**Maged Eid - Customer Churn Prediction Project**

**السلام عليكم – Hello! 👋**

In my solution, I built a machine learning model to predict customer churn **for a music streaming platform**. My main goal is to identify users who are likely to cancel their subscriptions based on their activity behavior. **All the necessary requirements are provided in this document and in the Jupyter Notebook uploaded to my GitHub.**  
The solution includes data preprocessing, feature engineering, model training, evaluation, API deployment, scheduled retraining, and Docker packaging.

**My GitHub**: <https://github.com/Maged-Ibrahim/Thmanyah---Maged-Eid>

**Business Impact:**

* Retaining users reduces revenue loss.
* Enables personalized interventions (offers, notifications) to reduce churn.
* Provides insights into user behavior patterns that correlate with churn.

**Challenges:**

1. **Class imbalance (عدم توازن الفئات)** – churners are initially fewer than active users.
2. **Difficulty defining churn (صعوبة تحديد الانسحاب)** – users might pause instead of cancel.
3. **Potential data leakage (احتمالية تسرب البيانات)** – timestamp and session features could inadvertently reveal churn.
4. **Missing Data** – song-related data is missing in ~20% of records.

**Features Implemented**

* **Exploratory Data Analysis (EDA)**: Missing values, distributions, categorical analysis, and churn distribution visualization.
* **Feature Engineering**: Aggregated user-level metrics (sessions, songs, logins, playlists, days since registration, gender).
* **Class Imbalance Handling**: Using **SMOTE** to balance the churn classes.
* **Machine Learning Models**:
  + Random Forest (for comparison)
  + XGBoost (final model)
* **Evaluation Metrics**: Classification report, confusion matrix, ROC-AUC.
* **MLflow Tracking**: Logs parameters, metrics, and models for reproducibility.
* **Scheduled Retraining**: Using **APScheduler** to update the model weekly.
* **Data Drift Monitoring**: Checks for significant changes in input features.
* **FastAPI**: Serves predictions via an API endpoint.
* **Docker**: Containerized project for consistent deployment.

**Project Structure**

customer\_churn\_project/

│

├─ notebook.ipynb # Complete notebook with EDA, model, API

├─ requirements.txt # Python dependencies

├─ Dockerfile # Docker setup for API

└─ README.md # Project documentation

**Findings from the Data – (I used customer\_churn full size file and not the mini) (البيانات)**

**Source:** Event logs of user activity (سجلات الأحداث).  
**Size:** 543,705 rows × 18 columns.

**Columns include:**

* userId, sessionId, page, auth, level, artist, song, length, location, gender, registration, userAgent, firstName, lastName.

**Missing Data Overview:**

|  |  |  |
| --- | --- | --- |
| Column | Missing Count | Percent Missing |
| location | 15,700 | 2.89% |
| userAgent | 15,700 | 2.89% |
| lastName | 15,700 | 2.89% |
| firstName | 15,700 | 2.89% |
| registration | 15,700 | 2.89% |
| gender | 15,700 | 2.89% |
| artist | 110,828 | 20.38% |
| song | 110,828 | 20.38% |
| length | 110,828 | 20.38% |

**Key Insights:**

* Paid users dominate the platform (~79%).
* Most common events: NextSong (~433k events).
* Male users slightly outnumber female users (M: 302,612, F: 225,393).
* Top locations: New York, Los Angeles, Boston, Chicago, San Francisco.

**User Activity Stats:**

* Top users have 5k–10k+ interactions.
* Users frequently play multiple songs per session (itemInSession mean ~107).
* Session lengths vary widely, max ~1,005 events per session.

**Popular Pages:**

|  |  |
| --- | --- |
| Page | Count |
| NextSong | 432,877 |
| Home | 27,412 |
| Thumbs Up | 23,826 |
| Add to Playlist | 12,349 |
| Add Friend | 8,087 |

**Gender Distribution:** Males: 302,612, Females: 225,393  
**Top Locations:** New York, Los Angeles, Boston, Chicago, San Francisco  
**Top Artists Played:** Kings of Leon, Coldplay, Florence + The Machine, Muse  
**Top Songs Played:** “You’re the One,” “Undo,” “Revelry”

**Visualizations Suggested:**

* Bar chart of top pages & top songs.
* Histogram of session lengths.
* Gender and level distributions as pie charts.
* Geographic heatmap of user activity by location.

**Target Variable:**

* churn\_flag – initially imbalanced, resampled to **277 churners vs. 277 non-churners** for modeling.

**Modeling (النمذجة) & Feature Engineering (هندسة الميزات)**

**Models Used:**

1. Random Forest Classifier
2. XGBoost Classifier

**Evaluation Metrics:**

* Accuracy, Precision, Recall, F1-score, ROC-AUC

**Random Forest Performance:**

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.83 |
| Precision | 0.82–0.84 |
| Recall | 0.82–0.83 |
| F1-score | 0.83 |
| ROC-AUC | 0.877 |

**XGBoost Performance:**

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.82 |
| Precision | 0.81–0.83 |
| Recall | 0.80–0.84 |
| F1-score | 0.82 |
| ROC-AUC | 0.865 |

**Interpretation:**

* Both models achieve **good accuracy (~82–83%)**.
* Balanced precision and recall indicate that churners and non-churners are correctly identified.
* **ROC-AUC ~0.87** indicates strong separation between classes.
* Random Forest slightly outperforms XGBoost in this dataset.
* Overall Great accuracy (~83%) and balanced precision/recall (دقة متوازنة).

**Visualizations:**

|  |  |
| --- | --- |
|  |  |
|  |  |

**A graph of a curve

AI-generated content may be incorrect.**

**Interpretation:**

* ROC-AUC 0.877 → strong separation between churners and non-churners.

**System Design (تصميم النظام)**

1. **Retraining:** Scheduled retraining to adapt to new user behavior.
2. **Environment & Development:** Model developed using **Python** through **Anaconda** in **Jupyter Notebook**.
3. **Hardware:** Work was performed on a laptop with **16GB RAM** and **12th Gen Intel(R) Core (TM) i7-12700H, 2300 MHz, 14 Cores, 20 Logical Processors (My own Dell Laptop)**.
4. **Experiment Tracking:** **MLflow** used to track models and experiments.
5. **Monitoring:** Data drift and concept drift detection (مراقبة الانحراف).

**Challenges & Considerations (التحديات)**

* Handling missing song data (~20%).
* Class imbalance required resampling.
* Avoiding potential data leakage from timestamp/session features.
* Maintaining model performance over time (concept drift).

**Suggestions for Improvement (اقتراحات التحسين)**

* Use advanced feature engineering (e.g., session patterns, song preferences).
* Experiment with other ML models like LightGBM, CatBoost, Neural Networks, and other advanced algorithms.
* Using more powerful hardware to be able to test different computational extensive techniques and algorithms.
* Implement automated retraining pipeline with monitoring dashboards.
* Consider incorporating time-based features for better churn prediction.
* Include time-based features: user inactivity periods, peak listening hours.
* Add personalized churn predictions per user segment (paid vs free).

**How to Run**

**1. Using Jupyter Notebook**

1. Install dependencies:

pip install -r requirements.txt

1. Open Thmanyah - Maged Eid.ipynb in Jupyter Notebook and run all cells.

**2. Using FastAPI (Depends on your systems settings, better to use it directly from Jupyter Notebook)**

1. Run the API:

uvicorn notebook:app --host 0.0.0.0 --port 8000

1. Access API at http://localhost:8000
   * GET / → Welcome message
   * POST /predict → Send JSON with user features to get churn prediction

**3. Using Docker (Depends on your systems settings, better to use it directly from Jupyter Notebook)**

1. Build Docker image:

docker build -t churn\_api .

1. Run container:

docker run -p 8000:8000 churn\_api

1. Access API at http://localhost:8000

**Notes**

* **Scheduled Retraining:** Implemented to updates the model weekly with new data.
* **Data Drift Monitoring:** Compares feature distributions to detect significant changes.
* **MLflow Tracking:** Logs all runs, metrics, and models for reproducibility.
* **Docker:** Ensures a consistent environment for deployment.

**Summary**

**This project demonstrates a complete end-to-end MVP machine learning workflow, from data cleaning and analysis to model deployment and monitoring.** It is ready for production deployment via API and Docker, while tracking experiments and handling real-world challenges like class imbalance, concept drift, and retraining.

**References**

* MLflow Documentation: https://mlflow.org
* FastAPI Documentation: https://fastapi.tiangolo.com
* XGBoost Documentation: https://xgboost.readthedocs.io

This README provides a **clear overview**, **explains results**, and **documents challenges & future improvements**. Thank you for the opportunity to be considered. I look forward to your response and the possibility of contributing to your team.

**Kind regards,  
Maged Eid**