User identification based on behavioral patterns of mobile devices

Conducted by Kseniia Yakunina

Contents

1 Introduction

The main goal of this project is to build a system that can identify users based on their motion behavior in mobile sessions, using training data with known users and applying it to test sessions where the user is unknown.

To achieve this goal, the following steps were taken: research analysis of data, preprocessing of data and creation of new features, as well as testing various approaches to feature selection and training different types of models.

The final stage is the selection of the best model and the creation of results for further use.

The main result of this project is the creation of the best model for user prediction.

This report contains our main assumptions and thought process, key insights from EDA, feature selection and modification strategy, model simulation and evaluation, as well as business implications and recommendations.

In this study, we presented initial hypotheses that different users behave differently with regard to different sensor metrics. We also assume that different users may have different activity times (day, night) and differences in session duration. For example, gamers will have longer gadget usage sessions and will also, for example, turn it more actively.

2 Data Overview

Dataset:

- Total Events: 2,318,350
- Total sessions: 300 (15 сессий на пользователя)
- Total Users: 20
- Sensor types: 6 (Gyroscope, Accelerometer, Gravity, Magnetometer, Rotation, User Acceleration)
- Created features: 516

For the initial data analysis, we performed both a visual assessment of the data through visualization and statistical tests.

Data Observations:

- Accelerometer generates the highest number of events (741,521 or 32%)
- All sensors are evenly represented (approximately 300k events each)
- Balanced distribution of users
- Average session duration: 7,728 events (median: 5,567)

2.1 Tests for Normal Distribution

As a result of the normality analysis, we see that the data does not follow a normal distribution and has heavy tails and outliers. As a result, it is better to use ensemble models and neural networks for forecasting, as they are more suitable for such types of data. Robust statistics should also be taken into account when creating new parameters.

2.2 Autocorrelation tests

In all sensors, the Ljung–Box test gives extremely low p-values, which means that autocorrelation is statistically significant, while ADF confirms stationarity everywhere. Overall, there is an "inertia" of signals: strong short-term memory and weaker long-term memory.

2.3 Descriptive conclusions from exploratory data analysis

The majority of the data (32%) is generated by the accelerometer, whose signals reflect both the influence of gravity and the dynamics of movement, while the remaining sensors are distributed evenly (13-14% each) (Figure 1) . Despite a fivefold difference in the number of events between users, the dataset remains perfectly balanced for modeling due to the strictly equal number of sessions (15 per person).

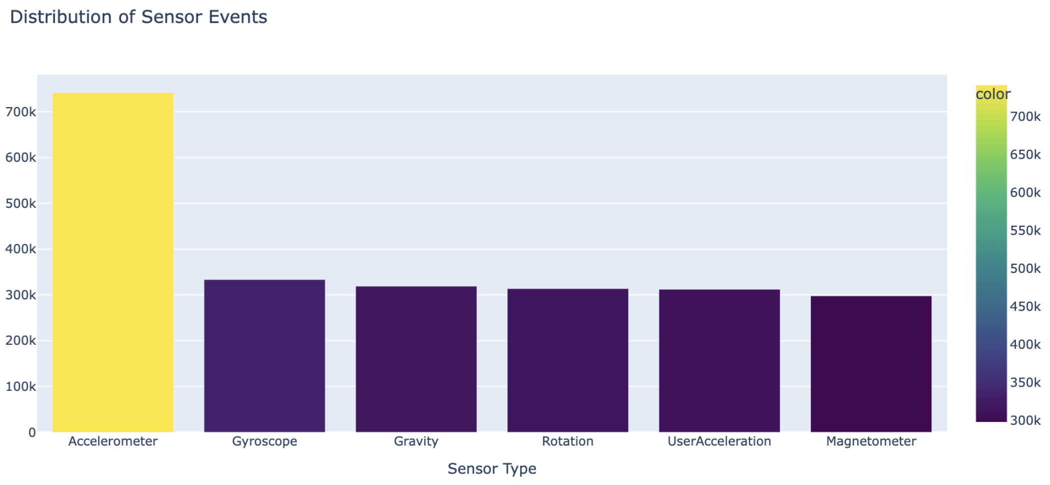

Figure 1: Events distribution by sensors

Analysis of the temporal distribution of events revealed distinct patterns of mobile device usage throughout the day. The time distribution graph (Figure 2) shows two distinct peaks of activity: the first occurs around 9 a.m. with approximately 400,000 events, which corresponds to the typical start of the workday when users actively interact with their devices to check notifications, read news, and plan tasks. The second, even more significant peak occurs at 2 p.m., reaching a maximum of approximately 540,000 events, which coincides with lunch break and afternoon hours. This indicates that users interact most intensively with their devices in the middle of the day, likely using them for social networking, communication, and entertainment. After 2:30 p.m., there is a gradual decline in activity to approximately 200,000 events by 3 p.m., followed by a slight increase around 8 p.m. (120,000 events) —

evening use after work. The night hours (from 10 p.m. to 6 a.m.) are characterized by minimal activity (about 50,000 events).

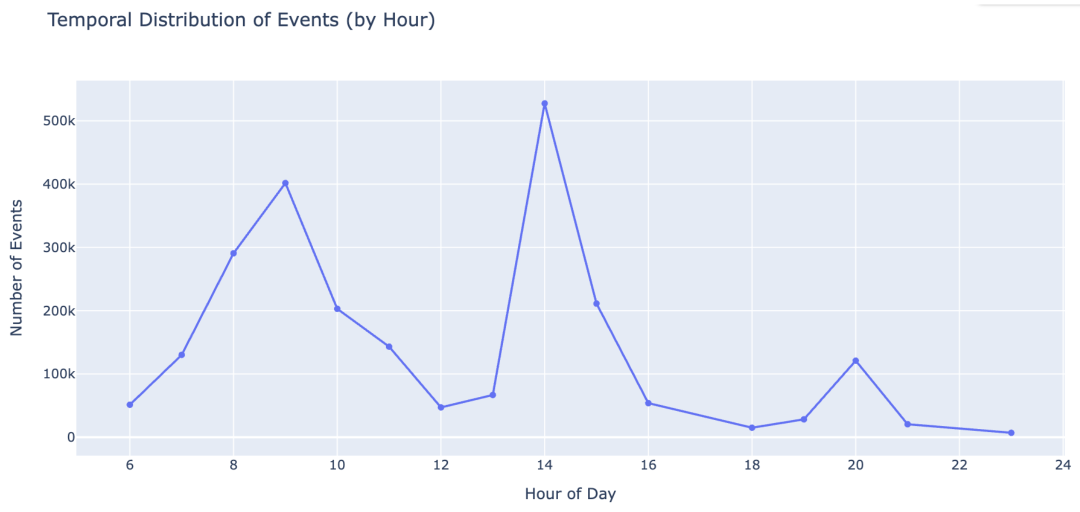

Figure 2: Time distribution

2.4 Cluster analysis of users

User profiles can be clearly divided into two layers: behavioral (how often and how long sessions last) and sensory (how "noisy" and dynamic movements are within sessions). The visualization (Figure 3) shows that it is behavioral characteristics that cause the greatest heterogeneity: the number of events, session duration, and average interval between events fluctuate significantly, forming a contrast between "active" and "static" interaction styles. Sensory characteristics appear to be more stable for most people, especially those related to gravity, which is logical—they reflect a more constant background orientation of the device.

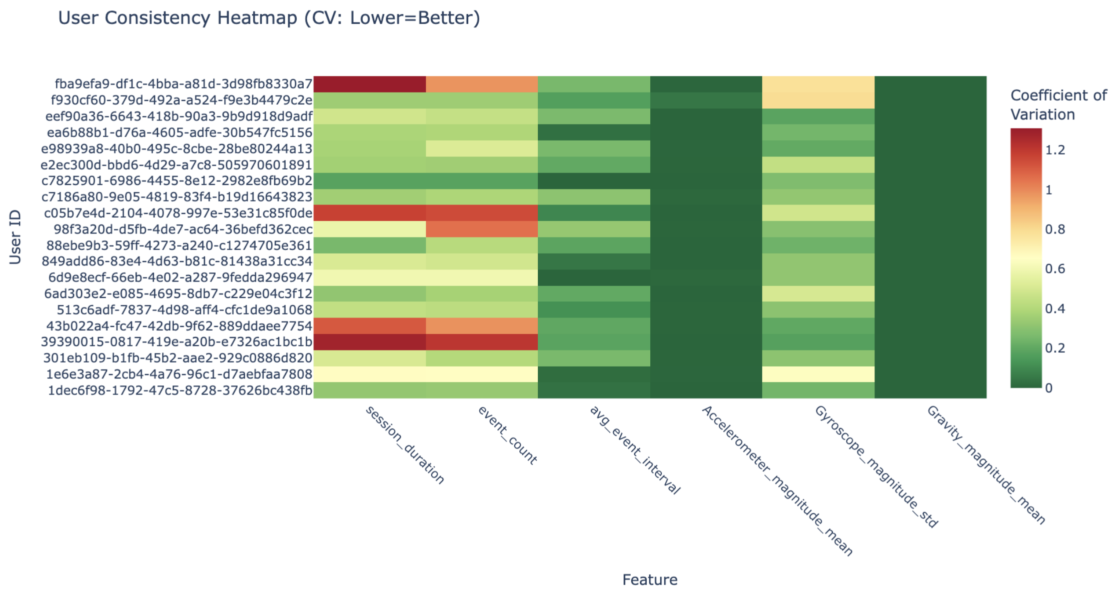

Figure 3: User Consistency

At the interpretation level, three characteristic types of behavior can be distinguished: "active" profile — frequent events, high total accelerometer energy, and shorter intervals between events; “static” profile — infrequent events, low energy, and large intervals, which is similar to calm use without sudden movements; “dynamic” profile — pronounced variability in gyroscope and gravity (more rotations and changes in device position), which can manifest even without the maximum number of events. Overall, the picture (Figure 4) shows that users differ primarily in the pace and structure of their interactions, with sensor metrics acting as a refining layer that distinguishes “simply frequent use” from “movement-intensive” behavior.

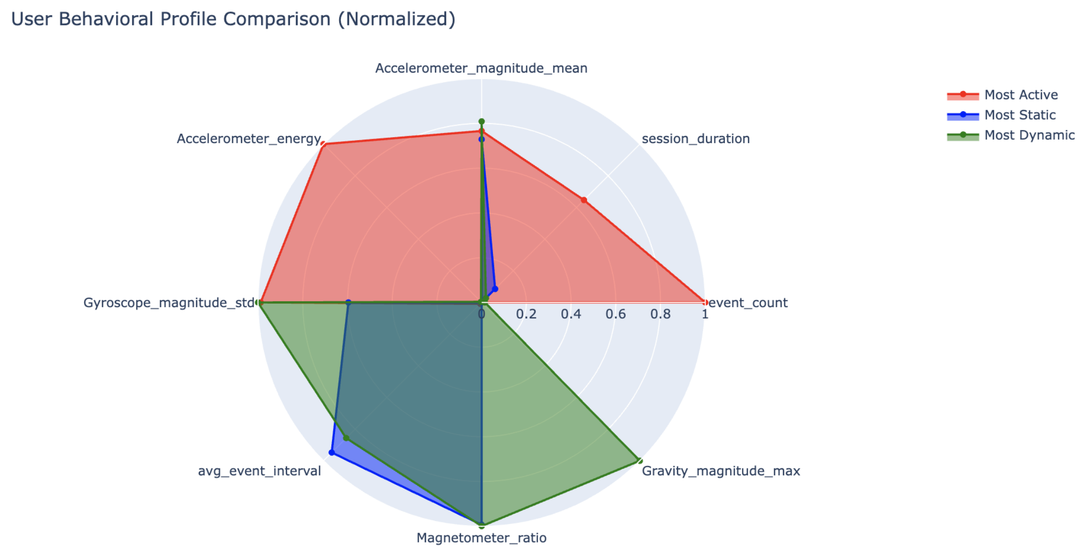

Figure 4: User Behavior

3 Feature Engineering

Based on raw data from six types of sensors (accelerometer, gyroscope, gravity, magnetometer, rotation, and user acceleration), which generate thousands of events per session, we aggregated the information to the session level, creating 516 features distributed across four categories. First, based on EDA, we looked at temporal characteristics such as session duration, number of events, and statistics on the intervals between events. The second category included sensor activation ratios, such as how the user holds the device and their current activity level (dynamic or static). Third, we created 10 statistical metrics based on sensor field indicators. And the last category of features is derived features of magnitude and energy for three-axis sensors — calculated as the Euclidean norm of acceleration/rotation vectors and their quadratic sum, ensuring invariance to device orientation and capturing the overall intensity of the user’s physical activity.

These metrics allow us to digitize the user’s unique signature: the intensity of their movements through magnitude and energy features, as well as their characteristic manner of holding the device through signal distribution. Based on the results of Feature Engineering, we see that the most important factors for user identification are aggregated features such as the average interval between events and sensor frequency ratios, which reveal the unique “rhythm” and style of holding the device. While the mutual information metric highlights behavioral patterns, the F-statistic indicates the extremely high discriminative power of gravity and rotation indicators, confirming the stability of smartphone orientation for each individual user (Figure 5).

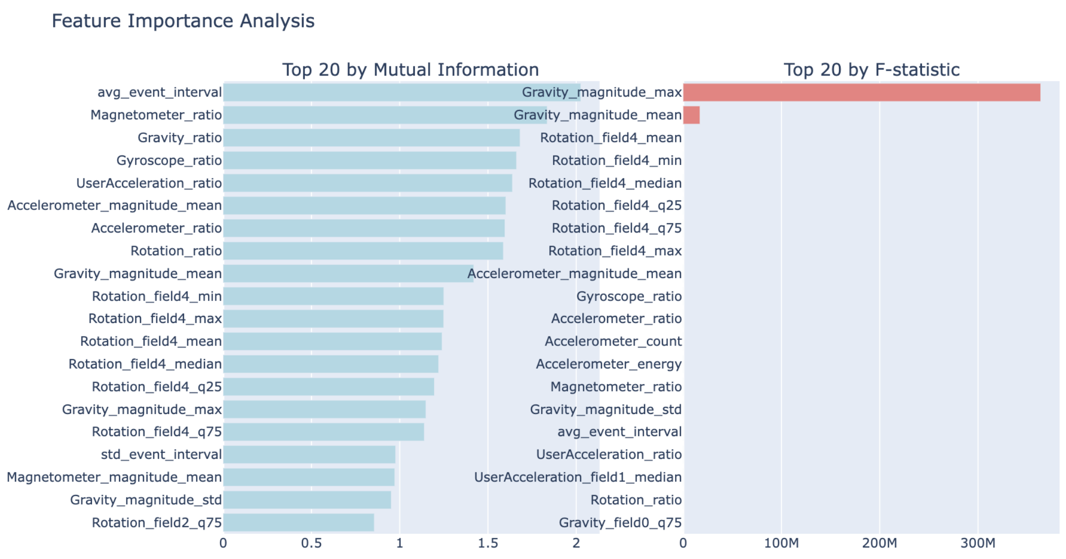

Figure 5: Feature Importance

4 Models

Models of different levels were used for analysis. The baseline model was logistic regression, followed by more complex machine learning models such as Random Forest and XGBoosting. We then tried ensembles and stacking models consisting of trees and boosting models, as combining models can improve quality, although it can affect the speed of model execution. The Random Forest and XGBoosting models had hyperparameter tuning to achieve the best prediction stage. The last model in our analysis was a neural network with the following scheme:

```
Input(516) → Dense(256) → Dropout(0.3) → BatchNorm → Dense(128) →
Dropout(0.3) → BatchNorm → Dense(64) → Dropout(0.2) → BatchNorm →
Dense(32) → Dropout(0.2) → Softmax(20) [Output]
```

These models were chosen because they have different levels of complexity and speed, as well as different mathematical approaches to data prediction, which allows us to find the optimal option for prediction.

4.1 Evaluation Metrics

For a comprehensive assessment of the models, we used a set of metrics that went beyond simple accuracy, as the overall percentage of correct answers does not always take into account the quality of the separation of similar users. This is especially true given that we have a multi-class classification. The key indicator was the F1-Score, which harmonizes accuracy and completeness, which is critical for security systems where it is equally important not to miss an attacker and not to block a legitimate user.

Particular attention was paid to the Matthew's correlation coefficient (MCC) and Cohen's kappa, which evaluate the model for random guessing and take into account all elements of the error matrix. MCC values close to 1 confirm the high robustness of our algorithms. The analysis is completed by the ROC-AUC metric, which confirms the excellent ability of the models to rank probabilities even when the sensitivity threshold of the system changes.

4.2 Feature Selection

A comparison was also made between different numbers of feature selections and dimensionality reduction, and how this affected the quality of the predicted models. A comparison was made between the baseline and random forest models.

Experiments on dimensionality reduction showed that Feature Selection is more useful than PCA for this task. PCA does reduce dimensionality and speed up training, but in results of comparisons, it is accompanied by a deterioration in quality and instability of results. Feature selection, on the other hand, preserves interpretability and often improves metrics on a compact set (especially noticeable for ensembles), which is why feature selection is recommended as a basic optimization step for production.

4.3 Results

Overall, we can see that even the least successful models show fairly good accuracy, provided that we have little data by creating additional features (Figure 6). The small amount of data per user could also affect the quality of neural network training, which is why this project recommends using classic machine learning methods.

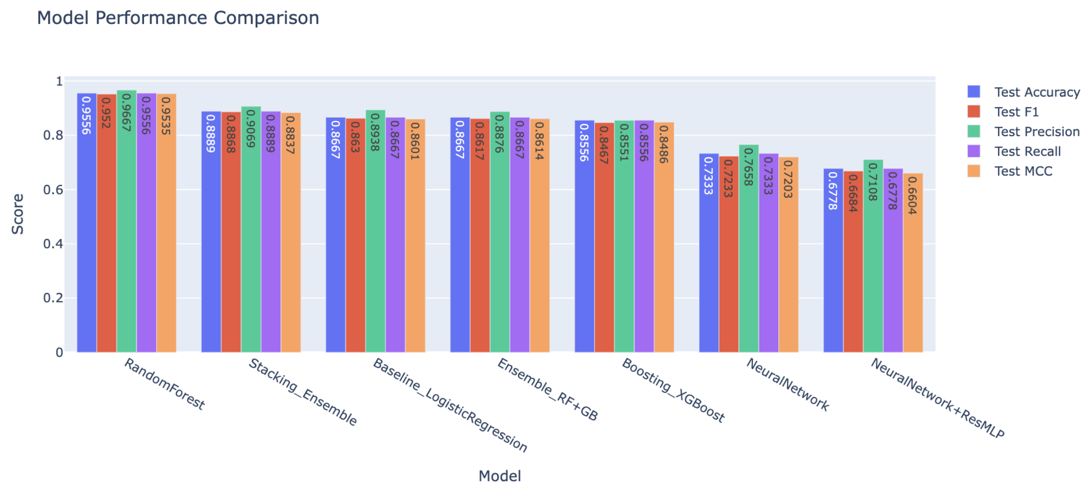

Figure 6: Results

The results of the comparative analysis show that the clear leader in terms of quality in the initial features is RandomForest: it gave the highest values of Test Accuracy = 95.56%, F1 = 0.952 and MCC = 0.953, i.e., the best "robustness" and stability among all approaches tested. A comparative analysis of the models showed that the best results are achieved with ensemble methods, with RandomForest as the key leader. When training the final model on the complete set of features, high quality was achieved (Accuracy = 92.0%, F1 = 0.905, MCC = 0.917, AUC = 0.9985) with a realistic training time of ∼7.8 seconds.

The most practical option for production turned out to be RandomForest + Feature Selection (Top-200): it simultaneously improved quality and accelerated training. The model achieved Accuracy = 93.33%, F1 = 0.918, MCC = 0.931, AUC = 0.9978 with a training time of 3.75 seconds, which is approximately twice as fast as the option on all features, while surpassing it in terms of metrics. Thus, the final recommendation for implementation is to use RandomForest with a selection of top 200 features as the main identification algorithm.

The best parameters for Random Forest were selected using the RandomizedSearchCV approach :

```
Random_Forest_params = {
    "n_estimators": 100,
    "min_samples_split": 5,
    "min_samples_leaf": 2,
    "max_features": None,
    "max_depth": 20,
    "class_weight": None,
    "bootstrap": True,
    "n_jobs": -1,
    "random_state": 42,
}
```

Table 1: Results Table

Model	**Accuracy**	**F1**	**MCC**	**Time**	**Conclusion**
RandomForest_Best (FS Top-200)	93.33%	0.918	0.931	3.75 c (Fast)	RECOMMENDED final model: best balance of quality/speed, high robustness, fast fit.
RandomForest_Best (All features, without RandomizedSearchCV)	92.0%	0.905	0.917	7.83 c (Medium)	Good, but it loses to the Top 200 in terms of both quality and time; selection by characteristics is preferable.
RandomForest (All features + RandomizedSearchCV)	95.56%	0.952	0.953	548.41 c (Slow)	Maximum quality, but training time is very high. Suitable if maximum quality is important and retraining is rare.
RandomForest (FS Top-50)	90.67%	0.906	0.902	1.41 s (Fast)	Good fast option, but noticeably inferior to Top-200 in quality.
RandomForest (FS Top-300)	90.67%	0.892	0.903	6.55 s (Medium)	No advantage over Top-200: lower quality, longer time.
XGBoost (All features, FS-experiment)	90.0%	0.898	0.896	156.77 s (Very slow)	Good quality, but too slow compared to RF Top-200 with worse metrics.
LogReg (FS Top-200, FS-experiment)	90.0%	0.898	0.896	0.10 s (Very fast)	Very strong fast baseline, but RF Top-200 gives higher quality with still moderate time.
RandomForest (FS Top-100)	89.33%	0.885	0.888	2.51 s (Fast)	Fast, but worse than Top-200; can be a compromise with even smaller feature set.
LogReg (FS Top-200)	89.33%	0.893	0.888	0.08 s (Very fast)	Edge/real-time: best LR option (fast + noticeably better than baseline LR).

Continued on the next page

Continued Table

Model	Accuracy	F1	MCC	Time	Conclusion
Stacking_Ensemble (All features)	88.89%	0.887	0.884	96.83 s (Slow)	Can be considered, but inferior to RandomForest in quality with significant training time.
LogReg (FS Top-300)	88.0%	0.877	0.874	0.10 s (Very fast)	Worse than Top-200 with similar time — not optimal.
Baseline_LogisticRegression (All features)	86.67%	0.863	0.86	0.15 s (Very fast)	Edge/real-time: very fast and interpretable, but worse in quality than RF.
Ensemble_RF+GB (All features)	86.67%	0.862	0.861	135.47 s (Very slow)	Not recommended: quality same as baseline LR, but significantly slower.
LogReg (FS Top-100)	86.67%	0.869	0.861	0.32 s (Very fast)	Fast option with no accuracy gain over baseline LR.
Boosting_XGBoost (All features, from summary table)	85.56%	0.847	0.849	136.55 s (Very slow)	In current configuration not justified: both slow and worse in metrics than RF/LR.
LogReg (All features)	85.56%	0.845	0.85	2.31 s (Fast)	Basic fast option, but inferior to LR Top-200 and RF Top-200.
LogReg (FS Top-100)	85.56%	0.851	0.85	0.06 s (Very fast)	Maintains quality, greatly accelerates; but LR Top-200 better in accuracy.
LogReg (FS Top-50)	84.0%	0.842	0.832	0.19 s (Very fast)	Very fast, but quality drops; usually better to use Top-200.
LogReg (FS Top-50)	82.22%	0.817	0.814	0.09 s (Very fast)	Too strong quality loss; not optimal.
NeuralNetwork (All features)	73.33%	0.723	0.72	17.21 s (Medium)	Insufficient data; current version noticeably inferior to trees/ensembles.

Continued on the next page

Continued Table

Model	**Accuracy**	**F1**	**MCC**	**Time**	**Conclusion**
NeuralNetwork+ResMLP (All features)	67.78%	0.668	0.66	39.52 s (Slow)	Not recommended: worse in quality and not faster than baseline solutions.

5 Conclusion

In summary, in this work, research on user data on their interaction with smartphones, identified interaction patterns, and models for predicting user activity with an accuracy of 93% and an F1-score of 91% has been developed.

In general, the choice of a specific architecture should be based on business priorities: if maximum accuracy is required, choose RandomForest; if speed and simplicity are critical for production/edge, Logistic Regression is the best choice. Nevertheless, for most scenarios, RandomForest provides the necessary depth of behavioral data analysis while maintaining high information processing speed and excellent probability calibration.

It is also worth noting that these results were achieved under controlled conditions and require validation using real data in production. Therefore, the main recommendations are: monitoring critical aspects such as changes in user behavior patterns and data from different types of smartphones. Also, it is mandatory to have A/B testing of the model to measure real metrics such as False Acceptance Rate and False Rejection Rate and conduct a final testing of prediction latency and resource consumption.