Kevin Amaya - Junior Data Analyst

Waze User Churn Analysis

Understanding User Behavior to Reduce Monthly Churn

25 May, 2025

Problem Overview

Objective:

Predict monthly user churn in the Waze app based on behavioral data.

Approach:

- Exploratory Data Analysis (EDA)
- Statistical testing
- Machine Learning modeling
- Actionable insights and recommendations

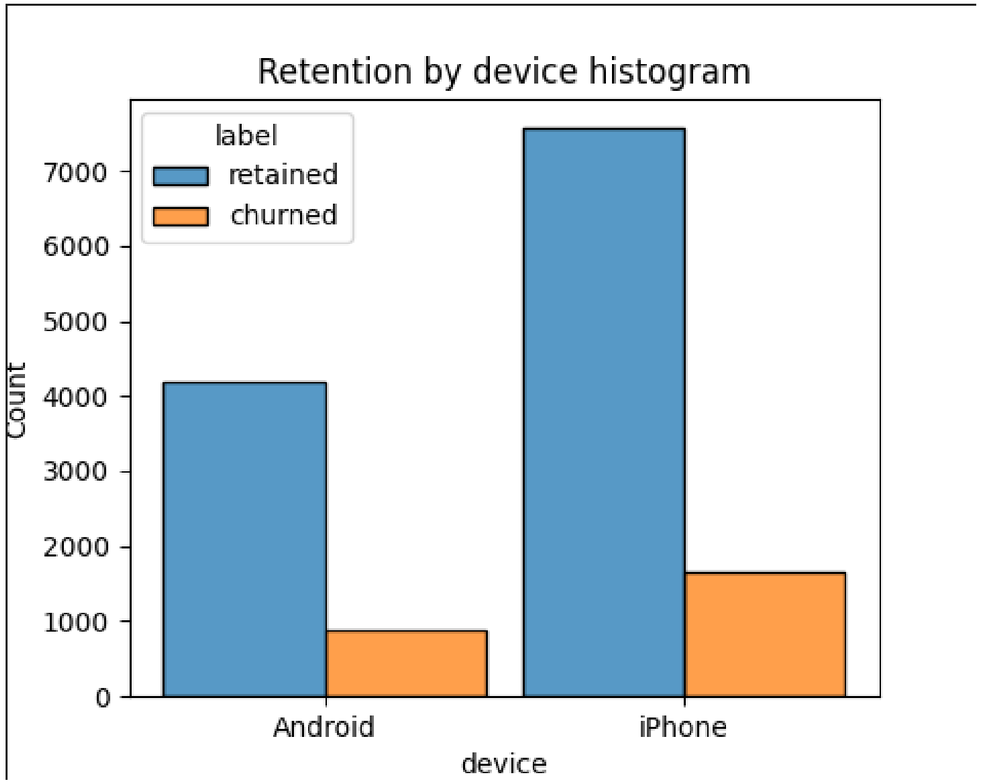
Dataset Summary

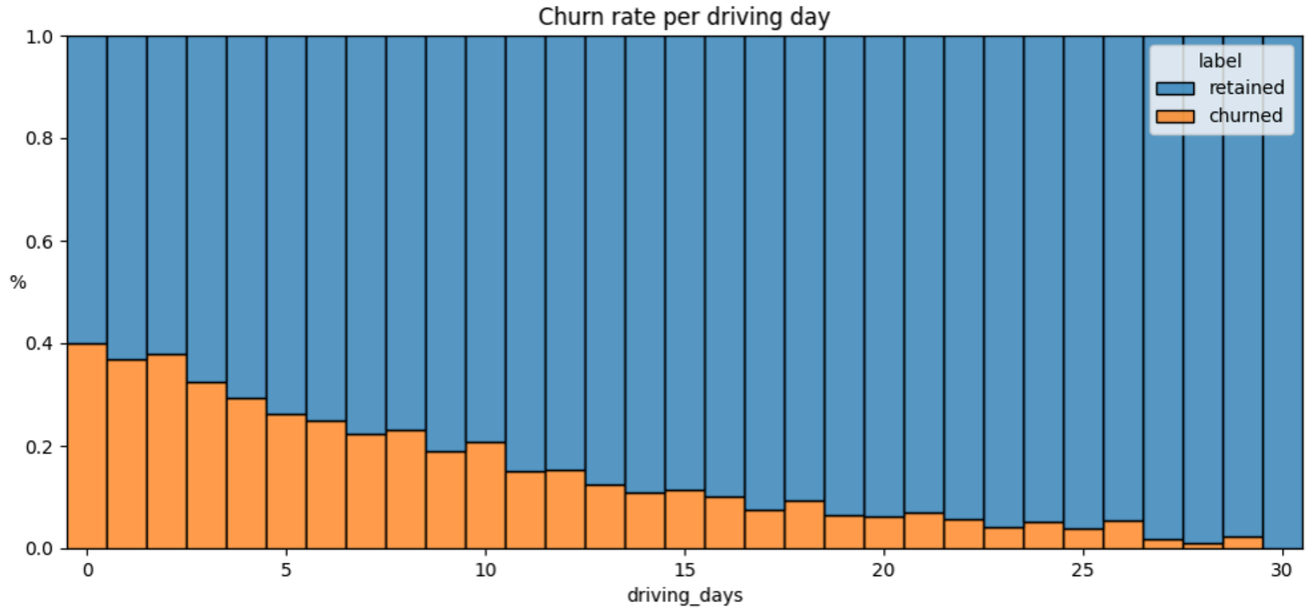
Source: Simulated Waze app usage data

Size: 5,000 users, 3 months

Key features:

- activity_days, driving_distance, user_type, car_type
- monthly_churn (target)
- Limitations:
- Missing churn labels
- Sparse feature detail
- Potential outliers

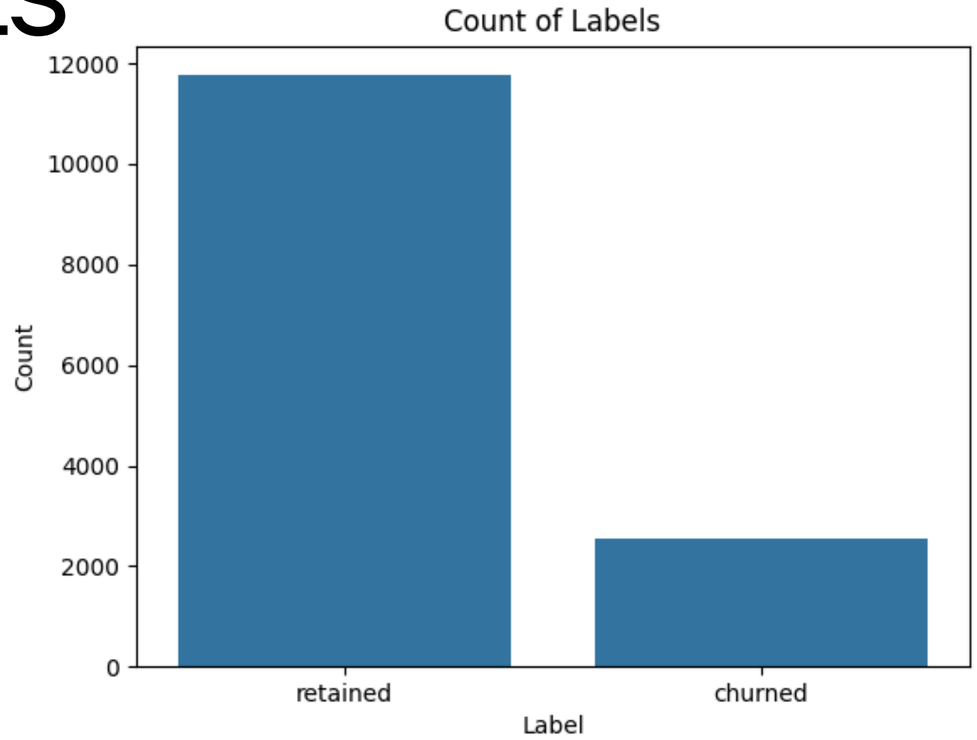


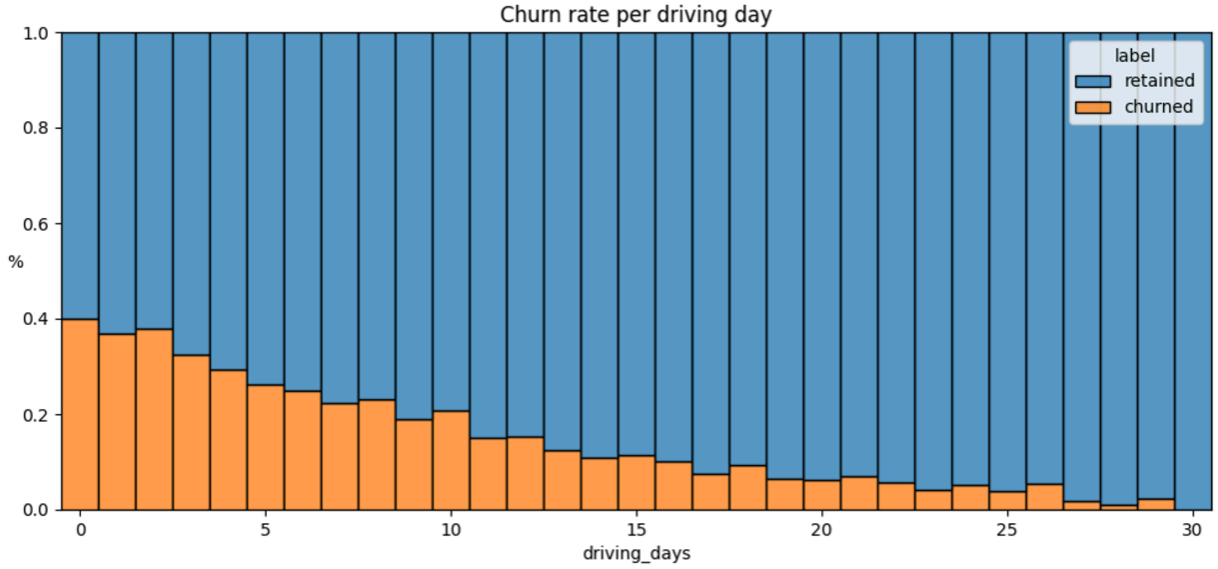


EDA Insights

Insights:

- Churn rate ~33% (after cleaning)
- Users who churn drive more per day but use the app fewer days
- Clear behavioral divide between retained and churned users
- Visuals:
- Include 2 plots:
- Boxplot: driving_distance vs monthly_churn
- Histogram: activity_days by churn status





Statistical Testing

- Tests Performed:
- Independent t-tests
- Pearson correlation
- Key Results:
- activity_days and monthly_churn: Strong negative correlation
- driving_distance: Higher for churned users
- user_type, car_type: No significant impact on churn

Machine Learning Models

```
# Calculate recall manually
recall = cm[1,1] / (cm[1, 0] + cm[1, 1])
recall
```

✓ 0.0s

```
np.float64(0.08990536277602523)
```

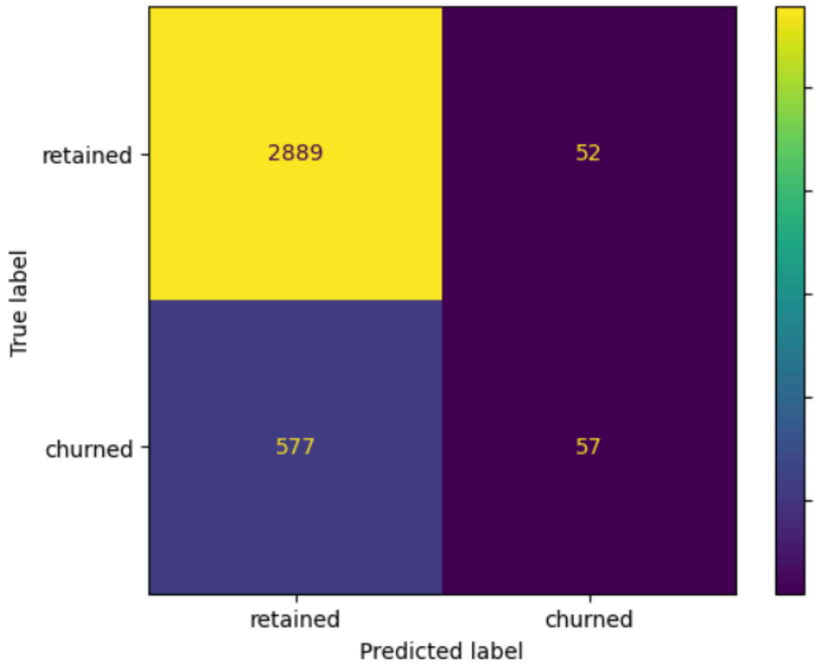
```
# Create a classification report
target_labels = ['retained', 'churned']
print(classification_report(y_test, y_preds, target_names=target_labels))
```

✓ 0.0s

```
              precision    recall  f1-score   support

    retained       0.83      0.98      0.90      2941
     churned       0.52      0.09      0.15       634

    accuracy                           0.82      3575
   macro avg       0.68      0.54      0.53      3575
weighted avg       0.78      0.82      0.77      3575
```



Models Tested:

- Logistic Regression
- Random Forest Classifier
- XGBoost Classifier
- Results:
- Best model: Logistic Regression
- Accuracy: ~66%, Precision: 72%, Recall: 9%
- Interpretation:
- Models struggled to generalize due to limited features
- Most churners were not captured by models

Machine Learning Models

```python
def get_test_scores(model_name:str, preds, y_test_data):
    '''
    Generate a table of test scores.

    In:
        model_name (string): Your choice: how the model will be named in the output table
        preds: numpy array of test predictions
        y_test_data: numpy array of y_test data

    Out:
        table: a pandas df of precision, recall, f1, and accuracy scores for your model
    '''
    accuracy = accuracy_score(y_test_data, preds)
    precision = precision_score(y_test_data, preds)
    recall = recall_score(y_test_data, preds)
    f1 = f1_score(y_test_data, preds)

    table = pd.DataFrame({'model': [model_name],
                          'precision': [precision],
                          'recall': [recall],
                          'F1': [f1],
                          'accuracy': [accuracy]
                          })

    return table
```

✓ 0.0s

```python
xgb_cv_results = make_results('XGB cv', xgb_cv, 'recall')
results = pd.concat([results, xgb_cv_results], axis=0)
results
```

✓ 0.0s

	model	precision	recall	F1	accuracy
0	RF cv	0.457163	0.126782	0.198445	0.818510
0	XGB cv	0.424500	0.179364	0.251974	0.811167

```python
# Use random forest model to predict on validation data
rf_val_preds = rf_cv.best_estimator_.predict(X_val)
```

✓ 0.1s

```python
# Get validation scores for RF model
rf_val_scores = get_test_scores('RF val', rf_val_preds, y_val)

# Append to the results table
results = pd.concat([results, rf_val_scores], axis=0)
results
```

✓ 0.0s

	model	precision	recall	F1	accuracy
0	RF cv	0.457163	0.126782	0.198445	0.818510
0	XGB cv	0.424500	0.179364	0.251974	0.811167
0	RF val	0.445255	0.120316	0.189441	0.817483

```python
# Use XGBoost model to predict on validation data
xgb_val_preds = xgb_cv.best_estimator_.predict(X_val)

# Get validation scores for XGBoost model
xgb_val_scores = get_test_scores('XGB val', xgb_val_preds, y_val)

# Append to the results table
results = pd.concat([results, xgb_val_scores], axis=0)
results
```

✓ 0.0s

	model	precision	recall	F1	accuracy
0	RF cv	0.457163	0.126782	0.198445	0.818510
0	XGB cv	0.424500	0.179364	0.251974	0.811167
0	RF val	0.445255	0.120316	0.189441	0.817483
0	XGB val	0.395556	0.175542	0.243169	0.806294

- The efficacy of a binomial logistic regression model is determined by accuracy, precision, and recall scores; in particular, recall is essential to this model as it shows the number of churned users.
- The model has mediocre precision (53% of its positive predictions are correct) but very low recall, with only 9% of churned users identified. This means the model makes a lot of false negative predictions and fails to capture users who will churn.
- Activity_days was by far the most important feature in the model. It had a negative correlation with user churn.

Key Feature Importance

Top Predictors:

- activity_days
- driving_distance
- Observation:
- More active users are far less likely to churn
- Visuals:
- Bar chart: Feature importance from XGBoost or Random Forest

Thank You

- 587-893-7400
- kevin.amayav@outlook.com
- https://www.linkedin.com/in/kevin-amayav/