Time Series Forecasting of User Engagement in Python Software Applications Based on Usage Logs

V Janarthan[1]*, JDT Erandi[2], YMS Tharaka[3]
[1]*Department of Software Engineering, Faculty of Computing,*
[2]*Department of Data Science, Faculty of Computing,*
[3]*Department of Software Engineering, Faculty of Computing*
* janarthanvijayakumar@gmail.com

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. Therefore, research addresses this gap by developing a time series forecasting framework with use of Python application usage logs. This research study analyzes and compares three types of forecasting model like traditional statistical models (ARIMA, SARIMA and Prophet), deep learning models (LSTM and GRU), and hybrid models (Prophet-LSTM, SARIMAX-GRU and Prophet-ElasticNet), guided by research questions assessing by evaluating their forecasting performance 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 fewer 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 got high performance (MAPE are 6.35% and 8.53%) other than both statistical and deep learning models particularly under the sparse logs and bursty conditions. The research concludes that irregular features and hybrid residual prediction methods gave high forecast performance and offered valuable insights for Python developers and product teams.

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

Introduction

The motivation behind this research study arises from a current challenge found on software development among teams and clients that is understanding how software application users truly interact with Python-based applications and how these interactions are evolving with time series. The Python applications usage logs contain useful variables. Until now most existing analytical systems focus only on past analytics rather than future predictions and also, they only focused on single applications like Facebook, Instagram and TikTok. Such gaps are highly amplified under such circumstances as showing irregular, bursty, or sparse logs of Python applications or Python libraries when updating them or releasing a new version.

This motivates the development of time series forecasting methods that identify non-linear, seasonal, & Python-based engagement patterns. Statistical (ARIMA, SARIMA & Prophet), deep learning (LSTM & GRU) and hybrid models (Prophet-LSTM, SARIMAX-GRU and Prophet-ElasticNet) with neural networks show strong ways to develop accurate, context-related future predictions. This research focuses on comparing above models and finalizing high-performing models with Python usage logs and building a forecasting framework with Python-React frameworks and Libraries (Django + React Vite) combination by cooperating with contextual factors like release cycles, feature updates, and version updates to improve accuracy and operability.

Literature Review

The Literature Review of this research started from a theoretical concept of combining traditional time series modelling (ARIMA & SARIMA) and these models work with linear trends and seasonal expressions in data. ARIMA's mathematical idea gives base for analyzing predictable patterns within applications but struggles with nonlinear and irregular usage patterns (Zhang, 2003). To find solution to this limitation, the research adopts Recurrent Neural Network (RNN) theory, with Long Short-Term Memory (LSTM) networks that give nonlinear relationships capturing and temporal dependency capabilities that are important for irregular, bursty, or sparse logs (Chen et al., 2023). And some other studies strongly agree with hybrid modelling. Kong et.al (2021) explained that Prophet-LSTM combination increases forecasting performance by dividing seasonal structure from nonlinear dynamics, a principle set with this research's hybrid design (Kong et al., 2021).

Liu et al. (2019) demonstrated that user action and context changes like release periods, and version update cycles affect this modeling (Liu et al., 2019). Modern forecasts literature provides suggestions about hybrid and context-related model architectures that outperform standalone statistical and deep learning models, specifically in irregular scenarios (Miller et al., 2024). Finally, these theoretical ideas and literature findings are justifying that the research is focused on developing hybrid, context-integrated forecasting model and framework supported to Python application usage logs, also fill the methodological gap with accurate engagement prediction. Table 1 is displaying Literature Review Summary.

Table 1: Literature review summary table

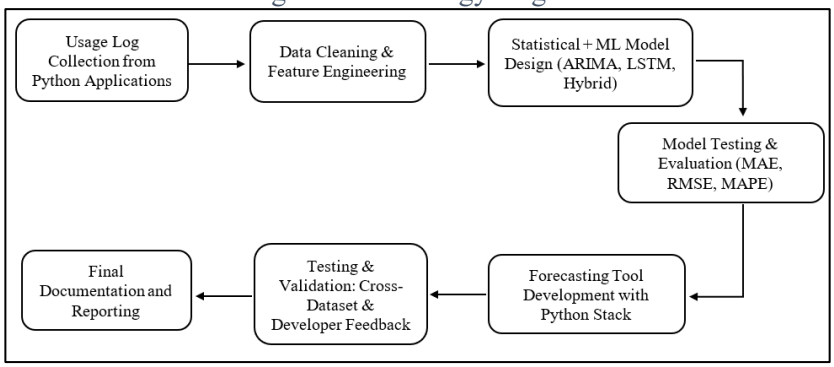
Concept	Purpose	Relevance to this Study
ARIMA	Finds seasonality & linear trends	Base statistical forecasting for stable engagement (Zhang, 2003)
LSTM	Nonlinear & long-term dependencies modelling	Handle irregular, bursty, or sparse logs by deep learning modelling (Chen et al., 2023)

Prophet-LSTM	Models separate nonlinear log data from trends	For the complex patterns improve performance (Kong et al., 2021)
Action-graph Behavior modelling	Context actions are used in sequential manner	Justifies the contextual variable integrations and modelling (Liu et al., 2019)
Hybrid TSF Surveys	Hybrid modelling has high performance.	It Supports selection of the hybrid frameworks (Miller et al., 2024)

Methodology

The study follows a structured, seven-phase methodology designed to develop, train and validate the model with Python Application usage logs and develop a forecasting framework for predicting user engagement. So below Fig. 1 showing Methodology diagram.

Fig. 1. Methodology diagram



1. Data Collection

The Kaggle repository is the place where the usage logs, which are the reflection of actual users' behavior in Python apps, come from. The data variables include TimeStamps, ApplicationName, DailyActiveUsers, SessionCount, AvgSessionDuration, LikeCount, ShareCount, DownloadCount, CommentCount, ErrorCount, DayofWeek, Month, IsWeekend and Release_Flag. The dataset comprises a bunch of datasets from Kaggle repositories. After filtering each source according to Python relevance, all datasets were brought up to the same schema and combined into one time-series dataset. The last dataset comprises a total of 18,479 day-wise records that range from 01 Aug 2012 to 03 May 2025, and the data is aggregated daily (one record for each application per date). Due to the absence of user-level identifiers, the "number of users" is indicated by DailyActiveUsers (DAU). The main variable to be engaged through interaction with the users and metrics related to errors is DailyActiveUsers, while contextual flags will be used as supporting features.

2. Data preprocessing and feature engineering

The dataset of raw usage log was cleansed by removing duplicates (none found) and treating missing values (1 missing like_count was substituted). Feature validity has been ensured. In addition, other calendar-related features (day_of_week, month, is_weekend, release_flag) were only used under confirmation. The distributions, correlations, and trends were examined using exploratory visualizations, and outliers were controlled. At last, records were arranged into a daily time-series data structure for the purpose of forecasting engagement dynamics.

3. Model Design, Development and Training

The research study developed three categories of models. ARIMA, SARIMA, and Prophet, the first statistical models, help in detecting linear trends with seasonality. The goal of deep learning modeling with LSTM and GRU is not only to detect long-range trends but also to

demonstrate the presence of non-linear relations, thus the study aims to develop and train deep learning models. The third type, the hybrid models, specifically Prophet-LSTM, SARIMAX-GRU, and Prophet-ElasticNet, make use of trends and residual learning. In the case of Prophet-LSTM and SARIMAX-GRU hybrids, they both utilized a 21-day window for inputs and 28-day forecast horizon, applied Adam optimizer, and used residual learning. The ARIMA/SARIMA models were applied using (p, d, q)/(P, D, Q, s) parameters. On the other hand, GRU and LSTM rely on windowed inputs along with a dropout regularization. Both Prophet and Prophet + ElasticNet utilized multiplicative seasonality and regressors.

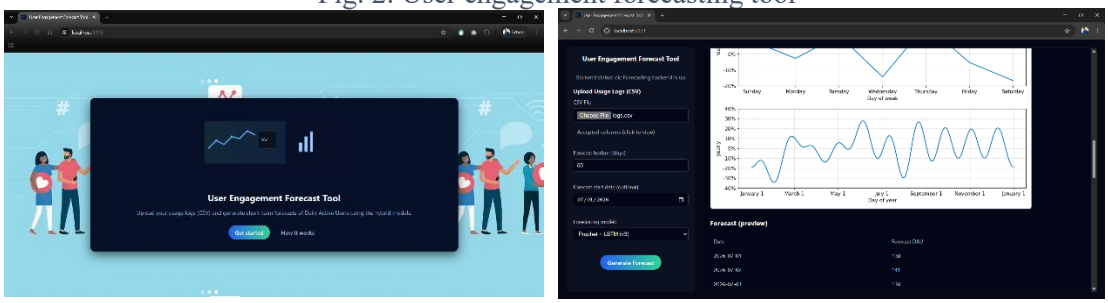
4. Model Evaluation and Testing

Models that have been developed were assessed through the use of MAE, RMSE, and MAPE (%). In order to prevent temporal leakage, a random split was avoided in favor of a chronological holdout where the performance was evaluated. Thus, the models were trained on the historical data and tested on the most recent forecast horizon (last 28 days). Moreover, the robustness of the model was tested by means of scenario-based testing with the three patterns being standard (baseline), sparse (low-usage), and bursty (spike-usage). In the process of verifying and confirming the properties of the model such as stability and generalization in different engagement dynamics, scenario-specific preprocessing (low-usage handling and spike control) and visual diagnostics were applied.

5. Forecasting Tool Development

A Python-based forecasting tool was developed using Python framework Django, React + Vite, TensorFlow, REST-API, and other Python libraries with Prophet-LSTM, and SARIMAX-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. Fig. 2 shows the framework tool's User Interface.

Fig. 2. User engagement forecasting tool



6. Tool Testing & Validation

Tool Testing and validation were carried out using chronological data splits and scenario tests (baseline, sparse, and bursty) to maintain the order of the data and avoid information leakage. Furthermore, peer reviews by Python programmers were used for assessments of usability and interpretability of results and relevance for forecasting.

Results and Discussion

The results of the research study explain the overall 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 2. According to research questions and objectives,

Statistical Models Performance: Traditional statistical models demonstrate low and moderate forecasting performance. ARIMA got a MAPE of 12.34%, SARIMA 47.84% and Prophet

37.40%. Therefore, explains among the statistical models, ARIMA is suitable for stable and non-seasonal Python usage logs, but struggles with nonlinear spikes connect with releases and updates. Therefore, proves that proposed expectation of research statistical models capture seasonality trends but not irregular patterns.

Performance of Deep Learning Models: 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 datasets. Also, the sparse and bursty scenarios make the model overfit. So deep learning models don't overperform simple statistical models in low resource availability.

Hybrid Models Performance: Hybrid models clearly explain best results. Prophet-LSTM got high forecast performance with an 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 SARIMAX /GRU provides high prediction for engagement forecasting.

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

Framework Tool Design: The forecasting tool developed with best hybrid models, enables developers to upload usage logs and find forecasts through visual dashboards. Even with bursty data, with the help of peer reviews, scenario-based evaluation shows robustness and strength in its performance.

Table 2: Final Model Performance Summary

	Model	Type	MAE	RMSE	MAPE
1	Prophet-LSTM	Hybrid	192.66	281.42	6.35%
2	SARIMAX-GRU	Hybrid	236.68	298.63	8.53%
3	ARIMA	Statistical	226.315	318.043	12.34%
4	Prophet + ElasticNet	Hybrid	1792.44	3292.89	18.68%
5	GRU	Deep	349.3	478.13	32.96%
6	LSTM	Deep	420.08	623.47	34.90%
7	Prophet	Statistical	474.533	888.112	37.40%
8	SARIMA	Statistical	686.786	873.892	47.84%

The statistical significance in the performance of forecasting was evaluated using the Wilcoxon signed-rank test on paired absolute errors from the last 28-day forecast horizon. The test showed that the combination of Prophet and LSTM gave a significantly better result over the SARIMAX and GRU combination ($p = 3.67 \times 10^{-6}$), thus verifying the claimed improvements to be statistically valid.

Implications/Conclusions

This research explained that hybrid time series approaches are performed better than statistical and deep learning models on forecasting Python application user engagement. Results help in both theoretical and applicative aspects, explaining the importance of trend and seasonality models in the context of remaining residuals with a non-linear pattern. Results aid the developers in adjusting according to the data-driven vision of future resource allocation and user behavior predictions. The future scope of the tool helps in expanding the hybrid model for real-time forecasting on a larger dataset with multiple applications. Sharing the findings can

help to improve the forecasting tool further. Key lessons learned are advanced modelling, handling sparse logs, statistical data management, design interpretation, Machine learning & Deep learning and domain specific forecasting tools.

Artificial Intelligence Disclosure

During the generation of this extended abstract document, author used Grammarly and ChatGPT (GPT-5.1) for refining words, grammar checks, and generating summary tables. Also, these tools align the contents with COMURS format guidelines. All AI-used contents were carefully reviewed, edited, and validated by the author, who has all the responsibilities for accuracy, integrity and final presentation.

References

Chen, Z., Ma, M., Li, T., Wang, H., & Li, C. (2023). Long sequence time-series forecasting with deep learning: A survey. *Inf. Fusion*, *97*, 101819.

Kong, Y. H., Lim, K. Y., & Chin, W. Y. (2021, 2021//). Time Series Forecasting Using a Hybrid Prophet and Long Short-Term Memory Model. Soft Computing in Data Science, Singapore.

Liu, Y., Shi, X., Pierce, L., & Ren, X. (2019). 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,

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. arXiv:2401.13912. Retrieved January 01, 2024, from https://ui.adsabs.harvard.edu/abs/2024arXiv240113912M

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159-175. https://doi.org/https://doi.org/10.1016/S0925-2312(01)00702-0