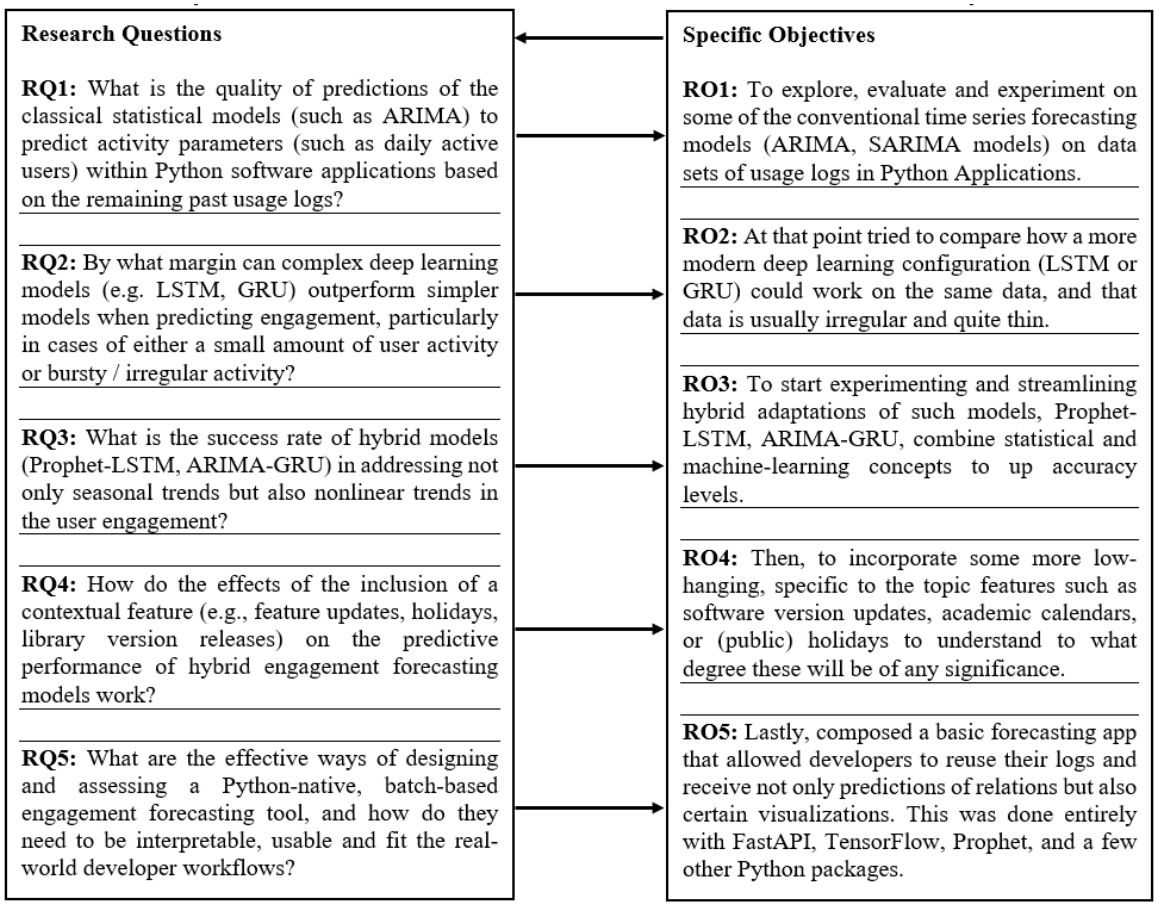

Figure 1: Mapping of Research Questions and Research Objectives

4. Methodology

The study follows a great structured, 7 phase methodology designed to develop, train, train and validate the model with Python Application usage logs and develop a forecasting framework with predicting user engagement. So below Figure 2 showing Methodology diagram. The research will follow a gradual method where each step will be based on the needs of anticipating users' interactions with software applications that have been made with Python. It focuses on the processing of domain-specific data as well as the creation of algorithms using the tools provided by the language Python which reflects as model building hybrid constructs, contextual modeling, and tool implementation. The entire process, or "workflow", as is often the case with software applications developed in Python, is characterized by fragmented, periodic, or burst-like patterns in user activity.

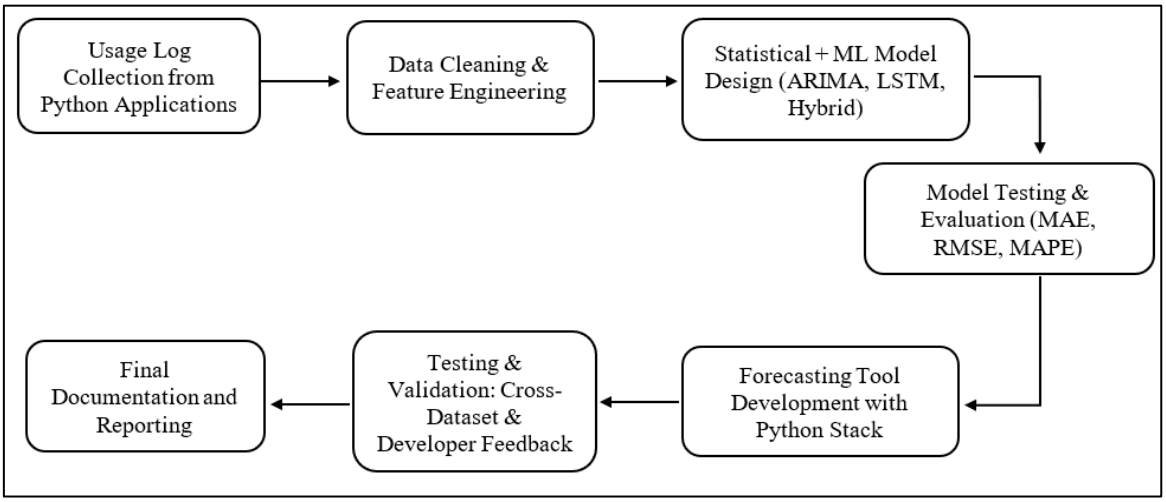

Figure 2: Methodology diagram

4.1 Usage Log Collection from Python Applications

The first step consists of gathering or creating user interaction data from the educational Python-based software tools, open-source utilities, and productivity scripting tools. Usage logs got from the Kaggle repository that were collected on actual behavior patterns in Python applications, with data variables like timestamps, ApplicationNames, DailyActiveUsers, SessionCount, AvgSessionDuration, LikeCount, ShareCount, DownloadCount, CommentCount, ErrorCount, DayofWeek, Month, IsWeekend and Release_Flag. No real logs will result in the production of datasets through Python utilities (such as Faker and Locust) creating phony situations of increased version engagement and academic period activity participation.

4.1 Data preprocessing and feature engineering

The raw usage log data cleaning with activities of remove duplicates, Missing values handling, Revisions for finer time-resolutions do not matter so much, Contextual features such as holidays, release cycles, flag features, etc., are added and feature changes validation. The analysis of visualization provided data metrics, correlations, and trends, finally managed outliers and done feature extractions. The cleaned data rows are aligned into daily time series format, while the contextual variables like releases cycles, version updates, and download counts are added to train and learn model and find real engagement dynamic changes. The logs will be cleaned and formatted into a time series that is more suitable for predictive analysis. With this technique, a dataset created for a particular domain, tailored to Python software will be assembled. This will specifically focus on capturing distinct trends and seasonality pertaining to the software.

4.2.1 Forecasting and Variable Design

This proposed research will help to forecast user interaction with Python software applications in the future based on organized usage log data. To fulfill this goal, it is necessary to accurately identify the dependent variable (forecasting), which in this case is a question of future user engagement, and the independent variables (supportive), which will be generated out of log data through feature engineering. Figure 3 shows the workflow diagram of Variables and Forecasting Structure.

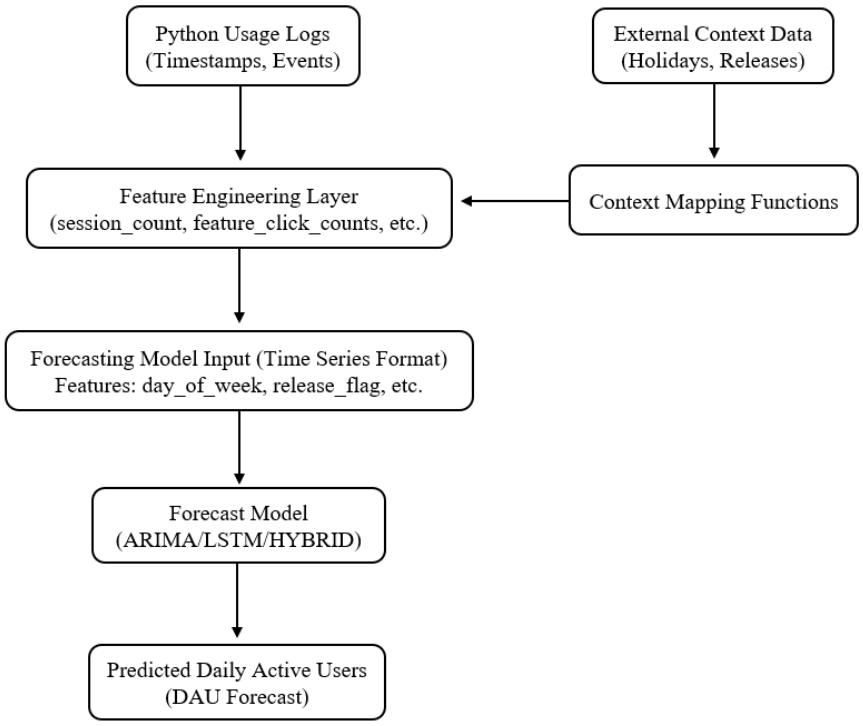

Figure 3 - Variable and Forecasting Structure Workflow

Dependent Variable (Forecasting Target)

- **Daily Active Users (DAU):** Daily Active Users, abbreviated as DAU, is the main metric of forecasting. This indicator captures the sum of unique people using the application in one day, and, therefore, works as an actual engagement measure. The total is the result of adding to a log element, either group-based or session-based on the day of a display or latent interaction or conversion.

Independent (Supportive) Variables

- The variables specified below Table 2 are important in determining the time frame as well as the explaining behavioral patterns and seasonal activators in any given forecasting model.

Table 2: Independent (Supportive) Variables

Category	Feature Name	Description
Temporal	**DayofWeek**	A well-established weekday marker (0-6) that would allow one to identify the weekly patterns of the usage of an application.
	Month, IsWeekend	The information incorporated in this research is across seasons and patterns in behavior that influence users.
Usage Metrics	**SessionCount**	The progress of the analysis also follows the amount of sessions that are initiated daily.
	AvgSessionDuration	Average time span of activities performed by a user during daytime.

	Feature_Click_Counts, LikeCount, ShareCount, CommentCount	The frequency of using specific features like, comment, share buttons in the course of time is also mentioned in the dataset.
	DownloadCount	The frequency of downloads on the day
Contextual / External	**Holiday_Flag**	It may be binary in nature to make a statement whether a day is a holiday or not.
	Release_Flag	Another variable is days after the introduction of new versions of software.
	Error_Count	Lastly, it is the count of user-affecting errors that are stipulated per day.

Note: The list of variables presented above will be viewed as the first baseline for model training. The feature set, however, is expected to change in the development phase depending on the availability of data, analysis of feature importance, or changing knowledge during experimentation. It is possible to add or modify the new features depending on how they affect the accuracy of the forecast.

4.3 Model Design, Development and Training

The research study is developed three model categories. ARIMA, SARIMA, and Prophet, the first statistical models, help in detecting linear trends with seasonality. The aims of detecting long-range trends and showcasing non-linear relations study develop and train deep models, like LSTMs and GRUs. The third type, the hybrid models, specifically Prophet-LSTM, ARIMA-GRU, and Prophet-ElasticNet, make use of trends and residual learning.

4.4 Model Evaluation and Testing

Developed models were evaluated using the MAE, RMSE, and MAPE, and they were also tested with standards, sparse and busty usage scenarios. Here for testing scenario-based preprocess and spike/low usage scenarios were considered for test robustness by cross-validation & visual representation for ensure generalization.

4.5 Forecasting Tool Development

A Python-based forecasting tool was developed using python framework Django, React + Vite, TensorFlow, REST-API, other python libraries Prophet-LSTM, and ARIMA-GRU models. Tool helps developers to upload logs.csv file, run models, and view forecasting results as Graph and table for accurate data driven engagement planning.

4.6 Tool Testing & Validation and Reporting

Testing and validation can be facilitated by leaving a different set of data in validation and crossvalidation and getting feedback from other peers. The end document explains the process in terms of modeling, testing, architectural detail of the tools, and the workflows.

5. RESULTS AND DISCUSSION

The results of the research study explain the clean performance of all three types of models. Also, the model performance evaluated using MAE, RMSE, and MAPE, and tested using scenario-based standard, sparse, and bursty engagement patterns. Final comparison of all the trained models is displayed in Table 3.

Table 3: FINAL MODEL PERFORMANCE SUMMARY

Model	Type	MAE	RMSE	MAPE (%)	Accuracy (%)
Prophet + LSTM	Hybrid	192.66	281.42	6.35	93.65
SARIMAX + GRU	Hybrid	236.68	298.63	8.53	91.47
ARIMA	Statistical	226.315	318.043	12.34	87.66
Prophet + ElasticNet	Hybrid	1792.44	3292.89	18.68	81.32
GRU	Deep Learning	349.3	478.13	32.96	67.04
LSTM	Deep Learning	420.08	623.47	34.90	65.10
Prophet	Statistical	474.533	888.112	37.40	62.60
SARIMA	Statistical	686.786	873.892	47.84	52.16

According to research questions and objectives,

5.1 Statistical Models Performance

Traditional statistical models demonstrate low and moderate accuracy. ARIMA got a MAPE of 12.34%, SARIMA 47.84% and Prophet 37.40%. So, this explains among the statistical models, ARIMA is suitable for stable and non-seasonal Python usage logs, but struggle with nonlinear spikes connect with releases and updates. So, this proves that proposed expectation of research that statistical models capture seasonality trends but not irregular patterns.

5.2 Deep Learning Models' Performance

Deep learning models performed less effectively. GRU got MAPE of 32.96% and LSTM got 34.90%. From these tests, it appears that the RNN models have the ability to identify long period patterns, but they require a lot of tuning and learning and use much larger data sets. Also, the sparse and bursty scenarios make the model overfitted. So according to RQ2, deep learning models don't overperform simple statistical models in low resource availability.

5.3 Hybrid Models Performance

Hybrid models clearly explain best results. The prophet + LSTM got high accuracy with a MAPE of 6.35% and SARIMAX + GRU followed 8.53%. This result ensures that research's expectation that Prophet/LSTM provide nonlinear residual learning and LSTM/GRU provides high prediction for engagement forecasting.

5.4 Contextual Features

The hybrid models recognize and manage the contextual variables, including holidays, launch times, and version updates. Both hybrid models demonstrated residual learning after contextual increment, supports research expectation that Python log patterns depend on domain specific temporal variables.

5.5 Framework Tool Design

The forecasting tool developed with best hybrid models, enable developers to upload usage logs and find forecasts through visual dashboards. Even with bursty data, with the help of peer reviews, scenario-based cross-validation shows robustness and strength in its performance. Figure 4 and Figure 5 show the framework tool's User Interface. And Figure 6 show the tool's results Bar charts for visualizing the models' performance metrics of all models with matrices.

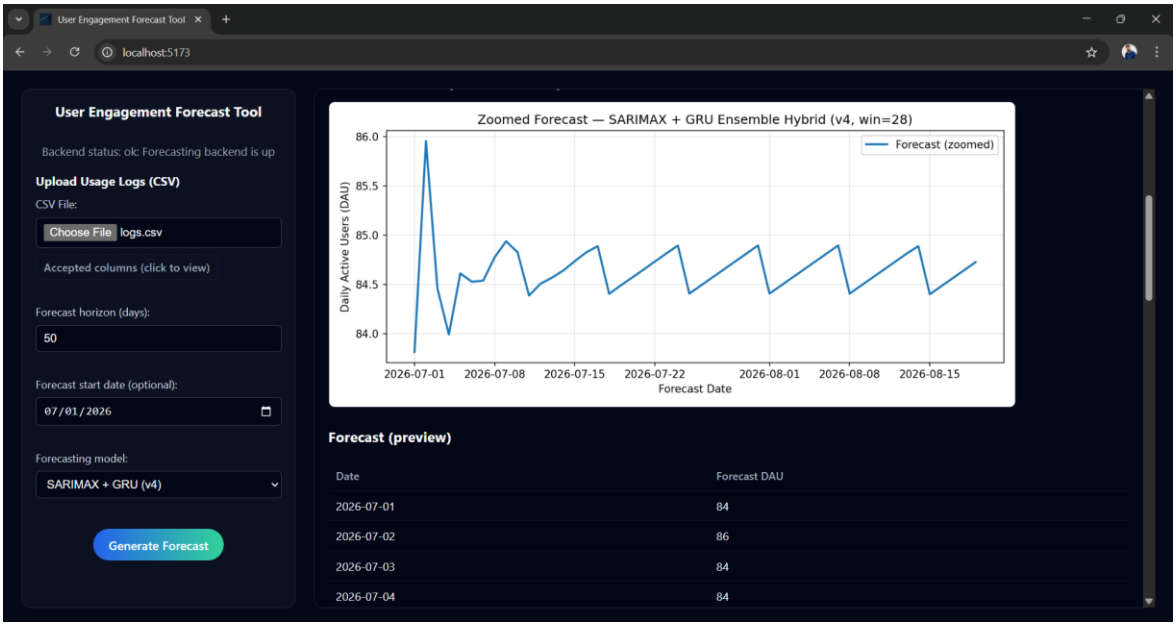

Figure 4: SARIMAX + GRU Forecast Result

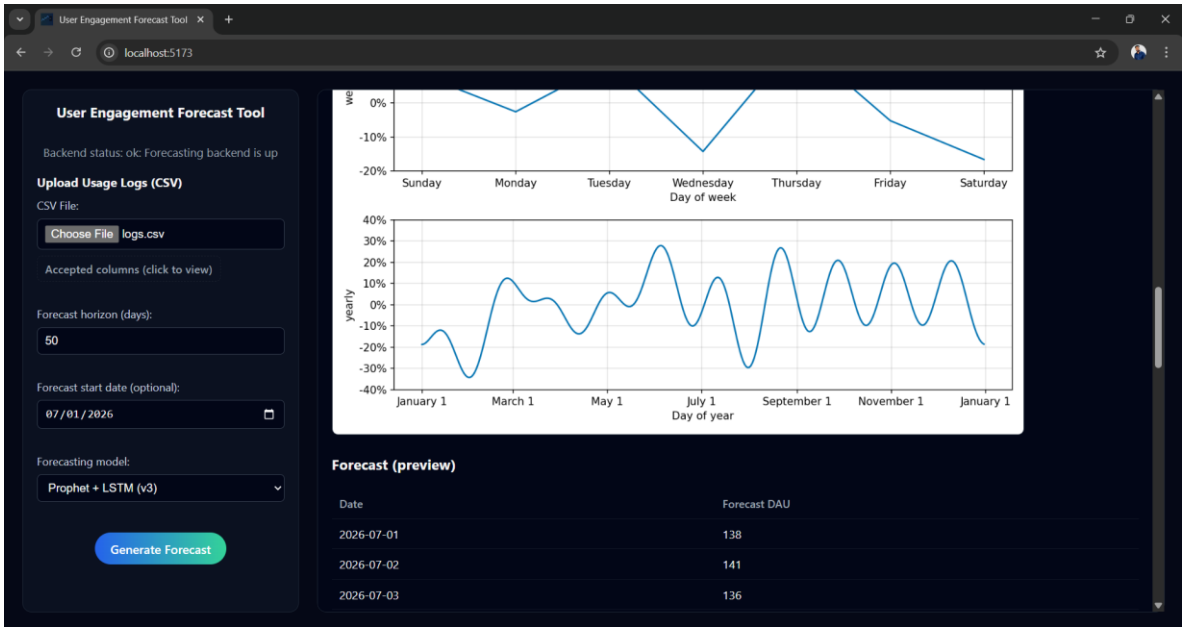

Figure 5: Prophet + LSTM Forecast Result

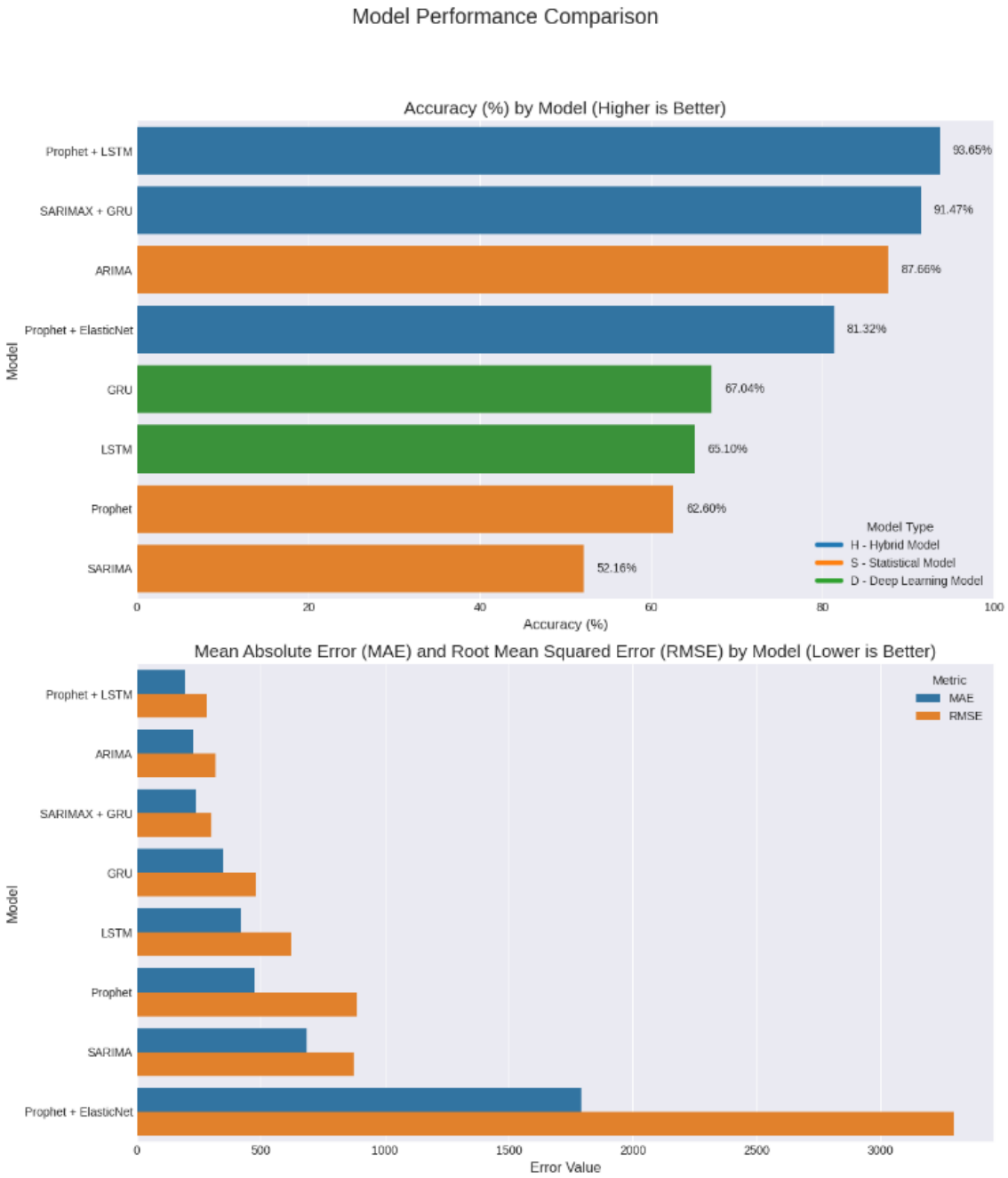

Figure 6: Bar charts for visualizing the models' performance metrics

6. IMPLICATIONS/CONCLUSIONS

This study illustrates that the hybrid time series forecasting models, especially Prophet–LSTM and SARIMAX–GRU, considerably surpass the traditional statistical and deep learning methods in predicting user engagement in Python software applications. The results have a significant impact on practice, theory, and policy. Practically, the outcomes of the study indicate that hybrid models can be the main assistance for software teams, developers, and product managers in comprehending the user engagement changes during the release cycles, feature updates, or even in the inactive periods. For the Software Engineering program and the university, research has asserted the need for incorporating data-driven forecasting, machine learning, and Python engineering skills in the curriculum. In a theory, the project provides a significant contribution by showing that the combination of trend-seasonality decomposition and nonlinear residual modeling could lead to a substantial increase in forecasting accuracy in cases of irregular and domain-specific engagement behavior.

The study presents various recommendations. Departments and colleagues can think of employing hybrid forecasting methods in course, end-year project, or research projects. The industrial companies might even encourage more developer-driven analytics projects through time series datasets and infrastructure provision. For the academic community at large, this research points out the necessity to investigate forecasting models in areas other than social media and finance, especially for open-source Python tools. Real-time forecasting support for the model could be one of the future paths of research, system evaluation based on a variety of applications, or incorporation of explainable AI techniques for better interpretability. Can carry on with this research by increasing the automation of the forecasting tool, adding the contextual variables, and fine-tuning the hybrid models with larger datasets.

Findings will be disseminated by presenting the results in academic forums, research sharing and updating internal teaching resources. As an example, the prediction tool can be taught at Software Engineering or Data Science laboratories, and examples of models can be used in assignments or exam assignments. It is possible to invite developers and teams to workshops to introduce hybrid forecasting models and Python-based tools. The evaluation of improvements will rely on the feedback of developers, checking the functionality of the tools on new data, and assessing the level of interaction with the workshops or training materials.

This study provided rich experience in sparse log processing, preprocessing of the data, hybrid modelling, evaluation methods, and development of the tools. Participators in trial of the tool also got introduced to actual forecasting processes and knew the relevance of contextual variables in predictive modelling. On an individual level, my understanding of methodological design and computational modelling has been increased. In future practice, focus more on the importance of early feature engineering, model interpretability, and a more user-friendly design of the tools because these approaches can make predictive systems more practical.

References

BOLLIG, B., FÜGGER, M. & NOWAK, T. 2024. *A Framework for Streaming Event-Log Prediction in Business Processes*.

BORA, N. 2021. Understanding ARIMA Models for Machine Learning. *Capital One* [Online]. Available from: https://www.capitalone.com/tech/machine-learning/understanding-arima-models/ [Accessed November 8, 2021 2021].

BROWNLEE, J. 2020. How to Make Out-of-Sample Forecasts with ARIMA in Python. *Time series forecasting* [Online]. Available: https://machinelearningmastery.com/make-sample-forecasts-arima-python/ [Accessed December 28, 2020].

DOROSHENKO, A. Y., HAIDUKEVYCH, Y. O., HAIDUKEVYCH, V. & ZHYRENKOV, O. S. 2024. User-centric technology stack for weather and air pollution forecasting. *PROBLEMS IN PROGRAMMING*.

GNANASEKARAN, H., P, D. & KÖSE, U. 2023. Time-series Forecasting of Web Traffic Using Prophet Machine Learning Model. 1**,** 161-177.

HYNDMAN, R. J. & ATHANASOPOULOS, G. 2018. *Forecasting: principles and practice*, OTexts.

JIN, M., KOH, H. Y., WEN, Q., ZAMBON, D., ALIPPI, C., WEBB, G. I., KING, I. & PAN, S. 2024. A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 46**,** 10466-10485.

KIM, J., KIM, H., KIM, H., LEE, D. & YOON, S. 2025. A comprehensive survey of deep learning for time series forecasting: architectural diversity and open challenges. *Artificial Intelligence Review,* 58**,** 216.

KONG, Y. H., LIM, K. Y. & CHIN, W. Y. Time Series Forecasting Using a Hybrid Prophet and Long Short-Term Memory Model. *In:* MOHAMED, A., YAP, B. W., ZAIN, J. M. & BERRY, M. W., eds. Soft Computing in Data Science, 2021// 2021 Singapore. Springer Singapore, 183-196.

LI, F., DU, L., FU, Q., HAN, S., DU, Y., LU, G. & LI, Z. 2023. DIGMN: Dynamic Intent Guided Meta Network for Differentiated User Engagement Forecasting in Online Professional Social Platforms. *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining.* Singapore, Singapore: Association for Computing Machinery.

LIU, X. & WANG, W. 2024. Deep Time Series Forecasting Models: A Comprehensive Survey. *Mathematics*.

LIU, Y., SHI, X., PIERCE, L. & REN, X. Characterizing and forecasting user engagement with in-app action graph: A case study of snapchat. Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019. 2023-2031.

MAKRIDAKIS, S., WHEELWRIGHT, S. & HYNDMAN, R. 1984. Forecasting: Methods and Applications.

MCKINNEY, W. 2010. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*.

MILLER, J. A., ALDOSARI, M., SAEED, F., BARNA, N. H., RANA, S., BUDAK ARPINAR, I. & LIU, N. 2024. A Survey of Deep Learning and Foundation Models for Time Series Forecasting. Available: https://ui.adsabs.harvard.edu/abs/2024arXiv240113912M [Accessed January 01, 2024].

MOUMANE, K., ASRI, I. E., RHAROUBI, I., ABDERRAHMAN, H. A. & FAQIHI, S. 2024. Enhancing banking governance: A machine learning-based credit risk classification. *Journal of Autonomous Intelligence*.

NECULA, S.-C., HAUER, I., FOTACHE, D. & HURBEAN, L. 2025. Advanced Hybrid Models for Air Pollution Forecasting: Combining SARIMA and BiLSTM Architectures. *Electronics,* 14**,** 549.

NG, K. W., MUBANG, F., HALL, L. O., SKVORETZ, J. & IAMNITCHI, A. 2023. Experimental evaluation of baselines for forecasting social media timeseries. *EPJ Data Science,* 12**,** 8.

SEABOLD, S. & PERKTOLD, J. 2010. Statsmodels: Econometric and Statistical Modeling with Python. *Proceedings of the 9th Python in Science Conference,* 2010.

SECA, D. 2021. TimeGym: Debugging for Time Series Modeling in Python. *arXiv preprint arXiv:2105.01404*.

SOURCE, F. O. 2025. *Prophet Python API* [Online]. Facebook: Facebook. Available: https://facebook.github.io/prophet/docs/quick_start.html [Accessed 2025].

STROMER, R., TRIEBE, O., ZANOCCO, C. & RAJAGOPAL, R. 2024. Designing forecasting software for forecast users: Empowering non-experts to create and understand their own forecasts. *ArXiv,* abs/2404.14575.

TANG, X., LIU, Y., SHAH, N., SHI, X., MITRA, P. & WANG, S. Knowing your fate: Friendship, action and temporal explanations for user engagement prediction on social apps. Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020. 2269-2279.

TUNNICLIFFE WILSON, G. 2016. Time Series Analysis: Forecasting and Control,5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung, 2015. Published by John Wiley and Sons Inc., Hoboken, New Jersey, pp. 712. ISBN: 978-1-118-67502-1. *Journal of Time Series Analysis,* 37**,** n/a-n/a.

VAN ACKER, J., MAENHOUT, L. & COMPERNOLLE, S. 2023. Older Adults' User Engagement With Mobile Health: A Systematic Review of Qualitative and Mixed-Methods Studies. *Innov Aging,* 7**,** igad007.

ZEMKOHO, A. A basic time series forecasting course with python. Operations Research Forum, 2022. Springer, 2.

ZHANG, G. P. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing,* 50**,** 159-175.