Waze User Chrun Prediction

Andres Cordeiro

August 2024

1 Introduction

Churn prediction is a critical task in the realm of customer relationship management, where the primary objective is to anticipate which users are likely to discontinue using a service. In this project, we focus on predicting user churn for Waze, a popular GPS navigation software, using a dataset sourced from Kaggle. Accurate churn prediction enables companies like Waze to implement targeted retention strategies, reducing the loss of users and enhancing long-term profitability.

The dataset provided includes various features related to user behavior and interactions with the Waze application. To tackle this predictive task, we employ two machine learning models: Random Forest and Extreme Gradient Boosting (XGBoost), and several deep learning models, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), and Gated Recurrent Units (GRU). Each model is evaluated based on key metrics such as accuracy, precision, recall, F1-score, and AUC-ROC to determine its effectiveness in predicting churn.

This report delves into the data exploration, preprocessing steps—including outlier treatment and scaling—and the performance comparison of the different models. Our goal is to identify the most effective model for accurately predicting churn and to provide actionable insights based on the analysis.

2 Main Objective of Analysis

The main objective of this project is to develop a predictive model that can accurately identify users who are at risk of churning from the Waze platform. By leveraging various deep learning techniques, the aim is to analyze patterns in user behavior and interactions with the app to forecast potential churn. Successfully predicting churn allows Waze to proactively engage with at-risk users through personalized interventions, thereby reducing churn rates and fostering user retention. This project not only seeks to determine the most effective model for this task but also to provide insights that can inform data-driven strategies for enhancing user loyalty and satisfaction.

3 Description of the Data Set

In this section, we gave a brief description of the Data Set and its attributes. The dataset used in this project is sourced from Kaggle and is specifically designed to predict user churn for the Waze platform. It consists of multiple features that capture various aspects of user behavior and interactions with the application. The dataset includes a mix of categorical and numerical variables, providing a comprehensive view of user engagement.

Attribute	**Type**	**Description**
ID	Categorical	A unique identifier for each record or user.
label	Binary	The target variable indicating whether a user has churned (1 if churned, 0 otherwise).
sessions	Numerical	The number of app sessions initiated by a user.
drives	Numerical	The number of driving sessions recorded by the user.
total_sessions	Numerical	The total number of sessions (driving and non-driving) initiated by the user.
n_days_after_onboarding	Numerical	The number of days since the user completed the onboarding process.
total_navigations_fav1	Numerical	The total number of navigations to the user's first favorite location.
total_navigations_fav2	Numerical	The total number of navigations to the user's second favorite location.
driven_km_drives	Numerical	The total distance (in kilometers) driven by the user during driving sessions.
duration_minutes_drives	Numerical	The total duration (in minutes) of all driving sessions recorded by the user.
activity_days	Numerical	The total number of days the user was active on the app.
driving_days	Numerical	The total number of days the user used the app while driving.
device	Categorical	The type of device used by the user (e.g., iOS, Android).

Table 1: Attributes of the Waze User Churn Dataset

The dataset has a total of 13 attributes,and a total of 14999 entries, we have a of summary the total attributes in Table 1. For more details and information about the dataset, please refer to Kaggle's page `https://www.kaggle.com/datasets/juliasuzuki/waze-dataset-to-predict-user-churn/data/`)

4 Data Exploration

In the Exploratory Data Analysis (EDA) section, we began by summarizing the key statistical characteristics. We used histograms and box plots to examine the distribution of numerical features like: comments,share,paid,total interactions, and others identifying outliers and skewed distributions.

4.1 Descriptive Statistics

The Descriptive Statistics section is critical in the Data Exploration process as it provides a summary of the central tendency, dispersion, and shape of the distribution of each variable in the dataset. This helps in gaining an initial understanding of the data and sets the stage for deeper analysis.

Numerical features are often the focus in initial descriptive statistics because they provide insights into the distribution and central tendency of the data.

	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives
count	14292.000000	14292.000000	14292.000000	14292.000000	14292.000000	14292.000000	14292.000000
mean	80.663308	67.288763	189.587148	1751.977120	121.712427	29.633361	4043.899849
std	80.736547	65.946640	136.200356	1008.610971	147.665950	45.350345	2504.494865
min	0.000000	0.000000	0.220211	4.000000	0.000000	0.000000	60.441250
25%	23.000000	20.000000	90.466649	878.750000	10.000000	0.000000	2217.278591
50%	56.000000	48.000000	158.728448	1749.000000	71.000000	9.000000	3496.214642
75%	112.000000	93.000000	253.564188	2627.250000	178.000000	43.000000	5299.878068
max	743.000000	596.000000	1216.154633	3500.000000	1236.000000	415.000000	21183.401890

Figure 1: Descriptive Statistics

From the Table 1, we can observe the central tendency and dispersion of features like sessions, drives and the other attributes

4.2 Univariate Analysis

In this section we will analyze each variable independently to understand its distribution, central tendency and spread. This is a critical step in identifying the characteristics of each feature in the dataset, whether they are numerical or categorical.

To analyze numerical variables we have done:

- Histograms: To visualize the frequency distribution of the data.
- Box Plots: To identify the central tendency and dispersion, and detect outliers.
- Pie Charts: To visualize the frequency distribution of the data

Analyze the skewness and kurtosis to understand the shape of the distribution.

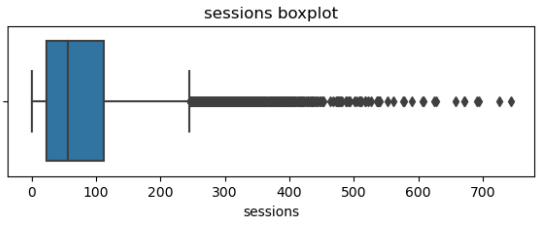

Figure 2: Boxplot of sessions.

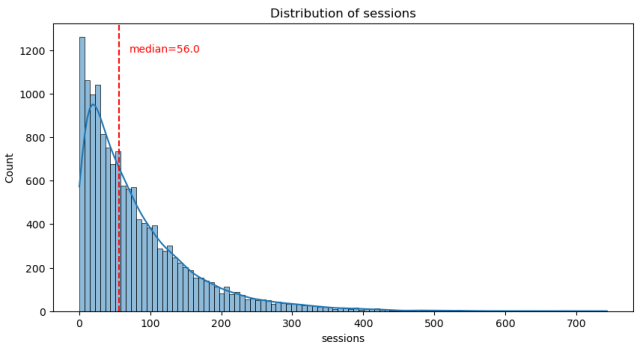

Figure 3: Distribution of sessions

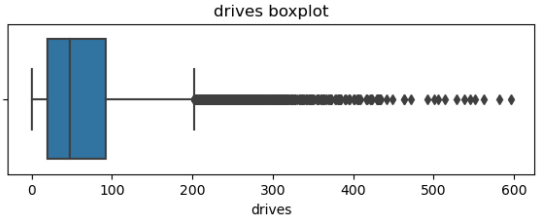

Figure 4: Boxplot of drives.

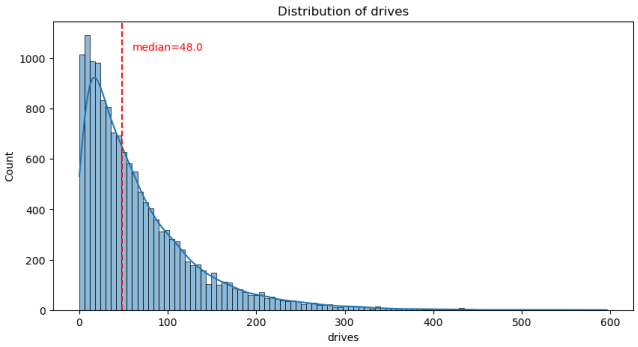

Figure 5: Distribution of drives

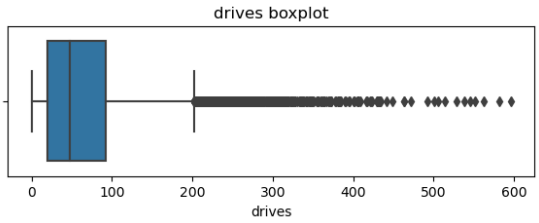

Figure 6: Boxplot of drives.

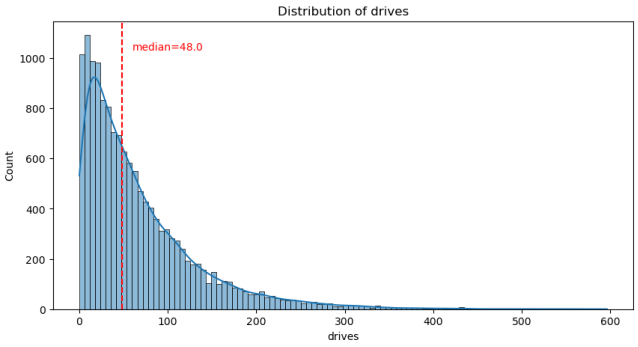

Figure 7: Distribution of drives

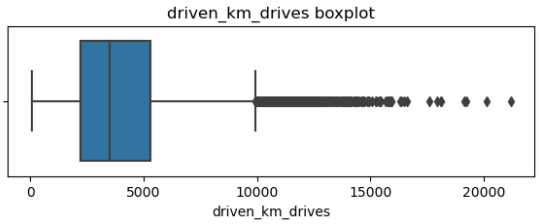

Figure 8: Boxplot of total distance driven by session.

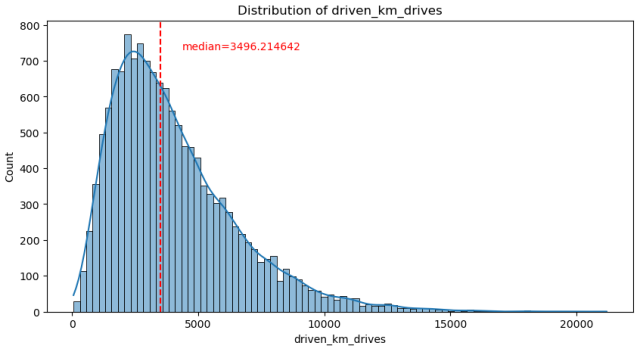

Figure 9: Distribution of total distance driven by session

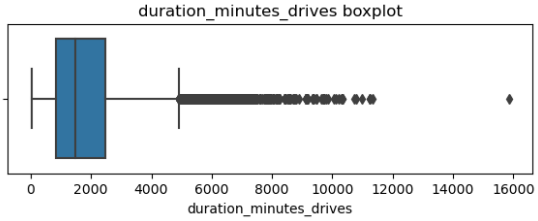

Figure 10: Boxplot of of total duration in minutes driven by session.

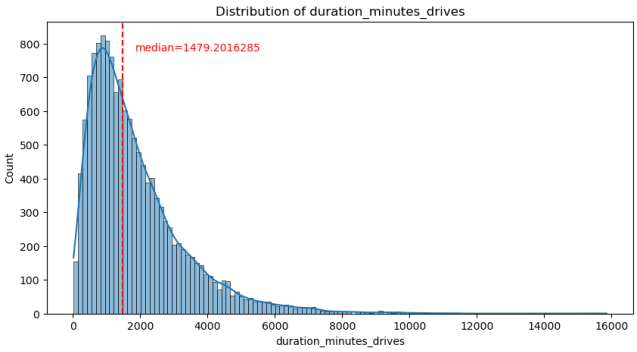

Figure 11: Distribution of total duration in minutes driven by session.

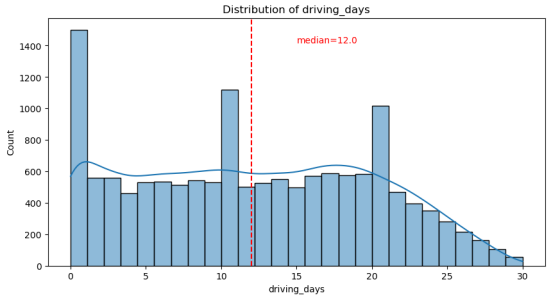

Figure 12: Distribution of number of days driven

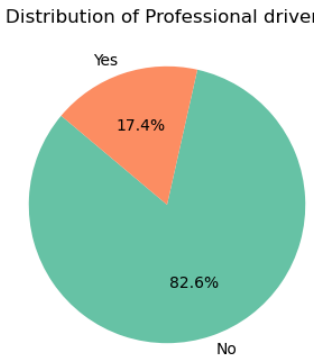

Figure 13: Pie chart of Professional drivers

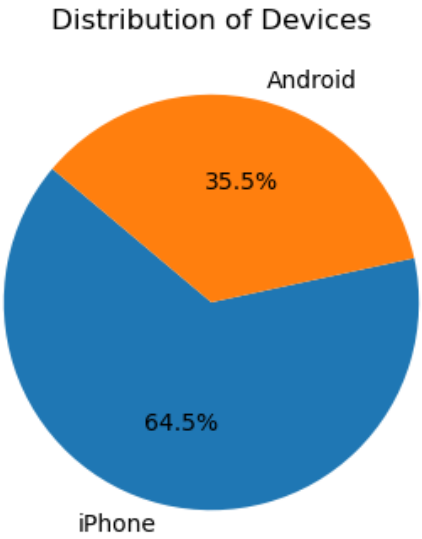

Figure 14: Distribution of devices

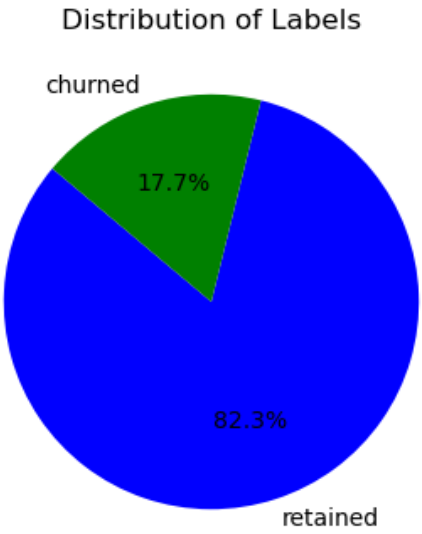

Figure 15: Distribution of Target variable

4.3 Bivariate Analysis

Bivariate analysis works on the basic of two variables at a time provides the basis for computation to derive the correlations, dependencies and interaction among the relations in the data. This analysis enables one to establish relationships that cannot be observed when outcome and predictor variables are analyzed separately. To deal with this type of data analysis, one variable is compared to another with the goal of finding out how one variable affects the other and various trends in the data samples are determined.

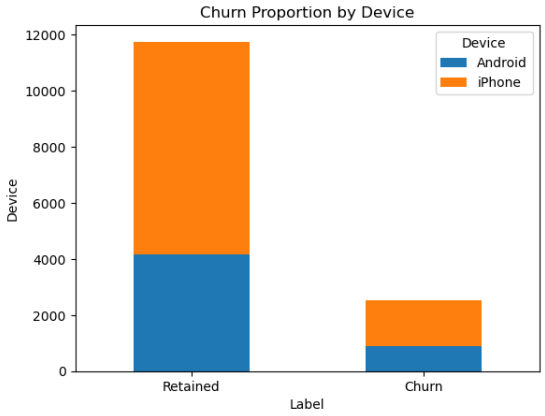

Figure 16: Churn proportion by Device

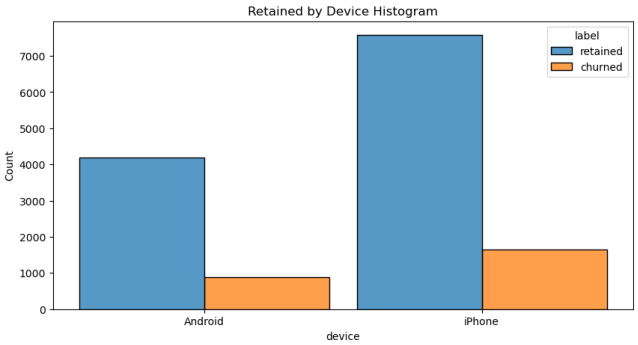

Figure 17: Histogram Churn proportion by Device.

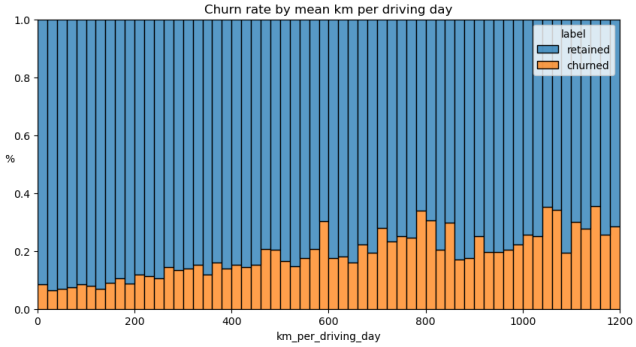

Figure 18: Churn rate by mean km per driving day

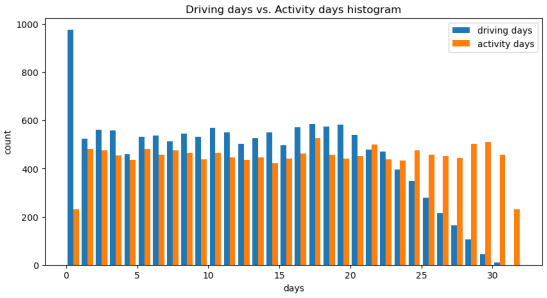

Figure 19: Histogram Driving days vs Activity days

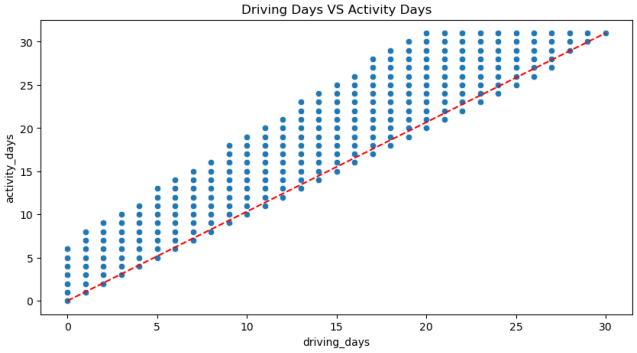

Figure 20: Driving days vs Activity days.

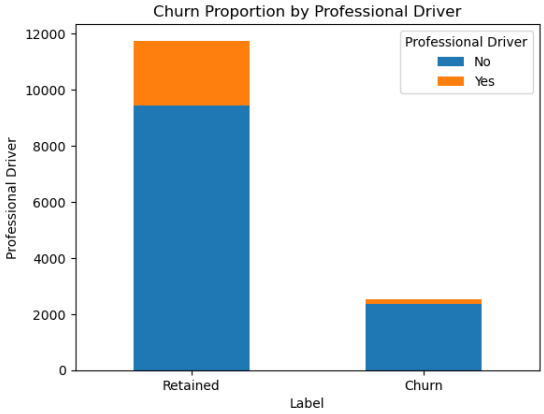

Figure 21: Churn proportion by Professional Driver

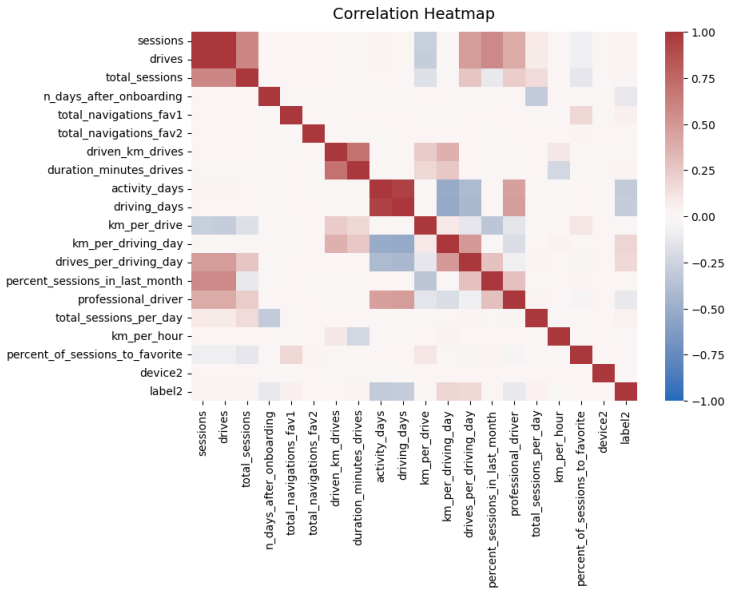

Figure 22: Correlation Matrix

As shown in the correlation matrix above, the coefficient being displayed exhibits the degree of the linear relationship between selected numerical attributes of the supermarket sales dataset. The matrix contains the correlation coefficients and each cell of the matrix contains the correlation coefficient of two variables, which can be the range of (-1, 1). For a stronger positive relationship which can be referred as a positive correlation coefficient is above 0 point, while for a stronger negative relationship of the negative correlation coefficient at below 0 point. Measurement closer to 0 indicates that there is no direct linear relationship between the two values being compared.

5 Data Preparation and Feature Engineering

In this section, we outline the steps taken to prepare the Waze dataset for the churn prediction task. Effective data preparation and feature engineering are crucial for building accurate and robust predictive models. We performed a series of preprocessing steps to clean and transform the data, ensuring it was suitable for input into deep learning models such as CNN, RNN, LSTM, and GRU.

5.1 Data Cleaning

In our data preparation process, we identified 700 observations with missing values, which accounted for less than 5% of the entire dataset. Given the relatively small proportion of missing data, we opted to use the dropna() method to

remove these rows. This approach ensures that our analysis is not significantly affected by missing information while maintaining the integrity and robustness of our dataset. By dropping these rows, we preserved the quality of the data fed into our models, avoiding potential biases that could arise from imputing or otherwise artificially modifying missing values.

5.2 Feature Engineering

As part of the feature engineering process, several new variables were created to enhance the predictive power of our models. These features were designed to capture more nuanced aspects of user behavior and engagement, which are critical for accurately predicting churn. Here is a breakdown of the variables created and other transformations applied:

- Kilometers per Drive: This feature is calculated as the total kilometers driven (driven_km_drives) divided by the number of drives (drives). It provides insight into the average distance covered in each drive, which can be an indicator of user driving habits.
- Kilometers per Driving Day: This variable is derived by dividing the total kilometers driven (driven _km_drives) by the number of driving days (driving_days). It helps in understanding the daily driving distance, capturing patterns of user engagement over time.
- Drives per Driving Day: This feature is computed by dividing the total number of drives (drives) by the number of driving days (driving_days). It reflects the average number of drives a user makes per day, which can indicate the user's driving frequency.
- Percentage of Sessions in the Last Month: This variable is calculated as the number of sessions in the last month (sessions) divided by the total number of sessions (total_sessions). It helps in identifying users who may have become less active recently, a potential indicator of churn.
- Professional Driver Indicator: This binary feature is created using a condition where users with at least 60 drives and at least 15 driving days are labeled as professional drivers (1), and others are labeled as non-professional drivers (0). This variable helps in distinguishing between regular and heavy users.
- Total Sessions per Day: This feature is computed by dividing the total number of sessions (total_sessions) by the number of days since onboarding (n_days_after_onboarding). It provides a measure of user engagement by showing the average number of sessions per day since the user started using the app.
- Kilometers per Hour: This variable is calculated as the total kilometers driven (driven_km_drives) divided by the total drive duration converted

to hours (duration_minutes_drives / 60). It represents the average speed of users, which can be related to driving behavior.

- Percentage of Sessions to Favorite Destinations : This feature is derived by summing the total navigations to the first and second favorite destinations (total navigations fav1 + total navigations fav2) and dividing by the total number of sessions (total sessions). It indicates the proportion of sessions that are directed towards frequent or favorite locations.

To facilitate model training, categorical variables were mapped to numerical values:

- Device Type Mapping (device2): The device type (device) was mapped to numerical values where 'Android' is represented by 0 and 'iPhone' is represented by 1. This transformation is necessary for models that do not handle categorical data directly.
- Churn Label Mapping (label2): The churn label (label) was mapped to numerical values with 'retained' represented by 0 and 'churned' represented by 1. This conversion simplifies the binary classification task.

To reduce dimensionality and eliminate irrelevant information, the ID attribute was removed from the dataset. The ID field is unique for each record and does not contribute to predicting churn, making it extraneous for model training. We used the drop method to remove this attribute.By creating these new features, mapping categorical variables, and removing unnecessary attributes, we enhanced the dataset's quality and relevance for the churn prediction task, enabling our deep learning models to capture complex patterns in user behavior more effectively.

5.3 Handling Outliers

In predictive modeling, outliers can greatly influence the performance of machine learning algorithms. Specifically, extreme values in certain features can skew the model, leading to poor generalization on unseen data. To address this issue, we employed an outlier handling technique that caps values exceeding the 95th percentile for each feature distribution. We defined a function that calculates the 95th percentile for each feature. For values that exceed this threshold, we replace them with the value at the 95th percentile, effectively capping extreme values. This method allows us to reduce the impact of outliers without entirely removing data points, preserving the overall distribution while mitigating distortion caused by extreme values. The technique was applied to the following features:

- sessions
- drives
- total_sessions

- driven_km_drives
- duration_minutes_drives
- total_navigations_fav1
- km_per_drive
- km_per_driving_day
- drives_per_driving_day
- total_sessions_per_day
- km_per_hour
- percent_of_sessions_to_favorite

For each of these features, the 95th percentile was calculated, and values exceeding this threshold were replaced with the value at the 95th percentile. This process helped to normalize the feature distributions, ensuring that extreme values did not disproportionately influence the model. The decision to impute rather than remove outliers ensures that important trends in the data are maintained while mitigating the risk of model overfitting due to rare but extreme cases.

By handling outliers in this manner, we improved the stability and generalization of our models, particularly in the context of predicting user churn for Waze.

5.4 Data Balancing

To address the class imbalance in our target variable, we used the Synthetic Minority Over-sampling Technique (SMOTE) to balance the dataset. Class imbalance, where one class significantly outnumbers the other, can lead to biased model performance, with the model favoring the majority class and failing to accurately predict the minority class. SMOTE helps to mitigate this issue by generating synthetic samples for the minority class. In our case, we set the sampling strategy parameter to 0.7, meaning that SMOTE increased the number of minority class samples to 70% of the majority class size. This approach created a more balanced dataset, ensuring that our deep learning models could learn to identify patterns in both retained and churned users more effectively. By using SMOTE, we aimed to enhance the model's predictive performance and robustness across both classes.

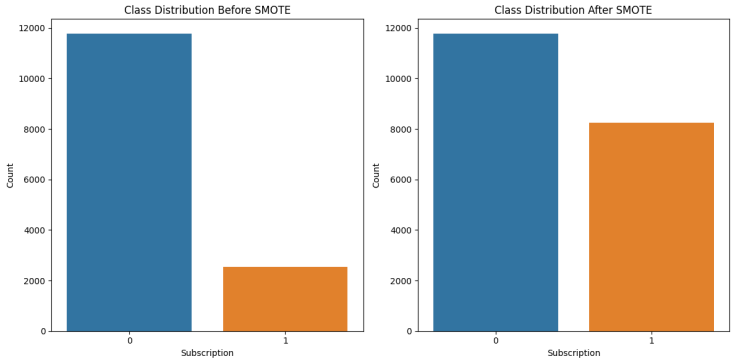

Figure 23: Before and After Outlier Balancing the Target Variable

5.5 Data Scaling

Before applying deep learning models, it was crucial to standardize the numerical features in the dataset to ensure consistent model performance. We used the StandardScaler from scikit-learn to transform these features so that they have a mean of 0 and a standard deviation of 1. This step is particularly important for deep learning models like CNN, RNN, LSTM, and GRU, which are sensitive to the scale of input data. Features with different scales can lead to models focusing disproportionately on larger-scaled variables, thereby skewing the learning process. By standardizing the data, we ensured that all features contributed equally during model training, leading to more stable and faster convergence. Additionally, normalization helped prevent gradient descent issues, such as exploding or vanishing gradients, by maintaining numerical stability during the training process. This preprocessing step ultimately improved the models' ability to learn from the data effectively and produce accurate churn predictions.

6 Machine Learning Models

In this section, we describe the application of two machine learning models—Random Forest and Extreme Gradient Boosting(XGBoost) to predict user churn for the Waze application. These models were chosen due to their high performance in handling tabular data and their ability to capture complex relationships between features.

6.1 Random Forest

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the mode of the classes (classification) or mean prediction (regression) from the individual trees. It reduces overfitting by averaging multiple decision trees, each trained on a random subset of data and features.

The model was tuned using GridSearchCV to optimize hyperparameters and improve its predictive performance. The following describes the key aspects of the model's development and evaluation:

1. **Model Selection**: A Random Forest classifier was chosen due to its robustness and ability to handle high-dimensional datasets effectively. By building an ensemble of decision trees, Random Forest reduces the risk of overfitting and provides a more generalized model.

2. **Hyperparameter Tuning**: To optimize the model's performance, we utilized GridSearchCV, which systematically searches over a predefined set of hyperparameters:

   - **Max Depth**: The maximum depth of the trees was set to unlimited (None), allowing trees to expand fully.
   - **Max Features**: The number of features considered for splitting at each node was set to 100% (max_features=1.0), allowing all features to be considered.
   - **Max Samples**: 100% of the dataset was used to build each tree (max_samples=1.0).
   - **Min Samples per Leaf**: A minimum of 2 samples was required in each leaf node, preventing the model from creating overly specific rules from small data points.
   - **Min Samples per Split**: A minimum of 2 samples was required to split a node, ensuring sufficient data in each split.
   - **Number of Trees** (n_estimators): The model was built using 300 trees, providing a strong ensemble that improves the overall stability and accuracy of the predictions.

3. **Scoring Metrics**: We evaluated the model on four key metrics: accuracy, precision, recall, and F1-score. While accuracy gives a general sense of how well the model classifies, we prioritized recall to minimize false negatives and capture as many true churners as possible. This is particularly important in churn prediction, where missing a potential churner could lead to higher customer attrition.

4. **Cross-Validation**: To ensure the model's reliability, we employed 4-fold cross-validation. The data was split into four subsets, with the model trained on three and tested on the remaining one. This process was repeated to ensure the model's performance was consistent across different data splits.

Model	**Precision**	**Recall**	**F1-Score**	**Accuracy**
RF CV	0.7622	0.7588	0.7605	0.7875

Table 2: Performance metrics of Random Forest Classifier (RF CV) for churn prediction

After training the Random Forest model, we evaluated its performance using cross-validation to ensure robust and generalizable results. Cross-validation helps to mitigate overfitting by splitting the dataset into multiple folds and averaging the model's performance across these folds.

- Precision: The Random Forest model correctly predicted churned users 76.2% of the time out of all users predicted to churn.
- Recall: The model identified 75.9% of all actual churned users.
- F1-Score: The harmonic mean of precision and recall, indicating a good balance between the two.
- Accuracy: The overall performance of the model, correctly predicting 78.75% of the cases.

These results suggest that the Random Forest model performs well in distinguishing between churned and non-churned users.

6.2 Extreme Gradient Boosting (XGBoost)

XGBoost is a powerful gradient boosting algorithm that has gained popularity for its high accuracy and speed in predictive modeling tasks. It works by building decision trees sequentially, where each new tree corrects errors made by the previous one. The model's objective is to minimize the error by combining weak learners (decision trees) into a strong one.

We utilized the XGBoost classifier to predict user churn. XGBoost was chosen due to its high performance and ability to handle complex datasets efficiently. The model was tuned using GridSearchCV, and the key aspects of the configuration are described below:

The following hyperparameters were fine-tuned during model training:

- **Max Depth**: Values of 6 and 12 were considered, where a deeper tree can model more complex interactions but may overfit the data.
- **Min Child Weight**: A range of 3 and 5 was tested. Higher values help prevent overfitting by requiring more data in each child node.
- **Learning Rate**: Two learning rates, 0.01 and 0.1, were tested. The learning rate determines how quickly the model adapts during training.
- **Number of Estimators**: The number of boosting rounds was set to 300, providing sufficient iterations for model convergence.

Scoring and Evaluation The model's performance was evaluated using accuracy, precision, recall, and F1-score. Since identifying churners was prioritized, the model was refit based on the highest recall score to minimize false negatives.

Cross-Validation To ensure the robustness of the model, 4-fold cross-validation was applied. This method splits the dataset into four parts, training on three while testing on the fourth, ensuring consistent performance across different data splits.

The XGBoost model, after hyperparameter tuning and cross-validation, yielded the following performance metrics

Model	Precision	Recall	F1-Score	Accuracy
XGB CV	0.7898	0.7880	0.7888	0.8125

Table 3: Performance metrics of Extreme Gradient Boosting for churn prediction

After training the XGBoost model, we have obteined this results:

- **Precision**: The XGBoost model achieved a precision of 0.7898, indicating that 78.98% of the users predicted to churn were actual churners. This suggests a high degree of accuracy in identifying true churners while minimizing false positives.
- **Recall**: With a recall of 0.7880, the model successfully identified 78.80% of the actual churners, demonstrating strong sensitivity to true churn cases. This is crucial in churn prediction, where missing a churner can result in loss of users.
- **F1-Score**: The F1-Score, which balances precision and recall, was 0.7888. This indicates that the model maintains a good trade-off between correctly identifying churners and minimizing false positives.
- **Accuracy**: The overall accuracy of the XGBoost model was 81.25%, meaning that the model correctly predicted 81.25% of both churners and non-churners.

7 Models Evaluation

In this section, we compare the performance of two machine learning models **XGBoost** and **Random Forest** used for predicting user churn in the Waze Churn Prediction project. Both models were optimized using GridSearchCV and evaluated on key performance metrics, including precision, recall, F1-score, and accuracy. The purpose of this comparison is to assess which model provides better predictive performance and is more suited for identifying potential churners, thus guiding the selection of the final model for deployment.

Figure 24: Models performance

Precision: XGBoost outperformed Random Forest in terms of precision (0.7898 vs 0.7622). This suggests that XGBoost was better at minimizing false positives and more effectively identified users that are actually at risk of churning.

Recall: Similarly, XGBoost had a higher recall (0.7880 vs 0.7588), meaning it captured more actual churners than Random Forest. This makes XGBoost more effective in identifying the at-risk users.

F1-Score: The F1-score for XGBoost (0.7888) was also higher than that of Random Forest (0.7605), indicating that XGBoost maintains a stronger balance between precision and recall.

Accuracy: XGBoost exhibited better accuracy, correctly predicting 81.25% of cases, compared to 78.75% for Random Forest. This shows that XGBoost provides a more generalizable model across the dataset.

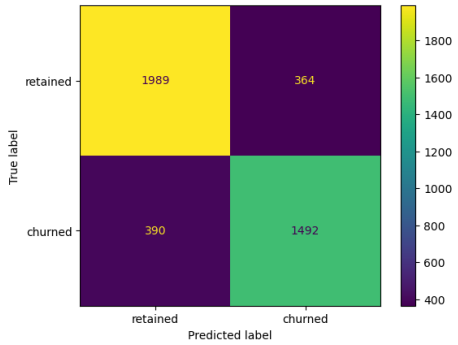

Figure 25: Confusion Matrix XGBoost

The confusion matrix visualizes the performance of the XGBoost model in predicting user churn. It provides a breakdown of how the model's predictions

compare to the actual labels (retained or churned). Here's how the results are interpreted:

- True Positives (1492): These are the cases where the model correctly predicted that users would churn. This means 1492 users who actually churned were correctly identified by the model.
- False Positives (364): The model incorrectly predicted that 364 users would churn when, in reality, they were retained. These are the cases where the model generated false alarms.
- True Negatives (1989): The model correctly identified 1989 users as retained, meaning these users were correctly predicted not to churn.
- False Negatives (390): These are users that the model predicted as retained, but they actually churned. These represent missed opportunities where the model failed to identify churners.

The model does well in identifying both retained and churned users, with relatively low false positives and false negatives. However, there are still 390 false negatives (missed churners) and 364 false positives (incorrectly predicted churners). The balance between correctly predicting churners and retained users indicates the model has a reasonable balance between precision and recall.

This graph represents the feature importance in the XGBoost model used for predicting user churn. The importance of each feature is determined by its contribution to the model's decisions, measured by the F-score (the number of times a feature is used to split the data across all trees in the model).

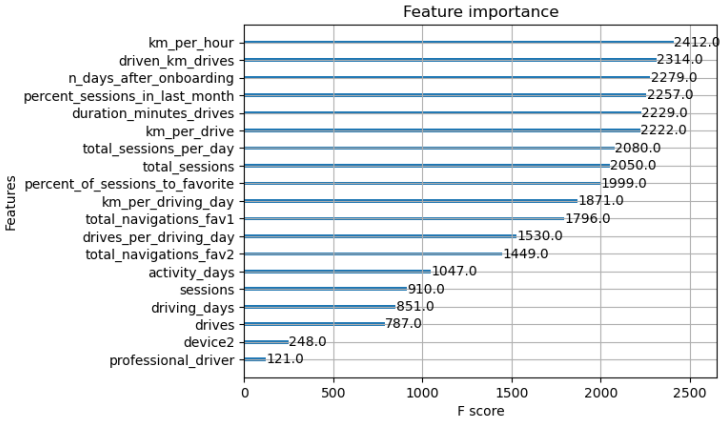

Figure 26: Feature Importance

The most important features in predicting churn in this model are primarily related to driving behavior, engagement with the app, and recency of activity. This aligns with the idea that churn is often driven by patterns of decreasing

usage or changes in user driving habits. Features like kilometers driven, recent session activity, and the duration of driving sessions are among the most predictive of whether a user is likely to churn. The least important features, such as device type or professional driver status, have a lower influence on the model's predictions.

8 Deep Learning Models

8.1 Convolution Neural Networks (CNN)

In this project, a Convolutional Neural Network (CNN) model was utilized to predict churn among Waze users. The CNN model architecture employed is a deep neural network designed specifically to process and analyze sequential data, making it suitable for the time-series nature of user activity logs.

8.1.1 Model Architecture

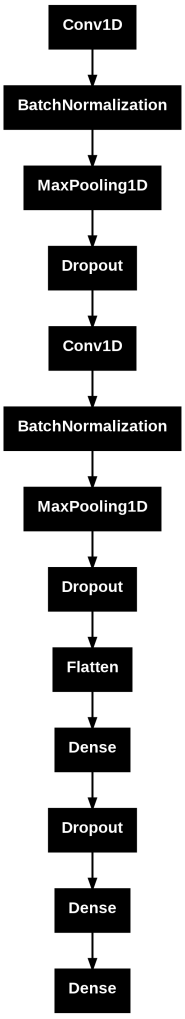

Figure 27: Architecture of CNN

The architecture consists of the following layers:

1. Conv1D Layer: The initial layer is a 1D convolutional layer that captures the essential features from the input sequence data.
2. BatchNormalization: This layer helps to stabilize and accelerate the training process by normalizing the activations of the previous layer.
3. MaxPooling1D: A pooling layer is used to reduce the spatial dimension of the data, helping to control overfitting and reduce computation.

4. Dropout: This layer randomly sets a fraction of input units to 0 at each update during training, which helps to prevent overfitting.
5. Conv1D Layer: Another convolutional layer to capture more complex patterns in the data.
6. BatchNormalization: Another normalization step to keep the model training stable.
7. MaxPooling1D: Additional pooling to further reduce dimensionality and focus on the most important features.
8. Dropout: Another dropout layer to prevent overfitting.
9. Flatten: This layer flattens the input, converting the 2D matrix into a vector, preparing it for the fully connected Dense layers.
10. Dense Layer: Fully connected layers to learn the complex interactions between features.
11. Dropout: Final dropout to prevent overfitting before the output layer.
12. Dense Layer: The final dense layer that outputs the prediction.

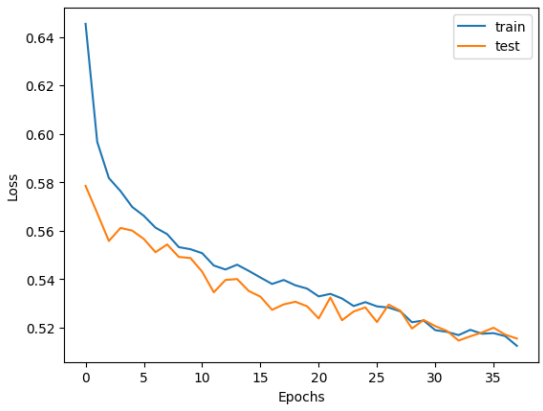

Figure 28: CNN loss on training/validation set

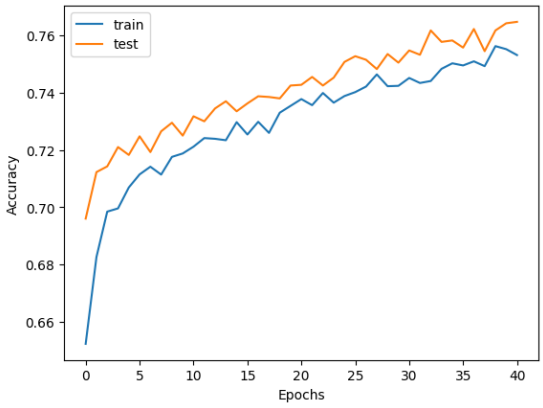

Figure 29: CNN accuracy on training/validation set

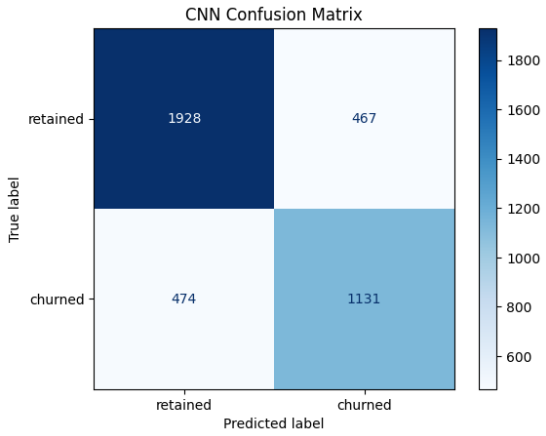

Figure 30: CNN Confusion Matrix

The model's accuracy of 76.5% indicates a good overall ability to correctly classify churn versus non-churn users. The F1-score of approximately 0.70 suggests a balanced performance between precision and recall, making the model effective at identifying churners while minimizing false positives. The AUC-ROC score of 0.754 highlights the model's strong discriminative power, effectively distinguishing between churners and non-churners.The CNN model provided a robust approach to predicting user churn in the Waze dataset. Its architecture allowed it to effectively capture patterns in user behavior, and the results suggest that it could be a valuable tool in understanding and predicting churn, potentially enabling proactive measures to retain users.

8.2 Recurrent Neural Networks (RNN)

In this project, another model that was implemented to predict churn among Waze users. The Recurrent Neural Network (RNN) architecture is particularly effective for sequential data, such as time-series data or logs of user interactions, which makes it a suitable choice for this churn prediction task.

8.2.1 Model Architecture

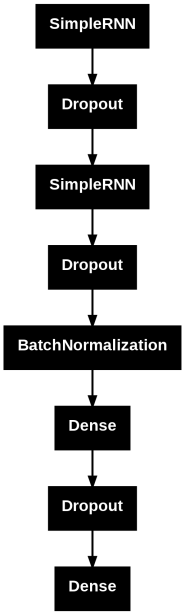

Figure 31: RNN Architecture

The architecture of the RNN model consists of the following layers:

1. SimpleRNN Layer: The model starts with a SimpleRNN layer, which captures sequential dependencies in the data. RNNs are capable of maintaining a memory of previous inputs, making them effective in handling sequence prediction tasks.

2. Dropout: To prevent overfitting, a dropout layer is used after each SimpleRNN layer. This layer randomly sets a fraction of input units to zero at each update during training, which helps to regularize the model.

3. SimpleRNN Layer: Another SimpleRNN layer is added to further capture sequential patterns and dependencies in the data.

4. Dropout: Again, a dropout layer follows to reduce overfitting.

5. BatchNormalization: This layer is included to normalize the activations from the previous layer, which can speed up the training process and improve the model's stability.

6. Dense Layer: A fully connected layer is added to learn the complex relationships between the features.

7. Dropout: Another dropout layer is included to prevent overfitting before the output layer.

8. Dense Layer: The final dense layer outputs the prediction for whether a user is likely to churn or not.

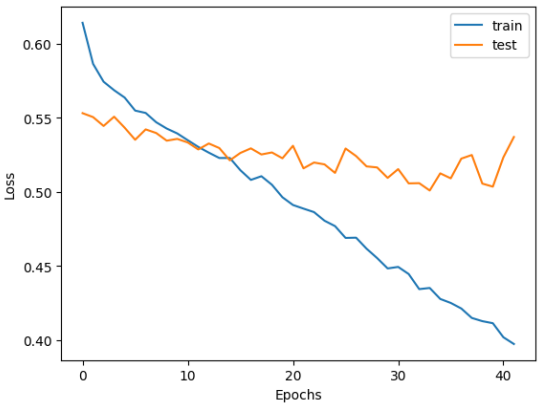

Figure 32: RNN loss on training/validation set

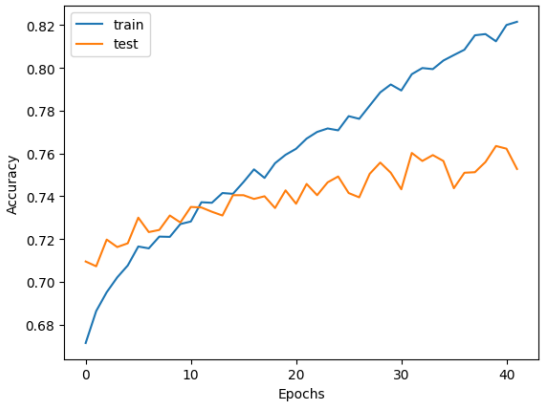

Figure 33: RNN accuracy on training/validation set

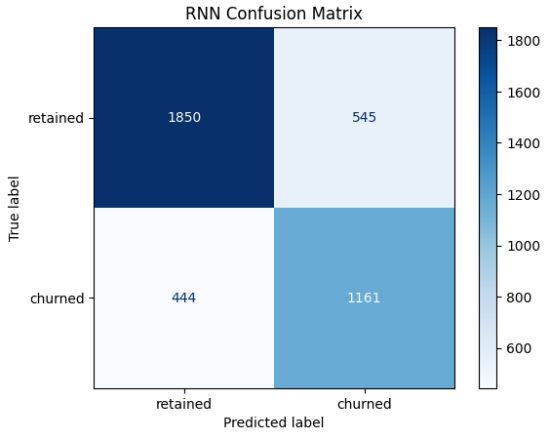

Figure 34: RNN Confusion Matrix

These results indicate that the RNN model performed well in predicting churn among Waze users. The accuracy of approximately 75.27% suggests that the model correctly identified a significant portion of churn cases. The precision

value of 0.68 indicates that when the model predicts a user is likely to churn, it is correct about 68% of the time. The recall value of 0.723 suggests that the model successfully identifies about 72.3% of the actual churners. The F1-score, which balances precision and recall, is 0.70, indicating a good trade-off between the two. The AUC-ROC score of 0.747 demonstrates the model's ability to distinguish between churners and non-churners effectively.

8.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) model was utilized to enhance the ability to capture long-term dependencies in the user behavior data. LSTMs are an advanced type of Recurrent Neural Network (RNN) designed specifically to overcome the limitations of traditional RNNs, especially in learning and retaining information over longer sequences.

8.3.1 Model Architecture

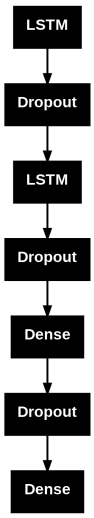

Figure 35: LSTM Architecture

The LSTM model was designed with the following layers:

1. LSTM Layer: The model begins with an LSTM layer, which is adept at learning long-term dependencies in sequential data. LSTMs incorporate mechanisms such as forget gates and memory cells, which allow them to maintain and adjust the flow of information over time effectively.

2. Dropout: To mitigate overfitting, dropout layers are introduced after the LSTM layers. This helps by randomly setting a fraction of input units to zero during training, thereby promoting model generalization.

3. LSTM Layer: Another LSTM layer follows to further refine the learning of sequential patterns and dependencies in the data.

4. Dropout: Another dropout layer is applied to reduce the risk of overfitting.

5. Dense Layer: A fully connected layer is added to capture complex relationships and interactions within the features after the sequence processing by the LSTM layers.

6. Dropout: This dropout layer is applied before the output layer to ensure that the model does not overfit to the training data.

7. Dense Layer: The final dense layer outputs the prediction, determining whether a user is likely to churn.

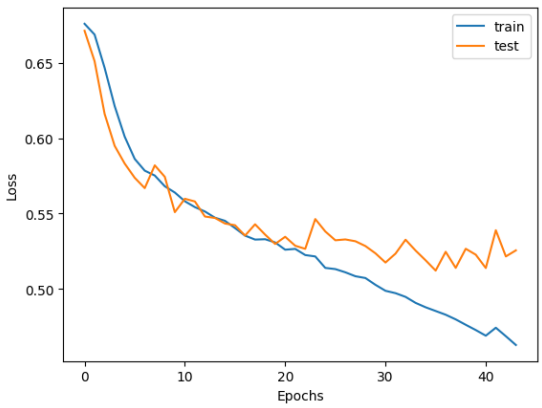

Figure 36: LSTM loss on training/validation set

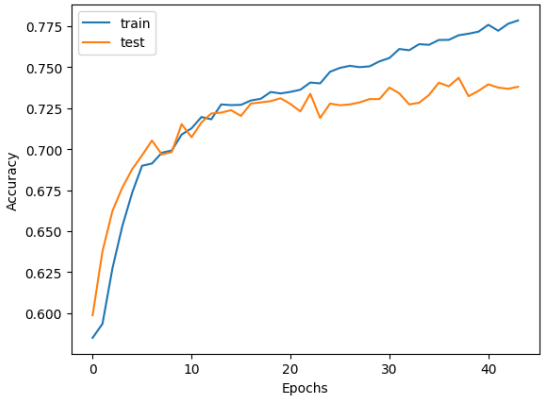

Figure 37: LSTM accuracy on training/validation set

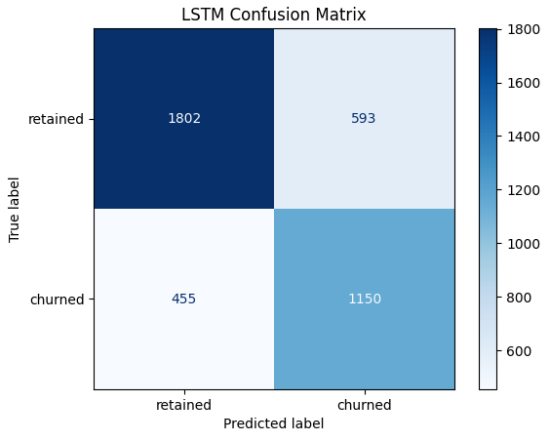

Figure 38: LSTM Confusion Matrix

These results demonstrate the LSTM model's ability to effectively predict churn among Waze users. The accuracy of approximately 73.8% indicates that the model correctly identified a significant portion of churn cases. The precision value of 0.659 shows that when the model predicts a user is likely to churn, it is correct about 65.9% of the time. The recall value of 0.716 suggests that the model successfully identifies around 71.6% of the actual churners. The F1-score, which balances precision and recall, is 0.686, indicating a solid trade-off between these two metrics. The AUC-ROC score of 0.734 reflects the model's competence in distinguishing between churners and non-churners.

8.4 Gated Recurrent Unit (GRU)

We implemented a Gated Recurrent Unit (GRU) model to predict user churn for the Waze dataset.

8.4.1 Model Architecture

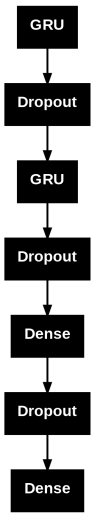

Figure 39: GRU Architecture

The GRU architecture, as depicted in the provided diagram, consists of two GRU layers, each followed by a Dropout layer to prevent overfitting by randomly setting a fraction of input units to zero at each update during training. After the GRU layers, we included Dense layers with Dropout in between to further enhance the model's generalization capability. The final Dense layer outputs the predicted class probabilities.

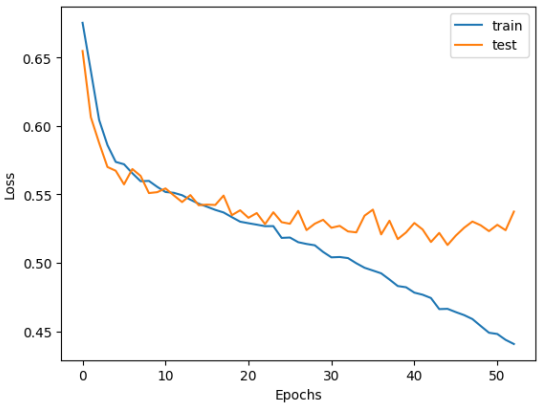

Figure 40: GRU loss on training/validation set

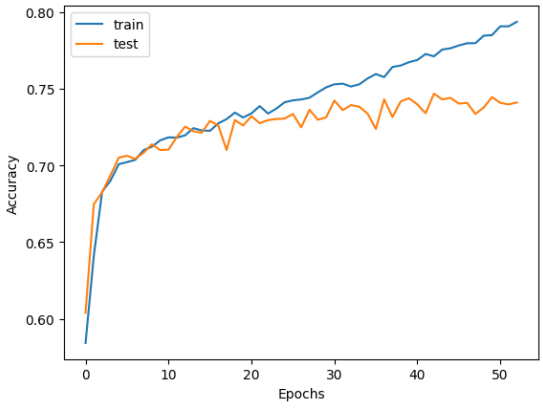

Figure 41: GRU accuracy on training/validation set

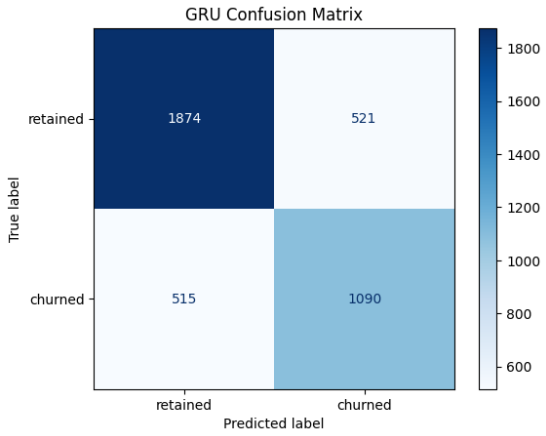

Figure 42: GRU Confusion Matrix

The GRU model achieved an accuracy of 74.10%, indicating that the model correctly predicted user churn and retention in 74.10% of the cases. The precision score of 67.65% shows that when the model predicted a user would churn, it was correct 67.65% of the time. The recall score of 67.91% reveals that the model was able to identify 67.91% of all actual churned users. The F1-score of 67.78% represents a balance between precision and recall, suggesting that the model has a good balance of correctly identifying both true positives and minimizing false positives. Additionally, the AUC-ROC score of 73.07% indicates a strong ability to distinguish between churned and retained users. These results demonstrate that the GRU model is effective in capturing temporal patterns in user behavior, making it a suitable choice for the churn prediction task in this project.

9 Models Evaluation

Based on the evaluation metrics and ROC curves for the different models applied in the Waze churn prediction project, we can assess the performance of each model to determine the best one for the task. The models evaluated include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory Networks (LSTM), and Gated Recurrent Units (GRU).

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
CNN	0.76475	0.707760	0.704673	0.706213	0.754842
RNN	0.75275	0.680539	0.723364	0.701299	0.747904
LSTM	0.73800	0.659782	0.716511	0.686977	0.734456
GRU	0.74100	0.676598	0.679128	0.677861	0.730796

Table 4: Performance metrics of different models

The CNN model achieved the highest overall performance with an accuracy of 76.47% and an AUC-ROC of 0.75. These metrics indicate that the CNN model had the best balance of true positive and false positive rates among all models, making it the most effective at distinguishing between churned and retained users. It also scored well in precision (70.78%) and recall (70.47%), resulting in a solid F1-score of 70.62%, indicating a good balance between precision and recall.

The RNN model performed well, with an accuracy of 75.28% and an AUC-ROC of 0.75. It excelled in recall, achieving the highest recall score of 72.34% among all models, meaning it was particularly good at identifying users likely to churn. However, its precision (68.05%) was slightly lower than that of the CNN model, leading to an F1-score of 70.13%.

The GRU model also delivered competitive performance, with an accuracy of 74.10% and an AUC-ROC of 0.73. Its precision (67.66%) and recall (67.91%) were balanced, resulting in an F1-score of 67.79%. While its performance was slightly lower than that of the RNN, it still achieved solid results.

The LSTM model had the lowest accuracy at 73.80% and an AUC-ROC of 0.73. Its precision was 65.98%, with a recall of 71.65%, resulting in an F1-score of 68.70%. This suggests that while the LSTM model was reasonably precise, it was slightly less effective than the other models at distinguishing churn cases.

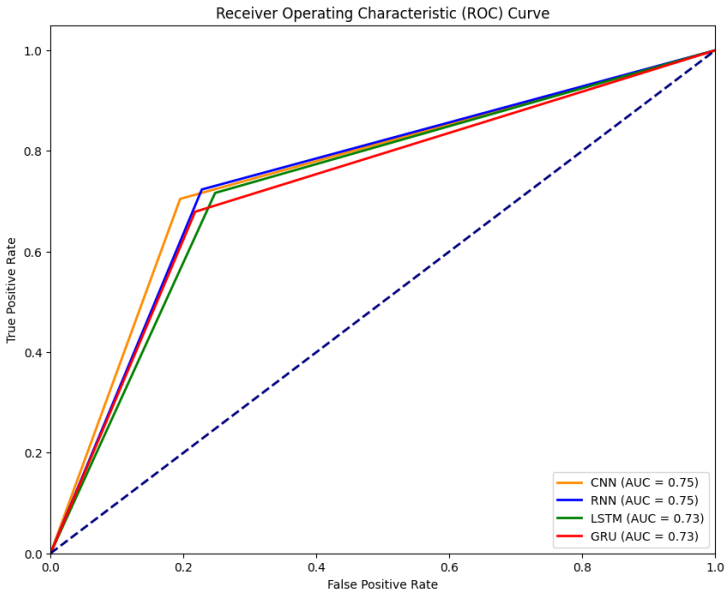

Figure 43: Models Evaluations

Overall, the CNN model is the best performer for this churn prediction task, given its highest AUC-ROC, accuracy, and balanced performance across precision, recall, and F1-score. The GRU model also shows strong performance,

particularly in recall, making it a good alternative if the goal is to minimize missed churn predictions.

10 Summary Key Findings and Insights

In this study, we evaluated four deep learning models: CNN, RNN, LSTM, and GRU using a Waze Churn Prediction dataset. The performance of each model was assessed based on several metrics: Accuracy, Precision, Recall, F1-Score, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC).

1. Overall Model Performance:
    - CNN emerged as the best-performing model in terms of Accuracy (0.76475), Precision (0.7078), and AUC-ROC (0.7548). The model achieved a balanced performance across all metrics, making it the most reliable choice for this specific prediction task.
    - RNN was the second-best model, with an Accuracy of 0.75275 and the highest Recall (0.7234). The RNN model performed well, particularly in capturing the positive class, which is crucial in churn prediction.
    - GRU and LSTM both achieved similar levels of Accuracy (0.74100 and 0.73800, respectively) but exhibited slight differences in other metrics. The GRU had a higher F1-Score (0.6779) compared to LSTM (0.6870), indicating a better balance between Precision and Recall for GRU.
2. Precision and Recall Analysis:
    - The CNN model had the highest Precision (0.7078), meaning it was more accurate in predicting the positive class compared to the other models. This is critical for reducing false positives, which is valuable in business scenarios like churn prediction.
    - The RNN model had the highest Recall (0.7234), indicating it was more effective at identifying all actual positive cases, reducing false negatives. This is particularly beneficial when the cost of missing a churn prediction is high.
3. AUC-ROC Curve Analysis:
    - The AUC-ROC values provide an overall measure of each model's ability to distinguish between the positive and negative classes. The CNN model had the highest AUC-ROC score (0.7548), followed closely by RNN (0.7479). The GRU and LSTM models had slightly lower AUC-ROC scores (0.7308 and 0.7345, respectively).
4. Trade-offs and Model Selection:

- The CNN model stands out as the top performer overall, making it the best candidate for deployment. However, the RNN model is particularly strong in Recall, suggesting it might be preferable in scenarios where capturing as many true positives as possible is critical.
- The GRU and LSTM models also provide robust performance and could be considered depending on specific business needs, especially if the model's interpretability or computational efficiency is a priority.

11 Conclusion and Suggestions

11.1 Conclusions

This study explored the effectiveness of four deep learning models: CNN, RNN, LSTM, and GRU as well as two machine learning algorithms: Random Forest and XGBoost for predicting churn in the Waze dataset. The performance of each model and algorithm was thoroughly evaluated using key metrics such as Accuracy, Precision, Recall, F1-Score, and AUC-ROC.

The CNN model emerged as the best overall performer among the deep learning models, achieving the highest scores in Accuracy (0.76475), Precision (0.7078), and AUC-ROC (0.7548). This indicates that CNN is the most reliable model for this specific churn prediction task, offering a balanced trade-off between correctly identifying churners and minimizing false positives.

Among the machine learning algorithms, XGBoost outperformed Random Forest, achieving a Precision of 0.7898, Recall of 0.7880, F1-Score of 0.7888, and Accuracy of 0.8125. XGBoost's superior performance highlights its effectiveness in this prediction task, surpassing Random Forest in all evaluated metrics.

The RNN model, while slightly lower in overall Accuracy, excelled in Recall (0.7234), making it an excellent choice in scenarios where capturing as many true churners as possible is crucial. This model's performance highlights its potential in situations where the cost of missed churn predictions (false negatives) is high.

The GRU and LSTM models also demonstrated competitive performance, with slight variations in F1-Score and other metrics. These models could be considered viable alternatives depending on the specific needs of the business, particularly if there are constraints related to interpretability or computational resources.

11.2 Suggestions

Deployment of the XGBoost Model.-Given its superior performance across multiple metrics, it is recommended to deploy the XGBoost model for churn prediction in the current scenario. This model's high Precision, Recall, and Accuracy make it a reliable choice for effectively identifying churners.

Consideration of RNN in High-Risk Scenarios.-For cases where the cost of false negatives is particularly high, such as when the business impact of

losing a customer is significant, deploying the RNN model could be advantageous. Its higher Recall rate ensures that more potential churners are correctly identified.

Further Model Tuning and Testing.-While the current models and algorithms perform well, there may be room for further improvement through hyperparameter tuning or by exploring hybrid models that combine the strengths of different architectures. Regular testing and validation on updated datasets are recommended to maintain model accuracy and relevance over time.

Integration with Business Strategies.-The insights gained from these models should be integrated into broader business strategies. For example, the XGBoost model's predictions can be used to tailor customer retention campaigns, while the RNN model's focus on high Recall can help prioritize interventions for customers most at risk of churning.

Continuous Monitoring and Updating.-Churn patterns can evolve, and it is essential to continuously monitor model performance and update the models with new data to ensure ongoing accuracy. Incorporating feedback loops where the model's predictions are compared against actual outcomes can also help in refining the models over time.