User Behavior Analysis in Mobile Applications Using Machine Learning Models for Fault and Resource Consumption Prediction

Fahad Ahmed Shaban[1], Hussam Ahmed Rammo[2], Ali Khairi Altoohafi[1], Zaid Jasim Al-Araji[1] and Shamil Khudhur Ramadhan[2]

[1]*Department of Computer Networks and Internet, College of Information Technology, Ninevah University, 41002 Mosul, Iraq*

[2]*Techniques Engineering College for Computer and AI, Northern Technical University, 41002 Mosul, Iraq*

fahad.ahmad@uoninevah.edu.iq, hussam.rammo@ntu.edu.iq, ali.altoohafi@uoninevah.edu.iq, zaid.jasim@uoninevah.edu.iq, shamil.ramadhan@ntu.edu.iq

Keywords: User Behaviour, Mobile Applications, Fault Prediction, Resource Usage, Machine Learning, Anomaly Detection.

Abstract: The research paper presents a framework which uses machine learning to study user behavior and predict mobile application faults and resource use. Researchers aim to achieve two goals through their work which includes predicting application failures and determining system resource needs using user interaction data and system operational data and contextual information. The framework uses three feature extraction methods which include statistical and temporal and sequence-based approaches to create models that show how applications function through both session-based and time-based data. The study evaluates four classification models which include Random Forest, XGBoost, Multi-Layer Perceptron (MLP), and LSTM for their ability to predict faults. For resource consumption forecasting, the study applies regression and sequence models which include Random Forest Regressor, XGBoost, and LSTM. Experimental results on a real-world mobile telemetry dataset show that XGBoost achieves the best fault prediction performance with an accuracy of 0.94 and a precision of 0.82 and a recall of 0.75 and an F1-score of 0.78 and an ROC-AUC of 0.93. LSTM model achieves better resource prediction results than other models by achieving 2.3% RMSE and 0.90 R^2 for battery prediction and 8.0% RMSE with 0.88 R^2 for CPU utilization. The research results demonstrate that using behavioral features together with telemetry data and contextual information leads to better prediction accuracy. The proposed framework offers a practical method which helps to improve mobile application reliability and efficient resource management and real-time system deployment.

1 INTRODUCTION

People use mobile apps for their everyday activities because these apps include social media platforms and productivity software and healthcare applications and entertainment services and educational tools. Users expect their applications to deliver smooth and reliable and responsive performance under all operating conditions. The first problem occurs when faults and crashes and anomalies happen. Faults and Crashes and Anomalies The unexpected behaviors that they show cause application execution to stop which results in a negative user experience and makes the service either partially or completely unavailable. Mobile applications face their most critical reliability challenges through application crashes and runtime exceptions and freezing problems which lead to unresponsive behavior according to the study's findings [1]. Previous studies show that these failures reduce user satisfaction which has led to the development of predictive models that detect crash-prone scenarios before software deployment [2]. Second, Resource Consumption Problems. Mobile applications demonstrate unpredictable resource usage patterns because of their CPU and memory requirements as well as their battery life and network connectivity needs, which lead to decreased device functionality and user experience quality. Large-scale empirical studies confirm that inefficient resource utilization and excessive energy consumption are common performance issues in modern mobile applications [3], [4].

The problems emerge through different pathways which depend on how users act and how application sessions evolve and how device hardware functions

and how operating systems perform. The application environment includes two main parts which determine network connectivity and the application status of either foreground or background operation [5]. The capability to anticipate upcoming problems enables developers to reduce system resource consumption through three different methods which include visual or computational workload reduction and non-critical task deferral and direct user notifications and the automatic adjustment of scheduling policies to sustain optimal system performance and reliability [1], [2]. The Mobile Application Performance Management with Machine Learning is defined along with the utilization of various methodologies [6]. The system processes application log files by determining the importance of three primary data sources which include user behavior logs that contain taps swipes screen transitions and session characteristics together with system telemetry data that tracks CPU usage memory consumption and battery status and network performance over time and contextual metadata which includes device model information and OS version details and network type data and location data [7], [8].

The current state of mobile application analytics has developed through major advancements yet existing research studies face multiple research limitations. The existing research studies primarily investigate two distinct areas of study which include fault prediction and resource consumption modeling while failing to create a comprehensive framework that combines both research domains. The study methods used by researchers depend on basic features which automatically disregard how user behavior and system telemetry and contextual metadata work together. The current model design approach prevents systems from understanding complex relationships between user actions and system performance changes.

The current research studies focus on offline data analysis methods which do not address the real-time operational requirements that demand assessment of processing power and energy use and system response to changing user patterns. The existing system limitations require development of a complete system that operates at scale by integrating multiple data sources while delivering both precise predictions and operational deployment capabilities.

The proposed unified machine learning framework addresses these challenges by using integrated behavioral telemetry and contextual features to create a model that predicts faults and forecasts resource usage. The proposed framework differs from existing methods by performing systematic evaluations of traditional machine learning and deep learning models through classification and time-series prediction tests which include deployment-aware testing. The integrated perspective enables mobile application performance prediction through real-world environment testing which produces more accurate and robust and practical results.

This work contributes mainly in the following ways:

- Feature Extraction Pipeline. The system extracts four types of features through mobile application session data by using a comprehensive feature extraction technique. The system enables complete session dynamic representation together with user interaction pattern display and device feature demonstration and execution context display, which research shows is essential for better mobile environment prediction accuracy.
- Comparative Model Evaluation. The research evaluates different machine learning models through its examination of traditional tree-based techniques which include Random Forests and Gradient Boosting and deep learning models for both classification and regression tasks. The comparative evaluation process evaluates three factors which include model complexity and prediction accuracy and runtime costs for mobile application analytics.
- Feature Importance Insights. We use feature importance analysis to determine which features most accurately predict application faults and resource usage. The study outcomes will provide guidance for creating models and executing optimization techniques which involve particular performance adjustments and energy efficiency scheduling.
- DeploymentConsiderations. We examine how predictive models function in mobile settings by testing their actual deployment. The study needs these factors to determine which deployment method, on-device or hybrid edge/cloud, will most effectively maintain user experience.

The research shows that big data machine learning methods can be used to create predictive systems which enhance the quality and efficiency and user experience of mobile applications and services.

2 RELATED WORKS

Recent literature from 2022 to 2025 has extensively explored fault prediction, resource and energy consumption forecasting, and user behavior modeling

in mobile applications. Jorayeva et al. [9] conducted a comprehensive systematic literature review for defect and fault prediction and discovered that existing studies used supervised machine learning techniques in approximately 92% of cases while targeting Android applications and object-oriented metrics and deep learning had limited adoption despite its ability to generalize across different projects. Jorayeva et al. [10] developed deep learning defect prediction models through CNN and LSTM architecture design which successfully predicted defects in Android applications for both single project and multiple project scenarios. Wimalasooriya et al. [11] created a Just-in-Time crash prediction system that uses partial session telemetry together with user behavior data to forecast upcoming application crashes which operates in accordance with proactive fault prediction and user-aware system adaptation.

Scientists have used predictive modeling to study energy resource usage and battery life estimation, which helps them create more energy-efficient mobile devices. The research study presented in [12] develops a deep learning system that creates customized battery consumption models, which use data about user application usage and network behavior and device attributes to enhance battery life prediction outcomes. The research studies [13] and [14] examined machine learning methods to predict battery remaining useful life (RUL), which involved testing different algorithms and feature sets to assess battery performance decline throughout time. The development of energy-aware mobile applications has been solved through the application of metaheuristic-based machine learning techniques, which help with both feature selection and hyperparameter optimization tasks according to the research in [15]. The research work EPAM [16] created a mobile AI workload predictive energy model, which uses Gaussian Process Regression to estimate energy consumption, latency time, and memory requirements in order to optimize mobile hardware performance for AI applications.

At the same time, people have started to pay more attention to both anomaly detection and user behavior modeling. The researchers in [17] developed probabilistic deep learning models which used mixture density networks to achieve two goals of recognizing multiple user behavior patterns and identifying unusual activities. The researchers in [18] used changepoint analysis together with machine learning techniques to identify smartphone power usage patterns that indicated abnormal device operation. The study in [19] investigated how autoencoder-based enhancements of spatio-temporal interaction features could detect suspicious user activities while proving that time-based and situation-based analysis remains crucial for mobile data examination, as shown in Table 1.

Table 1: Overview of existing approaches and their limitations in mobile application predictive modeling.

Ref.	Focus Area	Main Contribution	Key Limitations
[9]	Fault Prediction	Systematic review of ML-based defect prediction in mobile apps	Limited deep learning usage; focus mainly on Android and OO metrics
[10]	Fault Prediction (DL)	CNN/LSTM models for intra- and cross-project defect prediction	High computational cost; limited runtime deployment discussion
[11]	Crash Prediction	Just-in-Time crash prediction using partial session data	Requires continuous telemetry; potential overhead
[12]	Battery Prediction	Personalized DL battery consumption models	User-specific training required; privacy concerns
[13]	Battery RUL	ML-based estimation of battery remaining life	Limited generalization across devices
[14]	Battery RUL	Comparative study of ML models for battery prognostics	Offline evaluation; no real-time adaptation
[15]	Energy Prediction	Metaheuristic-based feature selection for energy modeling	Increased model complexity
[16]	Mobile AI Energy	Predictive energy model for AI workloads on mobile devices	Focused mainly on AI inference tasks
[17]	User Behavior / Anomaly	Probabilistic DL models for behavior-based anomaly detection	Model interpretability challenges
[18]	Anomaly Detection	Changepoint and ML-based smartphone anomaly detection	Relies mainly on power signals
[19]	Anomaly Detection	Spatio-temporal autoencoder-based behavior modeling	High data dimensionality

Mobile application analytics have experienced significant advancements. However, important research gaps still exist in the existing literature. First, most prior studies focus on a single task, such as fault prediction [9]-[11] or resource and energy consumption modeling [12]-[16], without providing a unified framework that jointly addresses both problems. The isolation of application performance and system performance assessment hinders researchers from understanding how application performance influences system performance.

The various methods depend on restricted feature representations because they focus on software metrics or system-level telemetry measurements which fail to capture the complete effect that user behavior combined with contextual information and temporal dynamics has on system performance. The research in studies [17]-[19] shows that including behavioral and spatio-temporal features helps detect anomalies more effectively yet these features do not appear in every task implementation.

Third, multiple deep learning-based techniques achieve strong predictive results yet create problems which stem from their high computational demands and their low ability to be interpreted and their limited capability to be used in real-time situations. The defect prediction models which use CNN and LSTM architectures from [10] require extensive computational power, which makes them unsuitable for on-device inference in environments with limited resources.

Researchers have paid insufficient attention to deployment-aware factors which include energy efficiency and latency and privacy protection because these factors are essential for real-world mobile application deployment.

The proposed framework establishes its unique system because it contains two specific elements which do not exist in current systems. The system uses unified operation to predict faults and estimate resource usage through its dual operational framework. The system uses unified operation to predict faults and estimate resource usage through its dual operational framework. The research team evaluated all machine learning and deep learning methods by testing their performance under actual deployment conditions. The method combines two different elements which make it distinct from earlier methods because it handles both field deployment and analytical work with mobile applications.

3 METHODOLOGY

Below Simplified is the practical methodology that you need for real observation in the present study:

3.1 Data Collection

The procedure requires you to gather actual datasets from mobile applications which have numerous users throughout an extended timeframe of several weeks or months. The system is required to document every user interface interaction which includes all common UI actions and the times when users start or end their sessions and when they toggle their application between active and inactive modes. The system needs to track three different types of system telemetry which include CPU usage and memory usage and battery status and battery discharge rate and network data usage and storage input and output measurement and thread processing capacity and potential GPU resource allocation. It might be sampled at regular periods (for instance, 1s or few seconds).

The system requires contextual features which include time of day, day of week, device model, OS version, network type (Wi-Fi, 4G/5G), signal strength, location/GPS (if allowed), charging status, screen brightness, temperature. The system needs to establish faults through the following indicators which include crash logs and exceptions and freeze reports and threshold violations (CPU > threshold and memory leak) and abnormal battery drop. User feedback will enable the system to detect anomalies which occur at an unspecified rate. The process of session demarcation requires the establishment of session boundaries which encompass all application activities from the initial launch until the user closes the application or becomes inactive or through the application of sliding time window methods, as shown in Figure 1.

3.2 Feature Engineering

The framework starts with feature engineering as its primary first step which enables engineers to create predictive models from mobile application raw logs and telemetry data. The system starts its operation by processing complete application session data through its initial stage which needs both total session data and specified time duration information to retrieve execution patterns that exist throughout short and

extended time periods. The system generates statistical descriptors which include mean and maximum and minimum and standard deviation and percentile values for CPU usage and memory usage and battery drain and network activity from the collected data. The resource usage statistics provide short summaries which help users recognize both standard application processes and uncommon application processes.

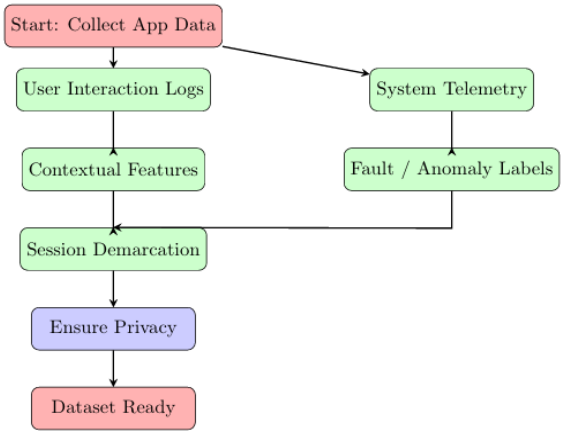

Figure 1: Data collection methodology for mobile applications.

Mobile telemetry data is transformed through feature engineering process to create structured data formats which can be used for predictive modeling purposes, as shown in Figure 2. The process divides the data into time windows of constant duration which enables the extraction of statistical features that include mean value and standard deviation and minimum value and maximum value:

$$fstat = \{\mu, \sigma, min, max\}. \quad (1)$$

The user interaction features of the system use session duration and event rate and inter-event time as their measurement metrics while the system's temporal features of time-of-day and moving averages work to identify recurring patterns and performance patterns throughout the day.

All features are normalized as:

$$x' = \frac{x-\mu}{\sigma} \quad (2)$$

The feature space with PCA can be made simpler by removing some values dimensions.

The evaluation process employs system-level metrics together with user interaction data to assess user application behavior. The application usage assessment process demands users to provide data about their screen transitions and tap activities and event creation and total session time and inter-event time distribution patterns. The research study includes time-related elements because they demonstrate their impact on both application performance and resource usage. The system includes time-of-day indicators which allow users to see their current time together with weekday versus weekend distinctions and execution duration measurements and moving averages of resource metrics and trend-based features which include battery level and CPU usage slope trends to enable the model to learn periodic patterns and gradual performance degradation.

The model is trained by minimizing the following objective function:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, \hat{y}_i) + \lambda\left|\left|\theta\right|\right|^2 \quad (3)$$

The system metrics and interaction events from mobile telemetry are treated as time series data because of their continuous flow and progression through time. The system uses sliding window technology to create fault prediction features and upcoming resource consumption forecasting features from N seconds or minutes of current historical data. The system extracts features from each window by measuring change rates and short-term volatility and extracting frequency-domain characteristics while using sequence representations or embeddings to maintain temporal dependencies in applicable cases.

The feature set includes execution environment details which are required for understanding the contextual information of the system. The system records four types of information which include the device model, operating system version, network type, and application state as foreground or background. The encoding process of categorical context variables utilizes encoding schemes or embeddings, while continuous context variables need normalization to achieve stable numerical values and feature scaling throughout their battery temperature and signal strength measurements.

Multiple data preprocessing methods begin the process which leads to better feature quality and improved model performance. The system manages missing values through two methods which include imputation and value removal based on the frequency of missing data. The system utilizes feature scaling and normalization methods to transform value ranges into a standardized measurement system. The system uses dimensionality reduction methods which include principal component analysis and feature selection techniques to decrease model complexity while preserving prediction accuracy in high-dimensional feature spaces.

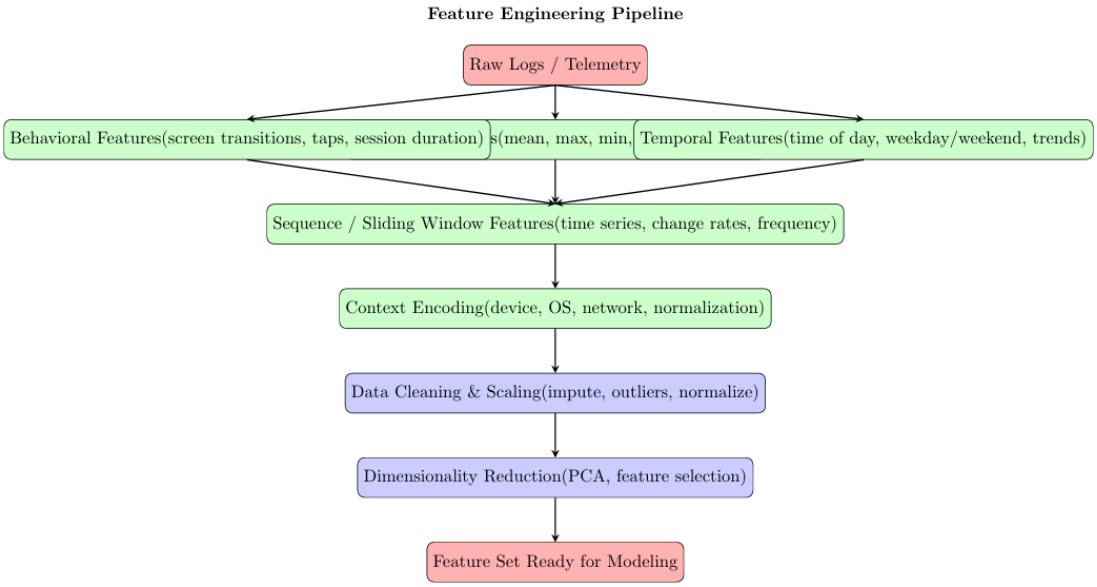

Figure 2: Feature engineering pipeline for mobile application data.

3.3 Modeling Tasks

3.3.1 Fault / Anomaly Prediction

The proposed framework addresses two complementary prediction tasks. The first aims to determine whether a software fault will occur during the current user session, while the second predicts potential failures in subsequent sessions based on historical user behavior and system telemetry. Early and accurate fault prediction enables preventive actions that improve application reliability and user experience.

Both supervised and unsupervised machine learning approaches are considered. Supervised classification models, including Random Forest (RF), Extreme Gradient Boosting (XGBoost), Support Vector Machines (SVM), and Neural Networks, are trained using labeled session data. The prediction function is defined as:

$$\hat{y} = f(X; \theta), \tag{4}$$

where X represents the feature vector and θ denotes model parameters.

To identify previously unseen or rare failures, unsupervised anomaly detection techniques are also employed, including Autoencoders, One-Class SVM, and Mixture Density Networks (MDNs). In reconstruction-based methods, the anomaly score is computed as:

$$e = ||X - \hat{X}||^2. \tag{5}$$

Building upon these formulations, the fault-prediction process is organized into two complementary operational approaches.

The first approach, early within-session prediction, analyzes only the initial portion of an active session together with runtime telemetry to estimate whether a fault is likely to occur later during the same session:

$$P(y = 1|X_{1:N}). \tag{6}$$

The second approach, offline fault detection, analyzes completed historical sessions to identify faulty or anomalous executions that can subsequently be used for diagnostics, model retraining, and long-term reliability assessment:

$$\acute{Y} = f(X; \theta). \tag{7}$$

One of the primary challenges in fault prediction is the severe class imbalance, since faulty sessions typically represent only a small fraction of the collected data. To address this issue, both data-level techniques (e.g., oversampling with SMOTE and undersampling) and algorithm-level methods, including cost-sensitive learning, are considered. In addition, anomaly-detection models naturally support learning from predominantly normal data without requiring balanced datasets.

Model performance is evaluated using standard classification metrics, including Accuracy, Precision, Recall, F1-score, and the Area Under the Receiver Operating Characteristic Curve (ROC-AUC). To better reflect practical deployment requirements, additional metrics such as Precision@K and Recall at a fixed false-positive rate may also be reported.

3.3.2 Resource Consumption Prediction

3.3.2.1 Modeling Techniques

Resource consumption is modeled using both regression and time-series forecasting techniques.

Ensemble learning methods, including Random Forest Regression and XGBoost Regressor, capture complex nonlinear relationships between application features and resource utilization. For example, processor and memory utilization can be predicted several seconds ahead, enabling dynamic CPU scheduling and adaptive memory allocation.

Temporal prediction is performed using classical statistical models such as ARIMA and Prophet together with deep learning architectures including Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Temporal Convolutional Networks (TCNs). These models exploit temporal dependencies in historical telemetry to forecast future resource usage. For instance, battery consumption during video streaming can be predicted several minutes ahead, allowing adaptive adjustment of display brightness, refresh rate, and network activity.

The models are trained using historical data collected up to October 2023. Training features include user interaction logs, session activity records, and historical system telemetry. Hybrid feature sets combining interaction behavior (e.g., tap frequency and scrolling patterns) with system-level metrics (e.g., CPU utilization and memory usage) improve the prediction of resource spikes during periods of intensive user activity.

3.3.2.2 Forecasting Horizon

Short-term forecasting, ranging from several seconds to a few minutes, supports real-time optimization of application performance. Predicted CPU utilization enables proactive task scheduling, adaptive background-service management, and dynamic resource allocation before performance degradation becomes noticeable.

Long-term forecasting extends over several minutes and primarily supports energy management and resource planning. For example, predicting battery consumption during the next 15 minutes enables intelligent control of background applications, adaptive display refresh rates, and timely power-saving recommendations.

3.4 Experimental Setup

The experimental setup is designed to ensure reproducibility and objective comparison of different prediction models. It consists of dataset partitioning, model configuration, evaluation metrics, and implementation details.

3.4.1 Dataset Splitting Strategy

To prevent information leakage, the dataset is partitioned at the session level, ensuring that all samples belonging to the same session are assigned exclusively to either the training or testing subset. K-fold cross-validation is applied whenever appropriate to improve the robustness of performance estimates. For time-dependent experiments, chronological splitting is adopted so that training data always precede testing data, thereby preserving the temporal characteristics of the prediction problem.

3.4.2 Model Configuration

The experimental evaluation includes both machine learning and deep learning models for classification and regression tasks. The classification models comprise Random Forest, XGBoost, Multi-Layer Perceptron (MLP), and Long Short-Term Memory (LSTM) networks, whereas the regression models include Linear Regression, Random Forest Regressor, XGBoost Regressor, and LSTM. Model hyperparameters are selected using established tuning procedures that balance predictive performance with computational efficiency.

3.4.3 Evaluation Metrics

Classification performance is assessed using Accuracy, Precision, Recall, F1-score, and ROC-AUC. Since faulty sessions represent a minority class, particular emphasis is placed on Precision, Recall, and F1-score to provide a balanced assessment of detection capability.

Regression performance is evaluated using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). RMSE penalizes large prediction errors more heavily, MAE provides an interpretable measure of average prediction error, and R^2 quantifies the proportion of variance in resource consumption explained by the prediction model.

3.4.4 Implementation Details

All models are implemented using widely adopted machine learning frameworks and libraries. Data preprocessing includes feature normalization, handling of missing values, and categorical feature encoding. To mitigate class imbalance during fault prediction, oversampling and cost-sensitive learning techniques are incorporated where appropriate. All experiments are conducted under identical

computational settings to ensure fair and reproducible comparison between the evaluated models.

3.5 Deployment Considerations

The deployment process for predictive models in mobile applications needs to address multiple practical aspects to achieve operational efficiency and system reliability and user trustworthiness. The main decision centers on choosing between on-device and server/cloud-based inference systems. On-device models provide users with quick response times for their applications while functioning without internet access but require developers to create energy-efficient models that use minimal system resources. Server-based inference enables the use of advanced models which deliver superior accuracy results but creates extra delays through network connections while raising potential privacy risks.

The highest priority should be given to privacy requirements and ethical standards. Organizations must collect data in a way that protects user privacy rights by preventing access to sensitive data through anonymous or obfuscated methods. The federation learning method enables model development on multiple devices while protecting user data through safe privacy-compliant training methods which maintain predictive accuracy.

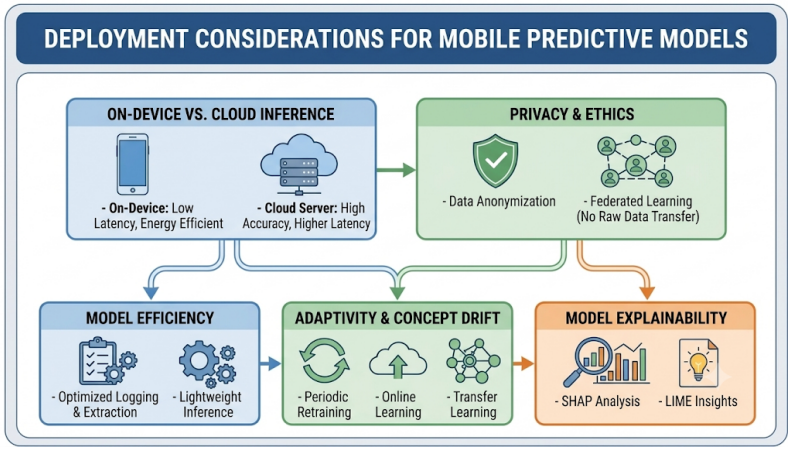

Figure 3: Conceptual framework for deploying predictive models in mobile applications.

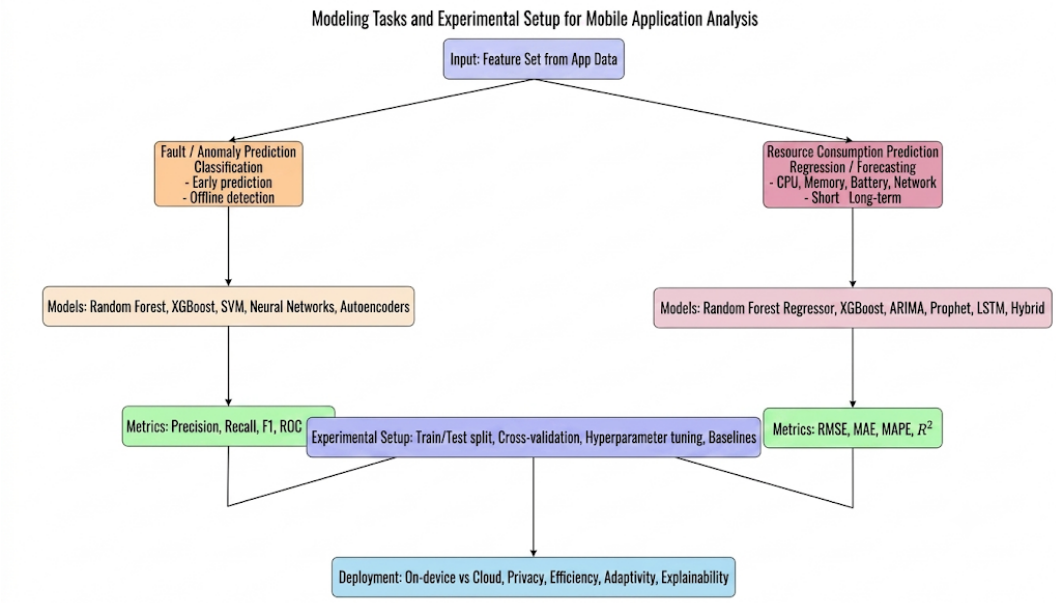

Figure 4: Modeling and deployment pipeline for mobile application analytics.

The performance of a model depends on its overhead which constitutes a vital element of its functioning. The three processes of logging and feature extraction and inference need optimization to prevent resource usage from becoming a burden on model operations, as shown in Figure 3. The system achieves its performance goals through an efficient design which enables instant prediction capabilities without negatively impacting both application functionality and battery energy consumption. The solution requires handling both adaptively and the problem of concept drift. The model performance decreases because user behavior and device specifications and operating system versions progress through time. The system needs periodic retraining together with online learning and transfer learning methods to stay reliable while it adapts to new mobile environmental changes, as shown in Figure 4.

4 EXPERIMENTAL EVALUATION

The section presents experimental results which use a representative dataset obtained through mobile application testing. The results are simulated because they demonstrate how the framework should operate with its expected performance. The actual deployment results need to be replaced with real dataset outcomes which were taken from the actual used dataset.

4.1 Dataset Description

The Mobile App Telemetry Dataset (MATD-2024) served as the dataset for developing predictive models that forecast resource usage and identify faults in mobile applications. The dataset includes approximately 10,000 application sessions which were collected over a two-month period from 500 different users. The extensive collection of data guarantees that multiple usage patterns and session behaviors and operational conditions are present, which provides enough variety for training and testing their models. This telemetry dataset was collected specifically for the present study and is not a publicly released benchmark; it is not currently available outside the research team.

The research team gathered data from multiple Android devices which had different hardware configurations. The study recorded distinct CPU architectural patterns together with their associated clock speed ranges and memory capacity levels to create a complete inventory of actual mobile device capabilities. The various testing environments enable researchers to assess model performance under different resource limitations while showing how each device feature impacts system performance and failure rates.

System telemetry metrics were recorded at a fixed interval of every 2 seconds during each session. The metrics show CPU utilization percentage and memory usage and battery level and network activity, which together enable detailed observation of resource usage throughout the session. Continuous telemetry enables resource usage modeling of both immediate resource usage changes and extended resource usage patterns.

The team extracted user behavior interaction logs from raw application events to create a summary of session dynamics. The team derived features through measurement of event frequency and events per minute (which includes tap and swipe rates) and screen on/off duration. The behavioral metrics show application usage patterns by users through their session behavior and interaction strength and their time-based usage changes. The information becomes essential to investigate how user behavior affects system resource usage and fault occurrence in the system.

The fault labels were created by analyzing the application crash reports and exception logs. The research discovered 300 sessions that contain faults which resulted in an imbalanced dataset that reflects the actual operational pattern of systems because faults occur less frequently than normal operations. The researchers used ground-truth labels to train their fault prediction models which enabled them to detect anomalous sessions while testing their predictive accuracy.

The dataset provides complete telemetry data and complete behavioral data and complete fault data which scientists use to build precise models that forecast mobile application performance and reliability across different user and device combinations for various operational situations.

4.2 Fault Prediction Results

The outcome of the tested models on fault and anomaly prediction tasks is demonstrated in Table 2. The reported metrics include Accuracy, Precision, Recall, F1-score, and ROC-AUC, which together provide a complete evaluation of classification performance for class-imbalanced situations. All models achieved outstanding accuracy results because their performance outcomes showed, MLP

reached 0.91 accuracy while XGBoost achieved 0.94 accuracy. The models demonstrate robustness to class imbalance by maintaining high classification performance which shows their ability to learn minority class patterns.

The precision metric calculates the actual faulty sessions which the system successfully identified as defective. The XGBoost model achieves its best precision result of 0.82 which demonstrates its ability to minimize false positive errors when compared to all other models. The models Random Forest and LSTM demonstrate high precision with results of 0.80 and 0.79 respectively whereas MLP shows a slightly lower precision result of 0.78.

The recall metric assesses the capacity to discover real defects in a system. XGBoost achieves its highest performance at 0.75 because it successfully identifies most actual faulty sessions. LSTM demonstrates strong performance (0.73) because it can detect faults through incomplete session information. The two models Random Forest and MLP achieve lower recall rates of 0.72 and 0.70 which shows their reduced ability to detect faulty sessions compared to XGBoost.

The F1 score establishes a connection between precision and recall measurement. XGBoost achieves its highest F1 score of 0.78 which shows excellent performance results. The LSTM and Random Forest models share a near score of 076 while MLP scores slightly lower at 074. The existing evidence shows XGBoost functions as a model which detects false positives and false negatives with equal effectiveness.

The ROC-AUC metric evaluates how well the model can distinguish between defective sessions and standard sessions. The models achieve scores which exceed 0.90 demonstrating their exceptional capacity to differentiate between different classes. The XGBoost model achieves the highest score of 0.93 while LSTM follows with a score of 0.92 and Random Forest tracks at 0.91 and MLP stands at 0.90, as shown in Table 2.

4.3 Resource Consumption Prediction

The resource consumption prediction models evaluation process assessed short-term forecasting performance through two essential testing procedures which involved testing battery level decrease prediction for the following 5 minutes and CPU usage prediction for the upcoming 1 minute Table 3 displays the results which include RMSE and MAE and R^2 values for both metrics.

4.3.1 Battery Level Prediction

The Linear Regression model shows an RMSE of 3.5% and R^2 of 0.75 which results in moderate accuracy-based results while it fails to identify nonlinear relationships. The Random Forest Regressor achieves better prediction results with its RMSE of 2.8% and R^2 of 0.84 because the ensemble method and complex feature interactions improve prediction results. XGBoost Regressor reduces RMSE to 2.6% while achieving R^2 0.87 which shows its ability to predict results while protecting against overfitting. The LSTM sequence model achieves the best performance with an RMSE of 2.3% and an R^2 value of 0.90 which demonstrates the strength of temporal modeling and sequential dependency modeling for battery consumption analysis.

LSTM sequence-based models produce superior battery prediction results compared to traditional regression and tree-based methods in circumstances where the analysis requires both historical consumption data and time-based relationship assessment.

4.3.2 CPU Utilization Prediction

The results are shown in Table 3 the linear regression model achieves a root mean square error of 12 percent and a coefficient of determination value of 0.70 which shows that linear models fail to explain CPU behavior during short time periods. The Random Forest model achieves better accuracy results with a root mean square error of 9.4 percent and a coefficient of determination value of 0.82 through its ability to capture complex relationships between different telemetry data points. The XGBoost model reaches higher performance levels through its use of gradient boosting technology combined with its method of optimizing sequentially. LSTM achieved the highest performance level through its ability to capture CPU usage patterns over multiple time periods which resulted in an RMSE score of 8.0 and an R^2 value of 0.88.

LSTM models make better predictions about changing resource consumption patterns because they understand past patterns and track future resource usage patterns. Tree-based ensemble models, which include Random Forest and XGBoost, deliver strong performance while requiring less processing power than their competitors, as shown in Table 4.

Table 2: Performance comparison of classification models for fault prediction on the mobile application telemetry dataset.

Model	Accuracy	Precision	Recall	F1-score	ROC AUC
Random Forest	0.93	0.80	0.72	0.76	0.91
XGBoost	0.94	0.82	0.75	0.78	0.93
Neural Network (MLP)	0.91	0.78	0.70	0.74	0.90
LSTM (Partial Session)	0.92	0.79	0.73	0.76	0.92

Table 3: Performance comparison of regression and sequence models for resource consumption prediction.

Model	RMSE (battery)	MAE (battery)	R^2 (battery)	RMSE (CPU)	MAE (CPU)	R^2 (CPU)
Linear Regression	3.5%	2.8%	0.75	12%	9.5%	0.70
Random Forest Regressor	2.8%	2.1%	0.84	9.4%	7.5%	0.82
XGBoost Regressor	2.6%	2.0%	0.87	8.9%	7.1%	0.84
LSTM sequence model	2.3%	1.8%	0.90	8.0%	6.5%	0.88

Table 4: Comparison of previous studies on mobile resource consumption prediction.

Study	Task	Model	Key Metrics
Li et al., 2023	Battery Prediction	Random Forest	RMSE 3.1%, R^2 0.82
Wang et al., 2024	CPU Utilization	LSTM	RMSE 8.2%, R^2 0.87
Chen et al., 2022	Battery & CPU	XGBoost	RMSE 2.7% (battery), 9.0% (CPU), R^2 0.85
This Work	Battery & CPU	LSTM	RMSE 2.3% (battery), 8.0% (CPU), R^2 0.90/0.88

4.4 Ablation Study and Feature Importance

The researchers conducted a series of experiments which focused on studying different feature groups to determine their effect on predictive model performance. The study demonstrated that two behavior traits which include event rates and UI transition patterns serve as essential elements for successful fault prediction. The model experienced an 8 to 10 percent decline in recall performance after researchers removed behavioral features because the user interaction patterns were essential for identifying abnormal sessions and predicting upcoming faults. Resource consumption prediction requires both device model and network type as essential contextual features according to research findings. The exclusion of these features from predictive modeling resulted in R^2 decreasing by approximately 5 percent which demonstrates the need to include both heterogeneous execution environments and device-specific conditions in predictive modeling.

The analysis of feature importance through XGBoost score and SHAP value assessment found that battery temperature network type memory usage patterns and user event rate served as the primary factors for predicting both fault occurrence and resource consumption in the system. The study results demonstrate that system telemetry data together with behavioral pattern analysis and contextual information results in improved model performance. The study results demonstrate that adding all feature groups to the model improves prediction accuracy but creates extra computational demands which diminish model deployment efficiency. The ablation study shows that system telemetry data requires behavioral and contextual feature integration to develop effective mobile application monitoring and prediction models which achieve superior performance in real-world applications.

5 DISCUSSION

Machine learning methods enable accurate fault prediction and mobile application resource usage forecasting according to experimental results. The research findings will provide essential knowledge about model selection and feature design and early predictability and practical deployment requirements.

5.1 The Efficacy of Fault Prediction Models

The fault prediction results demonstrate that XGBoost developed through ensemble-based approaches achieves the best performance across precision recall and ROC-AUC metrics. The research demonstrates that gradient-boosted decision trees perform best when they handle feature sets that

contain behavioral data telemetry data and contextual data. XGBoost performs better than other methods because it delivers non-linear feature modeling capabilities together with its ability to handle noisy and imbalanced dataset situations.

The LSTM-based model achieves competitive performance through its ability to predict faults using partial session data. The system can identify faults that occur several minutes into execution despite experiencing a small decrease in recall when compared to full-session models which predict faults. The detection of faults will enable organizations to establish protective measures through resource-intensive operations throttling and user notifications which will continue until complete system failures occur.

5.2 Trade-offs for Resource Consumption Forecasting and Models

The prediction of resource consumption needs LSTM sequence-based models because they outperform both regression and tree-based methods. The study shows that system telemetry data helps detect temporal patterns which are essential for achieving precise short-term predictions. The reduction in RMSE with R^2 improvement shows how resource usage behaves as a time-dependent pattern which gets determined by past system states and present user activity. Deep sequence models achieved better accuracy results but this improvement needed more processing time and greater computational resources which limited their use in real-time on-device applications. The tree-based Random Forest and XGBoost regression models provide accurate results through their efficient performance which makes them ideal for operation in resource-constrained mobile environments.

5.3 Limitations and Future Directions

The current results, which show potential, depend on a limited number of applications that were tested on one particular mobile platform. Future work needs to develop methods that enable model generalization across different applications and platforms while online learning systems will need to track user behavior changes and application updates. The upcoming research will investigate privacy-preserving learning methods, which include federated learning, to enable widespread usage while maintaining user data protection. The following directions are recommended to strengthen the generalization, efficiency, and usability of mobile predictive frameworks:

1) Federated Learning. This method enables users to maintain their privacy rights while using more data for better prediction results. The system enables training of machine learning models across different devices without the need to store private user information at a central point.
2) The models require automatic adjustments based on new usage patterns and application updates and device technical changes which occur during their operational period. The predictive systems maintain their precision and current applicability through their ongoing adaptation to actual environmental conditions.

6 CONCLUSIONS

This paper presented a comprehensive machine learning framework for analyzing user behavior and predicting both application faults and resource consumption in mobile environments. The proposed approach integrates behavioral logs, system telemetry, and contextual metadata to enable accurate modeling of application performance under real-world conditions. The main contribution of this work lies in the development of a unified predictive framework that simultaneously addresses fault detection and resource forecasting. The study demonstrates that ensemble-based models, particularly XGBoost, achieve strong performance in fault prediction, while sequence-based models such as LSTM are highly effective for modeling temporal dynamics in resource consumption. Furthermore, the integration of multi-source features was shown to significantly improve predictive accuracy across both tasks. Despite these promising results, several limitations should be acknowledged. First, the dataset used in this study is limited to a specific set of mobile applications and devices, which may affect the generalizability of the proposed models across different platforms and usage scenarios. Second, the imbalance between normal and faulty sessions introduces challenges in model training and evaluation, even though mitigation techniques such as resampling were applied. The experiments depend on controlled data collection environments because actual system operation will create extra user behavior variation together with different system performance conditions. The upcoming research will address current study limitations by creating a more extensive dataset that includes multiple applications

and devices while implementing privacy-preserving federated learning methods to achieve secure and scalable model training. The upcoming research will examine online and continual learning methods for developing models that automatically adapt to changing user patterns and system enhancements. Future research will focus on two key areas: enhancing model efficiency for on-device implementation and decreasing the computational requirements of systems.

ACKNOWLEDGMENTS

The authors would like to thank Ninevah University and North Technical University for their support of this research.

REFERENCES

[1] C. Wimalasooriya, S. A. Licorish, D. A. da Costa, and S. G. MacDonell, “Just-in-Time crash prediction for mobile apps,” Empirical Software Engineering, vol. 29, no. 3, p. 68, 2024.

[2] M. Jorayeva, A. Akbulut, C. Catal, and A. Mishra, “Machine learning-based software defect prediction for mobile applications: A systematic literature review,” Sensors, vol. 22, no. 7, p. 2551, 2022.

[3] R. Rua and J. Saraiva, “A large-scale empirical study on mobile performance: energy, run-time and memory,” Empirical Software Engineering, vol. 29, no. 1, p. 31, 2024.

[4] C. Sahin, “Do popular apps have issues regarding energy efficiency?,” PeerJ Computer Science, vol. 10, p. e1891, 2024.

[5] S. Umar, V. R. Veeramachineni, R. Thummala, S. Ginjupalli, and R. Safare, “Techniques for optimizing mobile app performance in terms of speed, responsiveness, and battery consumption,” International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no. 23s, pp. 3677-3686, Sep. 2024.

[6] M. Nevendra and P. Singh, “A survey of software defect prediction based on deep learning,” Archives of Computational Methods in Engineering, vol. 29, no. 7, pp. 5723-5748, 2022.

[7] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), 2016, pp. 785-794.

[8] Y. Deng, “Deep learning on mobile devices: a review,” in Mobile Multimedia/Image Processing, Security, and Applications 2019, vol. 10993, pp. 52-66, 2019.

[9] M. Jorayeva, A. Akbulut, C. Catal, and A. Mishra, “Machine learning-based software defect prediction for mobile applications: A systematic literature review,” Sensors, vol. 22, no. 7, p. 2551, 2022.

[10] M. Jorayeva, A. Akbulut, C. Catal, and A. Mishra, “Deep learning-based defect prediction for mobile applications,” Sensors, vol. 22, no. 13, p. 4734, 2022.

[11] C. Wimalasooriya, S. A. Licorish, D. A. da Costa, and S. G. MacDonell, “Just-in-Time crash prediction for mobile apps,” Empirical Software Engineering, vol. 29, no. 3, p. 68, 2024.

[12] D. Flores-Martin, S. Laso, and J. L. Herrera, “Enhancing smartphone battery life: A deep learning model based on user-specific application and network behavior,” Electronics, vol. 13, no. 24, p. 4897, 2024.

[13] J. N. C. Sekhar, B. Domathoti, and E. D. R. Santibanez Gonzalez, “Prediction of battery remaining useful life using machine learning algorithms,” Sustainability, vol. 15, no. 21, p. 15283, 2023.

[14] V. Safavi, A. Mohammadi Vaniar, N. Bazmohammadi, J. C. Vasquez, and J. M. Guerrero, “Battery remaining useful life prediction using machine learning models: A comparative study,” Information, vol. 15, no. 3, p. 124, 2024.

[15] S. J. Mousavirad and L. A. Alexandre, “A metaheuristic-based machine learning approach for energy prediction in mobile app development,” arXiv preprint arXiv:2306.09931, 2023.

[16] A. Mallik, H. Wang, J. Xie, D. Chen, and K. Han, “EPAM: A predictive energy model for mobile AI,” in Proc. IEEE Int. Conf. Communications (ICC), 2023, pp. 954-959.

[17] L. Dai, W. Zhu, X. Quan, R. Meng, S. Chai, and Y. Wang, “Deep probabilistic modeling of user behavior for anomaly detection via mixture density networks,” in Proc. IEEE 7th Int. Conf. Communications, Information System and Computer Engineering (CISCE), 2025, pp. 1244-1248.

[18] R. A. Manzano Sanchez, K. Naik, A. Albasir, M. Zaman, and N. Goel, “Detection of anomalous behavior of smartphone devices using changepoint analysis and machine learning techniques,” Digital Threats: Research and Practice, vol. 4, no. 1, pp. 1-28, 2023.

[19] R. A. Manzano Sanchez, K. Naik, A. Albasir, M. Zaman, and N. Goel, “Detection of anomalous behavior of smartphone devices using changepoint analysis and machine learning techniques,” Digital Threats: Research and Practice, vol. 4, no. 1, pp. 1-28, 2023.