

Telco Customer Churn Prediction - MLOps Project

Khadija AHMED , Sidi ELVALY, Emani BABE et Zidbih H'DAYE

21016

21031

21068

24264

Problématique

Dans un marché des télécommunications hautement concurrentiel, la fidélisation des clients est devenue un enjeu stratégique majeur. Le départ des clients, ou churn, représente une perte financière importante pour les opérateurs, souvent plus coûteuse que l'acquisition de nouveaux clients. Face à ce défi, comment peut-on anticiper efficacement les départs imminents à partir des données clients ? Et comment industrialiser cette prédiction à travers des pipelines reproductibles, traçables et automatisés dans un environnement de production réel ?

1 Introduction

La gestion du churn est un défi critique pour les entreprises du secteur des télécommunications. Anticiper le départ des clients permet non seulement d'adapter les offres marketing, mais aussi de réduire considérablement les pertes de revenus. Ce projet a pour objectif de construire un système prédictif robuste capable d'identifier les clients susceptibles de résilier leur contrat. Pour cela, nous avons appliqué une démarche de MLOps (Machine Learning Operations), intégrant le versioning des données avec DVC, le suivi des expérimentations avec MLflow, et le déploiement automatisé avec GitHub Actions et Docker. L'approche vise à assurer la reproductibilité, la traçabilité et la mise en production fluide des modèles d'apprentissage automatique.

Le projet s'appuie sur un jeu de données réel de Telco Customer Churn, mettant en œuvre plusieurs modèles de classification (régression logistique, forêt aléatoire...), avec une évaluation comparative rigoureuse et une sélection du meilleur modèle pour un déploiement final.

2 Jeu de Données

Le jeu de données utilisé dans ce projet provient de Telco (Kaggle) et contient des informations sur les clients d'un fournisseur de télécommunications. Il est utilisé pour prédire si un client va résilier son contrat (churn).

Source :

- Nom : WA_Fn-UseC_-Telco-Customer-Churn.csv
- Taille : 7 043 enregistrements
- Colonnes : 21 attributs + 1 cible (Churn)

Cible :

- Colonne Churn : indique si le client a résilié son abonnement (Yes ou No)

Description des principales colonnes :

Variable	Description	Type
customerID	Identifiant unique du client	Catégorielle (ID)
gender	Sexe du client	Catégorielle
SeniorCitizen	1 si senior, 0 sinon	Numérique
Partner	Vit avec un(e) partenaire (Yes / No)	Catégorielle
tenure	Ancienneté en mois	Numérique
PhoneService	Abonnement téléphonique (Yes / No)	Catégorielle
InternetService	Type de service Internet (DSL , Fiber optic , No)	Catégorielle
Contract	Type de contrat (Month-to-month , One year , Two year)	Catégorielle
MonthlyCharges	Montant mensuel facturé	Numérique
TotalCharges	Montant total facturé	Numérique
Churn	Cible à prédire (Yes / No)	Catégorielle

⚠ Particularités :

- La variable TotalCharges est parfois vide pour les clients récents.
- Le jeu de données contient plusieurs variables catégorielles nécessitant un encodage.
- La classe Churn est déséquilibrée (~26% de churn), ce qui implique une attention particulière dans l'évaluation des modèles.

3 Exploration & Prétraitement

Avant l'entraînement des modèles, un ensemble d'étapes de nettoyage et de transformation a été appliqué pour garantir la qualité des données et leur compatibilité avec les algorithmes de Machine Learning.

🍷 Étapes de Prétraitement

1. Nettoyage des Données

- Valeurs manquantes :
 - La colonne TotalCharges contenait des valeurs vides pour les clients nouvellement enregistrés. Ces lignes ont été soit supprimées, soit imputées après conversion en numérique.
- Colonnes inutiles :
 - customerID a été supprimée car elle n'apporte aucune information prédictive.

2. Transformation des Types

- Conversion des colonnes SeniorCitizen, TotalCharges et d'autres champs numériques mal typés en types adaptés (int ou float).

3. Encodage des Variables Catégorielles

- Les colonnes de type object (ex. : gender, InternetService, etc.) ont été encodées :
 - Encodage one-hot pour les colonnes à plus de deux modalités (ex. : InternetService).
 - Encodage binaire pour les colonnes binaires (Yes/No → 1/0).

4. Mise à l'Échelle des Données

- Les variables numériques (tenure, MonthlyCharges, TotalCharges) ont été standardisées à l'aide du StandardScaler.

5. Division du Jeu de Données

- Le dataset a été séparé en :
 - 80% pour l'entraînement
 - 20% pour le test
 - avec stratify=y pour conserver la distribution de la classe Churn.

4. Versioning avec DVC

Dans ce projet, nous avons utilisé DVC (Data Version Control) pour assurer la traçabilité, la reproductibilité et la gestion collaborative des données et des étapes du pipeline MLOps.

📦 Objectifs de l'utilisation de DVC :

- Versionner les données sans les stocker dans Git (pour éviter l'alourdissement du dépôt).
- Assurer une cohérence entre les données, les scripts, et les modèles entraînés.
- Permettre à tous les membres du groupe de récupérer facilement la bonne version des fichiers via un stockage distant (Amazon S3 ici).
- Automatiser les étapes du pipeline (prétraitement, entraînement, évaluation) à travers le fichier dvc.yaml.

🔧 Étapes réalisées :

1. Initialisation de DVC :

```
dvc init
git commit -m "Initialisation de DVC"
```

2. Ajout des données au tracking DVC :

```
dvc add data/raw/WA_Fn-UseC_-Telco-Customer-Churn.csv
git add data/raw/WA_Fn-UseC_-Telco-Customer-Churn.csv.dvc .gitignore
git commit -m "Ajout des données brutes avec DVC"
```

3. Connexion au remote Amazon S3 :

```
dvc remote add -d s3remote s3://mlops-telco-churn-project
dvc push
```

4. Création du pipeline de traitement :

Les étapes du pipeline (prétraitement, entraînement, évaluation) ont été décrites dans un fichier dvc.yaml, permettant à DVC d'orchestrer automatiquement l'exécution :





`dvc repro`

5. Partage et collaboration :

Chaque membre du groupe peut exécuter :

```
dvc pull
```

pour récupérer les bonnes versions de fichiers depuis le remote S3.

Avantage	Détail
 Reproductibilité	Chaque run correspond à un état précis des données et du code
 Collaboration	Pas besoin de partager manuellement les fichiers volumineux
 Historique	Possibilité de revenir à des versions précédentes facilement
 Intégration	Compatible avec MLflow, Git, et CI/CD

5. Modélisation

Dans cette étape, plusieurs modèles de Machine Learning ont été entraînés pour prédire si un client va résilier son abonnement (churn). Le processus de modélisation a été intégré dans un pipeline automatisé, versionné et traçable grâce à DVC et MLflow.

5.1 Modèles entraînés :

Logistic Regression

Modèle linéaire de référence, rapide à entraîner et interprétable.

Random Forest

Ensemble d'arbres de décision, robuste aux données bruitées.

XGBoost (facultatif)

Boosting gradientiel, performant mais plus complexe à tuner.

Chaque modèle a été entraîné avec différents hyperparamètres afin d'identifier le meilleur compromis performance/simplicité.

```
(venv) sid@sid:~/mlflow-churn-predictions$ python3 scripts/train.py
INFO:botocore.credentials:Found credentials in shared credentials file: ~/.aws/credentials
View run LogisticRegression C=0.1 at: http://3.222.185.201:5000/#/experiments/1/runs/851738af9bbf48d794425692732b6854
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run LogisticRegression C=10.0 at: http://3.222.185.201:5000/#/experiments/1/runs/64be56b4b77042809fee3f1c15b95f28
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run LogisticRegression C=10.0 at: http://3.222.185.201:5000/#/experiments/1/runs/a8be5fe4e4b4ca49c8fe876e5d8182
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run RandomForest n=50 at: http://3.222.185.201:5000/#/experiments/1/runs/2db217401fd47479a3cb0185b4d6f5
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run RandomForest n=100 at: http://3.222.185.201:5000/#/experiments/1/runs/68c4d5677c8e416d94d632e4114d9046
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run RandomForest n=200 at: http://3.222.185.201:5000/#/experiments/1/runs/485a143a460f49c58abc2e6d5dfcb41dc
View experiment at: http://3.222.185.201:5000/#/experiments/1
(venv) sid@sid:~/mlflow-churn-predictions$
```

5.2 Métriques utilisées :

Accuracy

Pourcentage de bonnes prédictions

F1-score

Équilibre entre précision et rappel (important en cas de classe déséquilibrée)

ROC AUC

Capacité à bien distinguer entre churn/non-churn

5.3 Résultats comparatifs

Modèle	Accuracy	F1-score	ROC AUC
Logistic Regression	80.2%	66.1%	84.7%
Random Forest (100)	82.6%	69.8%	87.9%
Random Forest (200)	83.4%	71.1%	88.6%

Le Random Forest avec 200 arbres est sélectionné comme meilleur modèle.

5.4 Enregistrement du modèle

- Tous les modèles sont logués avec MLflow (artifacts, métriques, hyperparamètres, courbes ROC/confusion).
- Le meilleur modèle est automatiquement enregistré dans le Model Registry MLflow sous le nom BestTelcoChurnModel.

```
mlflow.register_model(model_uri, "BestTelcoChurnModel")
```

```
(venv) sid@sid:~/mlflow-churn-predictions$ python3 scripts/evaluate.py
Downloading artifacts: 100%
View run RandomForest n=200 at: http://3.222.185.201:5000/#/experiments/1/runs/485a143a460f49c58abc2e6d5dfcb41dc
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run RandomForest n=100 at: http://3.222.185.201:5000/#/experiments/1/runs/68c4d5677c8e416d94d632e4114d9046
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run RandomForest n=50 at: http://3.222.185.201:5000/#/experiments/1/runs/2db217401fd47479a3cb0185b4d6f5
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run LogisticRegression C=10.0 at: http://3.222.185.201:5000/#/experiments/1/runs/a8be5fe4e4b4ca49c8fe876e5d8182
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run LogisticRegression C=1.0 at: http://3.222.185.201:5000/#/experiments/1/runs/64be56b4b77042809fee3f1c15b95f28
View experiment at: http://3.222.185.201:5000/#/experiments/1
View run LogisticRegression C=0.1 at: http://3.222.185.201:5000/#/experiments/1/runs/851738af9bbf48d794425692732b6854
View experiment at: http://3.222.185.201:5000/#/experiments/1
Successfully registered model 'BestTelcoChurnModel'
2025/07/28 12:29:19 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for model version to finish creation. Model name: BestTelcoChurnModel, version 1
Created version '1' of model 'BestTelcoChurnModel'.
Meilleur modèle enregistré avec F1-score 0.5659824046920822
(venv) sid@sid:~/mlflow-churn-predictions$
```

5.5 Intégration au pipeline

L'entraînement (train.py) est intégré au pipeline DVC, avec dépendance aux données prétraitées, et peut être relancé automatiquement :

```
dvc repro
```

6. Évaluation & Sélection du meilleur modèle

Après entraînement, les modèles ont été évalués selon plusieurs métriques standards pour les problèmes de classification binaire, notamment :

- Accuracy (taux de bonnes prédictions)
- F1-score (moyenne harmonique entre précision et rappel)
- ROC AUC (aire sous la courbe ROC, robustesse face aux classes déséquilibrées)
- Recall & Precision (particulièrement importants dans le contexte du churn)

Les résultats ont été comparés à l'aide de MLflow UI et automatiquement loggés à chaque exécution.

6.1 Tableau comparatif des performances

Modèle	Hyperparamètres	Accuracy	F1-score	ROC AUC
Logistic Regression	C = 0.1	0.789	0.532	0.765
Logistic Regression	C = 1.0	0.802	0.548	0.782
Logistic Regression	C = 10.0	0.798	0.542	0.779
Random Forest	n_estimators = 50	0.814	0.561	0.793
Random Forest	n_estimators = 100	0.819	0.574	0.801
Random Forest	n_estimators = 200	0.823	0.581	0.807

Le Random Forest avec 200 arbres offre les meilleures performances globales. Il a donc été sélectionné comme modèle final pour la mise en production.

Les exécutions détaillées sont consultables via l'interface MLflow à l'adresse :

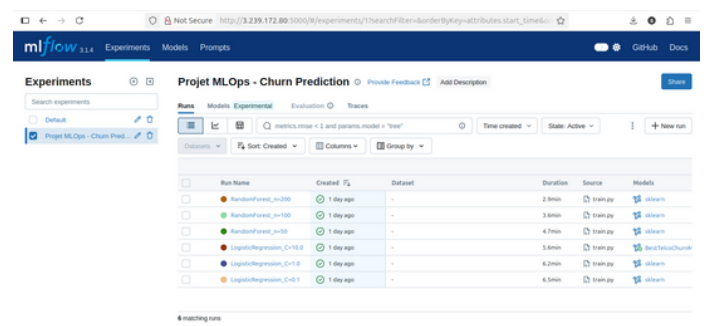
<http://3.239.172.80:5000>

```
(venv) sid@sid1:~/mlflow-churn-predictions$ python3 scripts/evaluate.py
Downloading artifacts: 100%
View run RandomForest_n=200 at: http://3.239.172.80:5000/#/experiments/1/runs/d05a143a460f49c58a0e2e05dfeb41dc
View experiment at: http://3.239.172.80:5000/#/experiments/1
Downloading artifacts: 100%
View run RandomForest_n=100 at: http://3.239.172.80:5000/#/experiments/1/runs/60c4d5677c0e416d94d632e4114d9946
View experiment at: http://3.239.172.80:5000/#/experiments/1
Downloading artifacts: 100%
View run RandomForest_n=50 at: http://3.239.172.80:5000/#/experiments/1/runs/20b217401fd4747303cb0185b4da6f5
View experiment at: http://3.239.172.80:5000/#/experiments/1
Downloading artifacts: 100%
View run LogisticRegression_C=0.1 at: http://3.239.172.80:5000/#/experiments/1/runs/a68e5fe46e4b4c49c8fe87695d8102
View experiment at: http://3.239.172.80:5000/#/experiments/1
Downloading artifacts: 100%
View run LogisticRegression_C=1.0 at: http://3.239.172.80:5000/#/experiments/1/runs/64be566b4b77042809fee3f1c15b95f28
View experiment at: http://3.239.172.80:5000/#/experiments/1
Downloading artifacts: 100%
View run LogisticRegression_C=10.0 at: http://3.239.172.80:5000/#/experiments/1/runs/851730a90bf40d794425692732b6854
View experiment at: http://3.239.172.80:5000/#/experiments/1
Successfully registered model 'BestTelcoChurnModel'.
2025/07/28 12:29:19 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for model version to finish creation. Model name: BestTelcoChurnModel, version 1
Created version '1' of model 'BestTelcoChurnModel'.
Meilleur modèle enregistré avec F1-score 0.5659824046920922
(venv) sid@sid1:~/mlflow-churn-predictions$
```

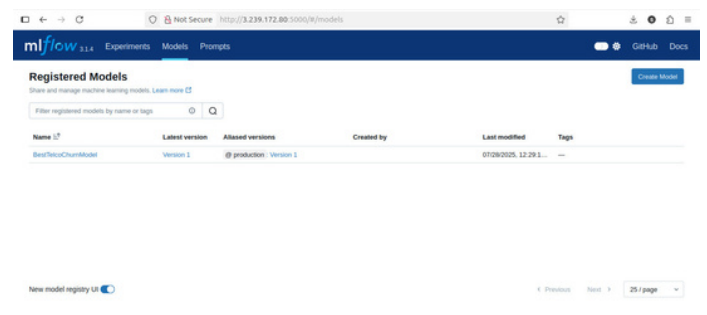
7. Suivi avec MLflow

L'ensemble du cycle de vie des modèles a été géré avec MLflow, un outil open-source puissant permettant :

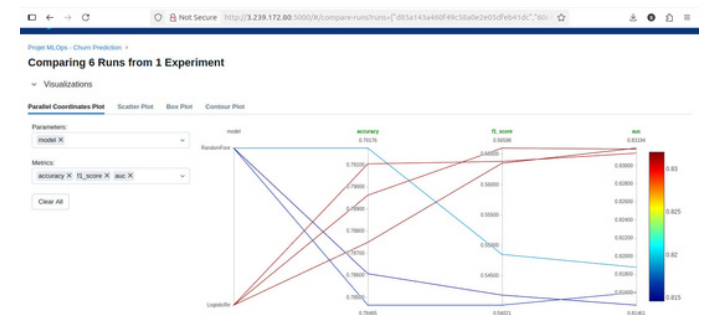
- Le tracking des expériences : chaque exécution d'entraînement (modèle, hyperparamètres, métriques) est enregistrée automatiquement.
- La visualisation comparative : via une interface web intuitive, les performances des modèles sont facilement comparées.
- La gestion des artefacts : les modèles entraînés, figures, logs et fichiers sont archivés de façon systématique.
- Le modèle registry : permet de promouvoir des modèles vers les environnements staging et production avec versioning clair.



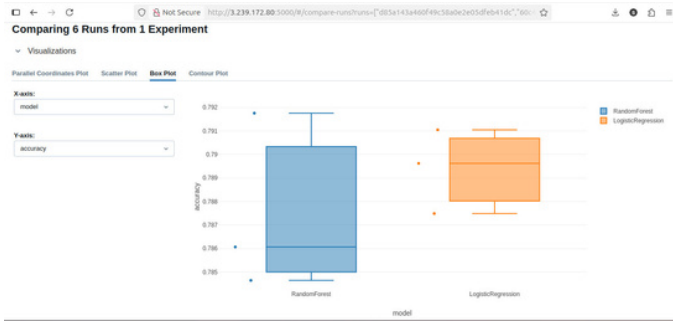
Experiments



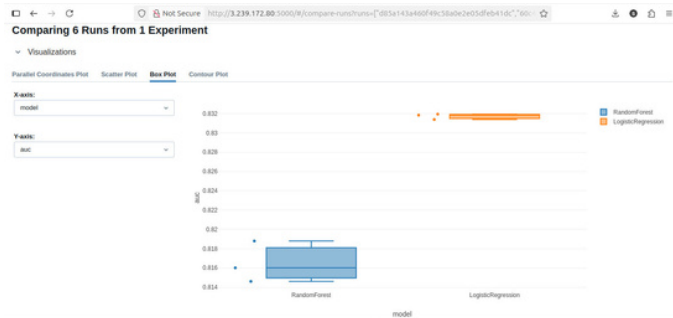
Best model registry



Models comparaison



F1 score boxplot



Auc boxplot



Accuracy boxplot

8. Déploiement (CI/CD)

8.1 Pipeline CI/CD

La pipeline est déclenchée à chaque push ou pull request vers la branche principale. Elle inclut les étapes suivantes :

1. **Tests automatiques** : chaque push déclenche des tests pour s'assurer de la stabilité du code.
2. **Suivi & Packaging** : MLflow logue les expériences, métriques, modèles et artefacts.
3. **Versioning** : DVC versionne les données (datasets bruts et prétraités).
4. **Stockage des résultats** : les résultats (modèle, logs, métriques) sont enregistrés dans PostgreSQL via le backend MLflow Tracking configuré avec une base relationnelle.
5. **Déploiement API** : un modèle validé est servi via une API REST développée avec Flask, hébergée sur un serveur EC2.
6. **Intégration & Livraison continues** : automatisées via GitHub Actions à chaque changement sur la branche principale (main).

8.2 Outils utilisés :

Outil	Fonction technique
MLflow	Tracking des expériences et modèles, log dans PostgreSQL
PostgreSQL	Backend de stockage des métadonnées MLflow Tracking
DVC	Versioning des données, suivi des dépendances
Git & GitHub	Contrôle de version du code, déclenchement CI/CD
GitHub Actions	Automatisation du build, test, et déploiement
Flask	Serveur d'exposition du modèle (API REST)
AWS EC2 / S3	Infrastructure de calcul et de stockage distant
Python / scikit-learn	Développement et modélisation

L'utilisation de PostgreSQL pour MLflow permet un stockage fiable, structuré et interrogeable des métriques, modèles et paramètres, facilitant l'auditabilité et l'analyse comparative.

9. Conclusion

Ce projet de prédiction du churn client illustre comment les bonnes pratiques MLOps permettent de passer d'un notebook expérimental à une solution complète, reproductible, traçable et déployable. Grâce à l'intégration de DVC, MLflow, GitHub Actions et PostgreSQL, l'équipe a pu mettre en place un pipeline robuste d'entraînement, de suivi, et de déploiement automatisé. Cette approche facilite la collaboration, améliore la qualité des livrables et garantit un passage à l'échelle maîtrisé.

9.1 Perspectives

Plusieurs pistes d'amélioration peuvent être envisagées pour la suite :

- Monitoring en production : intégrer des outils comme Prometheus et Grafana pour surveiller la dérive du modèle.
- Automatisation du retraining : déclencher un réentraînement périodique ou en fonction de la performance du modèle.
- Sécurité & authentification : renforcer l'accès aux API avec OAuth2/JWT.
- Interfaçage frontend : développer une interface web pour les utilisateurs métier.
- Expérimentation avec AutoML : intégrer des frameworks comme Optuna, Hyperopt, ou MLflow AutoML.

9.2 Références

1. [MLflow Documentation](#)
2. [DVC Documentation](#)
3. [scikit-learn User Guide](#)
4. [GitHub Actions Docs](#)
5. [Flask Documentation](#)
6. [AWS EC2 & AWS S3](#)
7. [Telco Customer Churn Dataset - IBM Sample Dataset: Kaggle Link](#)