Predicting Monthly User Churn in the Waze App

Author: Ronak Fathi
Objective: To identify behavioral indicators of user churn and develop a predictive model to support user retention strategies for Waze.

---

1. Introduction

Waze is a widely used free navigation app that helps drivers navigate efficiently. Like many digital products, Waze's success depends heavily on user retention. Churn, defined as users who uninstall or stop using the app, represents a key business challenge. This project builds a machine learning pipeline to predict churned users on the Waze app using user activity data, helping improve retention by identifying high-risk segments. We developed and evaluated several models, ultimately finding that XGBoost delivered the best performance.

---

2. Dataset Summary

The dataset consists of ~15,000 Users and 13 original features such as session counts, kilometers driven, device type, and activity days. The target variable was binary: churned (1) vs. retained (0).

Key Features:

- session: Number of times the app was opened
- device_type: Android or iPhone
- n_days_after_onboarding: Days since user onboarding
- total_navigations_fav1/fav2: Navigations to favorite locations
- driven_km_drives: Total kilometers driven
- duration_minutes_drives: Total drive time
- activity_days: Days user was active
- driving_days: Days user drove >1 km

We also engineered features such as:

- `km_per_driving_day`
- `percent_sessions_in_last_month`
- `km_per_drive`
- `professional_driver` (churn rate: 7.6% vs. 19.9%)

Churn Distribution: ~18% churned, 82% retained

---

3. Exploratory Data Analysis

In [ ]:
```
#Calculatte counts of churned vs. retained
print(df['label'].value_counts())
print()
print(df['label'].value_counts(normalize=True))
```

---

4. Modeling Approach

We trained and compared Logistic Regression, Random Forest, and XGBoost models. The target was imbalanced, so recall was especially important in assessing performance. Stratified 80/20 train-test split was used.

XGBoost Model Training and Evaluation

In [ ]:
```
#packages for Logistic Regression & Confusion Matrix
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, precision_score, \
recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
```

In [ ]:
```
#create a classification report
target_labels=['retained', 'churned']
```

```
print(classification_report(y_test, y_preds, target_names=target_labels))
```

In [ ]:
```
#This lets us see all of the columns, preventing Jupyter from redacting them.
pd.set_option('display.max_columns', None)

#Inport packages for data modeling
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import accuracy_score, precision_score, recall_score,\
f1_score, confusion_matrix, ConfusionMatrixDisplay, RocCurveDisplay, PrecisionRecallDisplay

from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

#This is the function that helps plot feature importance
from xgboost import plot_importance

#This module lets us save our models once we fit them.
import pickle
```

In [ ]:
```
#instantiate the XGBoost classifier
xgb = XGBClassifier(objective='binary:logistic', random_state=42)

#create a dictionary of hyperparameters to tune
cv_params= {'max_depth': [6, 12],
             'min_child_weight': [3, 5],
             'learning_rate': [0.01, 0.1],
             'n_estimators': [300]
           }

#define a dictionary of scoring metrics to capture
scoring = {'accuracy', 'precision', 'recall', 'f1'}

#instantiate the GridSearchCV object
xgb_cv = GridSearchCV(xgb, cv_params, scoring=scoring, cv=4, refit='recall')
```

Feature Importance

In [ ]:
```
#This lets us see all of the columns, preventing Jupyter from redacting them.
pd.set_option('display.max_columns', None)

#Inport packages for data modeling
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import accuracy_score, precision_score, recall_score,\
f1_score, confusion_matrix, ConfusionMatrixDisplay, RocCurveDisplay, PrecisionRecallDisplay

from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

#This is the function that helps plot feature importance
from xgboost import plot_importance

#This module lets us save our models once we fit them.
import pickle
```

---

5. Results and Conclusion

- **Best Model**: XGBoost (Accuracy: 81%, Precision: 42%, Recall: 17%)
- Feature engineering significantly improved model performance.
- Professional drivers had significantly lower churn rates.

Limitations

- The model had a low recall, detecting only 16.6% of churners.
- Further improvements could include better class balancing and domain-specific feature creation.

This analysis demonstrates the potential of behavioral modeling for churn prediction and offers a framework for future experimentation and model tuning.

Tools Used: Python, pandas, scikit-learn, xgboost, seaborn, matplotlib

GitHub Repository: Waze