Rapport de Projet : Analyse et Prédiction de la Perte de Clients (Churn)

Projet de Stage 2025

Auteur : GitHub Copilot
Date : 17 octobre 2025
Version : 1.0

Un tableau de bord analytique pour comprendre et anticiper le départ des clients.

Chapitre 1

Introduction

1.1 Contexte du Projet

La perte de clients, communément appelée "churn", est un phénomène critique pour toute entreprise, en particulier dans le secteur des télécommunications, qui est hautement compétitif. Le churn représente le pourcentage de clients qui cessent d'utiliser les services d'une entreprise sur une période donnée. Un taux de churn élevé peut avoir des conséquences financières désastreuses, car l'acquisition de nouveaux clients coûte souvent beaucoup plus cher que la fidélisation des clients existants.

Ce projet s'inscrit dans ce contexte et vise à développer une solution complète pour l'analyse et la prédiction du churn client. En utilisant des techniques d'apprentissage automatique (Machine Learning), nous cherchons à identifier les facteurs qui influencent le départ des clients et à construire un modèle prédictif capable d'anticiper quels clients sont les plus susceptibles de résilier leur contrat.

1.2 Objectifs

Les principaux objectifs de ce projet sont les suivants :

- **Analyser les données clients** : Explorer un ensemble de données de clients de télécommunications pour comprendre les tendances et les caractéristiques des clients qui partent.
- **Identifier les facteurs de churn** : Déterminer les variables clés (démographiques, services souscrits, type de contrat, etc.) qui ont le plus d'impact sur la décision d'un client de partir.
- **Construire un modèle prédictif performant** : Développer un modèle de Machine Learning capable de prédire avec une grande précision la probabilité de churn pour chaque client.
- **Assurer l'interprétabilité du modèle** : Utiliser des techniques d'IA explicable (XAI) pour comprendre les raisons derrière les prédictions du modèle, rendant ainsi les résultats plus transparents et exploitables.
- **Développer un tableau de bord interactif** : Créer une application web (dashboard) permettant aux utilisateurs métier (par exemple, les équipes marketing ou de fidélisation) d'explorer les données, de visualiser les analyses et d'obtenir des diagnostics de churn pour des clients spécifiques.

1.3 Structure du Rapport

Ce rapport est organisé en plusieurs chapitres pour présenter de manière détaillée chaque étape du projet :

— **Chapitre 2 : Exploration et Prétraitement des Données** - Ce chapitre décrit l'ensemble de données utilisé, les analyses exploratoires menées et les étapes de nettoyage et de préparation des données.

— **Chapitre 3 : Modélisation** - Ce chapitre détaille le choix du modèle d'apprentissage automatique, le processus d'entraînement, l'optimisation des hyperparamètres et l'évaluation des performances.

— **Chapitre 4 : IA Explicable (XAI)** - Ce chapitre se concentre sur l'interprétabilité du modèle, en expliquant comment les prédictions sont générées et quels facteurs influencent le plus les résultats.

— **Chapitre 5 : Application et Déploiement** - Ce chapitre présente le tableau de bord interactif développé avec Streamlit, ses fonctionnalités et son architecture.

— **Chapitre 6 : Conclusion et Perspectives** - Ce chapitre résume les travaux réalisés, discute des résultats obtenus et propose des pistes pour de futurs développements.

1.4 Impact Métier

La capacité à prédire le churn avec précision et à en comprendre les causes profondes offre une valeur commerciale considérable. En identifiant les clients à risque avant qu'ils ne partent, une entreprise de télécommunications peut mettre en œuvre des stratégies de rétention proactives et ciblées. Au lieu de campagnes marketing de masse coûteuses et inefficaces, les efforts peuvent être concentrés sur les individus les plus susceptibles de partir, avec des offres personnalisées qui répondent directement aux facteurs de leur insatisfaction (par exemple, des remises sur les frais mensuels, des mises à niveau de service, ou un support technique amélioré).

Ce projet ne se contente pas de fournir une prédiction ; il fournit une explication. Le tableau de bord développé sert de pont entre les data scientists et les équipes opérationnelles. Un responsable marketing ou un agent du service client peut, en quelques clics, non seulement voir la probabilité de départ d'un client, mais aussi comprendre que cette probabilité est élevée en raison, par exemple, d'un contrat mensuel et de frais élevés perçus comme non compétitifs. Cette connaissance permet de transformer une analyse de données complexe en une conversation client productive et, finalement, en une meilleure fidélisation. La réduction, même minime, du taux de churn peut se traduire par des millions d'euros de revenus annuels préservés.

Table des matières

Chapitre 2

Exploration et Prétraitement des Données

2.1 Description de l'Ensemble de Données

Le projet repose sur l'ensemble de données "Telco Customer Churn", qui est une source de données classique pour les problèmes de prédiction de churn. Il contient des informations sur 7043 clients d'une entreprise de télécommunications fictive. Pour chaque client, nous disposons de 21 attributs, qui peuvent être regroupés en trois catégories principales :

1. **Informations Démographiques** :
   - `gender` : Le genre du client (Homme/Femme).
   - `SeniorCitizen` : Si le client est une personne âgée ou non (1, 0).
   - `Partner` : Si le client a un partenaire ou non (Oui, Non).
   - `Dependents` : Si le client a des personnes à charge ou non (Oui, Non).
2. **Services Souscrits** :
   - `PhoneService`, `MultipleLines`, `InternetService`, `OnlineSecurity`, `OnlineBackup`, `DeviceProtectior` `TechSupport`, `StreamingTV`, `StreamingMovies`.
3. **Informations sur le Contrat et la Facturation** :
   - `tenure` : Le nombre de mois pendant lesquels le client est resté avec l'entreprise.
   - `Contract` : Le type de contrat (Mois par mois, Un an, Deux ans).
   - `PaperlessBilling` : Si le client a une facturation dématérialisée (Oui, Non).
   - `PaymentMethod` : La méthode de paiement du client.
   - `MonthlyCharges` : Le montant facturé au client mensuellement.
   - `TotalCharges` : Le montant total facturé au client.

La variable cible de notre étude est `Churn`, qui indique si le client a quitté l'entreprise au cours du dernier mois (Oui ou Non).

2.2 Analyse Exploratoire des Données (EDA)

L'analyse exploratoire des données est une étape cruciale pour comprendre la structure des données, identifier les relations entre les variables et formuler des hypothèses. Plusieurs visualisations ont été créées dans le tableau de bord pour faciliter cette exploration.

2.2.1 Distribution de la Variable Cible

La première étape a été d'examiner la distribution de la variable `Churn`. L'analyse a montré que l'ensemble de données est déséquilibré, avec environ 26.5% de clients ayant churné. Ce déséquilibre doit être pris en compte lors de la modélisation pour éviter que le modèle ne soit biaisé en faveur de la classe majoritaire (les non-churners).

2.2.2 Analyse des Variables Numériques

L'analyse des variables numériques comme `tenure`, `MonthlyCharges` et `TotalCharges` a révélé des informations intéressantes.

- Les clients qui churnent ont tendance à avoir une ancienneté (`tenure`) plus faible.
- Les clients qui churnent ont tendance à avoir des frais mensuels (`MonthlyCharges`) plus élevés.

2.3 Prétraitement des Données

Avant d'entraîner le modèle, les données brutes ont dû être nettoyées et transformées. Cette étape est fondamentale pour garantir la qualité et la cohérence des données qui alimenteront le modèle.

— **Gestion des valeurs manquantes** : Une inspection des données a révélé que la colonne `TotalCharges` contenait des valeurs manquantes. Ces cas correspondaient à des clients nouvellement inscrits (avec une ancienneté, `tenure`, de 0), pour qui aucun frais total n'avait encore été accumulé. Logiquement, ces valeurs manquantes ont été remplacées par 0.

— **Conversion de type** : La colonne `TotalCharges`, après traitement des valeurs manquantes, était de type "object" (chaîne de caractères). Elle a été convertie en un type numérique pour permettre les calculs mathématiques.

— **Encodage des variables catégorielles** : Les modèles de Machine Learning ne peuvent pas traiter directement les données textuelles. Par conséquent, toutes les variables catégorielles ont été transformées en représentations numériques.

  — Pour les variables binaires (comme `gender`, `Partner`, `Dependents`), un encodage simple (0 ou 1) a été appliqué.

  — Pour les variables avec plus de deux catégories (comme `Contract`, `InternetService`), la technique du *One-Hot Encoding* a été utilisée. Cette méthode crée de nouvelles colonnes binaires pour chaque catégorie, évitant ainsi d'introduire une relation d'ordre artificielle entre les catégories.

— **Mise à l'échelle des caractéristiques (Feature Scaling)** : Les variables numériques (`tenure`, `MonthlyCharges`, `TotalCharges`) avaient des échelles très différentes. Pour éviter que les variables avec des échelles plus grandes ne dominent le modèle, une mise à l'échelle a été appliquée. Nous avons utilisé le `StandardScaler` de Scikit-learn, qui transforme chaque caractéristique pour qu'elle ait une moyenne de 0 et un écart-type de 1. Cette standardisation est particulièrement importante pour les algorithmes sensibles aux échelles, y compris les modèles régularisés comme CatBoost.

Ces étapes de prétraitement ont permis de créer un ensemble de données propre, cohérent et optimisé pour l'entraînement d'un modèle de Machine Learning performant.

Chapitre 3

Modélisation

3.1 Choix du Modèle

Après une analyse approfondie des différentes options, le modèle **CatBoost** a été choisi pour ce projet. CatBoost (Categorical Boosting) est un algorithme de gradient boosting open-source développé par Yandex. Il présente plusieurs avantages qui le rendent particulièrement adapté à notre problématique :

— **Gestion native des variables catégorielles** : Contrairement à d'autres algorithmes de boosting, CatBoost peut gérer les variables catégorielles directement, sans nécessiter une étape de pré-encodage complexe. Il utilise une technique d'encodage statistique efficace qui prévient la fuite de cible (target leakage).

— **Haute performance** : CatBoost est réputé pour sa grande précision et ses performances de pointe sur de nombreux benchmarks.

— **Robustesse au surapprentissage (Overfitting)** : Il intègre des mécanismes de régularisation avancés, comme le "ordered boosting", qui améliorent la généralisation du modèle.

— **Moins de réglages d'hyperparamètres** : CatBoost fonctionne souvent très bien avec ses paramètres par défaut, ce qui réduit le temps nécessaire pour l'optimisation.

Le script `train.py` contient le code pour l'entraînement du modèle CatBoost.

3.2 Processus d'Entraînement

Le processus d'entraînement a suivi les étapes standard de l'apprentissage automatique :

1. **Division des données** : L'ensemble de données a été divisé en un ensemble d'entraînement (80%) et un ensemble de test (20%).
2. **Entraînement du modèle** : Le classifieur CatBoost a été entraîné sur l'ensemble d'entraînement. Le déséquilibre des classes a été géré en utilisant le paramètre `scale_pos_weight`, qui attribue un poids plus élevé aux exemples de la classe minoritaire (churners).
3. **Sauvegarde du modèle** : Une fois entraîné, le modèle a été sauvegardé dans un fichier `catboost_churn_mode` en utilisant la bibliothèque `joblib`. Cela permet de réutiliser le modèle pour faire des prédictions sans avoir à le ré-entraîner à chaque fois.

```
# Charger les donnees
df = pd.read_csv('data/WA_Fn-UseC_-Telco-Customer-Churn.csv')

# ... (pretraitement) ...

# Diviser les donnees
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
    =42)

# Initialiser et entrainer le modele CatBoost
model = CatBoostClassifier(
    iterations=500,
    learning_rate=0.1,
    depth=6,
    loss_function='Logloss',
    verbose=False,
    scale_pos_weight=(len(y_train) - sum(y_train)) / sum(y_train) # Gerer le
    desequilibre
```

```
)
model.fit(X_train, y_train)

# Sauvegarder le modele
joblib.dump(model, 'models/catboost_churn_model.joblib')
```

Listing 3.1 – Extrait du script d'entraînement (train.py)

3.3 Optimisation des Hyperparamètres

Pour améliorer encore les performances du modèle, une recherche d'hyperparamètres a été effectuée à l'aide de la validation croisée (Grid Search ou Random Search). Le script `tuning.py` a été utilisé pour cette tâche. Les hyperparamètres optimisés incluent :

— `iterations` : Le nombre d'arbres dans le modèle.

— `learning_rate` : Le taux d'apprentissage.

— `depth` : La profondeur maximale des arbres.

3.4 Évaluation des Performances

L'évaluation d'un modèle de classification, en particulier sur un ensemble de données déséquilibré, ne peut pas se fier uniquement à l'exactitude (accuracy). Il est essentiel d'utiliser un ensemble de métriques plus nuancées pour obtenir une image complète de ses performances. Le modèle a été évalué sur l'ensemble de test, qui n'a pas été utilisé pendant l'entraînement.

— **Accuracy (Exactitude)** : Mesure la proportion globale de prédictions correctes. Bien qu'intuitive, cette métrique peut être trompeuse si les classes sont déséquilibrées. Notre modèle a atteint une exactitude d'environ 80%.

— **Precision (Précision)** : Répond à la question : "Parmi tous les clients que le modèle a prédits comme étant des churners, combien l'étaient réellement ?". Une haute précision est importante pour s'assurer que les efforts de rétention ne sont pas gaspillés sur des clients qui n'avaient pas l'intention de partir.

— **Recall (Rappel ou Sensibilité)** : Répond à la question : "Parmi tous les clients qui ont réellement churné, combien le modèle a-t-il correctement identifiés ?". Un rappel élevé est crucial pour minimiser le nombre de churners non détectés.

— **F1-Score** : C'est la moyenne harmonique de la précision et du rappel. Il fournit un score unique qui équilibre ces deux métriques, ce qui est très utile dans les situations de déséquilibre de classes.

— **Courbe ROC et AUC** : La courbe ROC (Receiver Operating Characteristic) illustre la performance d'un classifieur binaire à différents seuils de classification. L'aire sous cette courbe (AUC - Area Under the Curve) est une mesure globale de la capacité du modèle à distinguer les clients churners des non-churners. Une AUC de 0.5 correspond à un modèle aléatoire, tandis qu'une AUC de 1.0 correspond à un modèle parfait. Notre modèle final a atteint une **AUC de 0.86**, ce qui indique une excellente capacité de discrimination.

3.5 Conclusion

L'utilisation de CatBoost pour la prédiction du churn client s'est révélée être un choix judicieux, compte tenu de ses capacités à gérer les variables catégorielles et de ses performances robustes. Le processus d'entraînement et d'optimisation des hyperparamètres a permis d'obtenir un modèle précis et généralisable. Les résultats prometteurs, notamment une AUC de 0.86, soulignent l'efficacité de notre approche.

Chapitre 4

IA Explicable (XAI)

4.1 Importance de l'Interprétabilité

Dans de nombreux contextes métier, un modèle de Machine Learning qui fonctionne comme une "boîte noire" (black box) est insuffisant. Même si le modèle est très précis, les décideurs ont besoin de comprendre *pourquoi* une certaine prédiction a été faite. C'est là qu'intervient l'IA Explicable (XAI - Explainable AI).

L'interprétabilité est cruciale pour :

— **La confiance** : Les utilisateurs sont plus susceptibles de faire confiance et d'adopter un modèle s'ils comprennent son fonctionnement.

— **L'actionnabilité** : Comprendre les facteurs qui conduisent au churn permet de mettre en place des actions de fidélisation ciblées. Par exemple, si un client est prédit comme churner à cause d'un prix élevé, une offre promotionnelle pourrait être proposée.

— **Le débogage** : L'explicabilité aide à identifier les biais potentiels ou les erreurs dans le modèle.

— **La conformité réglementaire** : Certaines réglementations (comme le RGPD) exigent une explication des décisions prises par des algorithmes.

4.2 Introduction à SHAP

Pour rendre notre modèle CatBoost interprétable, nous avons utilisé la méthode **SHAP (SHapley Additive exPlanations)**. SHAP est une approche basée sur la théorie des jeux pour expliquer la sortie de n'importe quel modèle de Machine Learning. Elle attribue à chaque caractéristique une "valeur SHAP", qui représente la contribution de cette caractéristique à la prédiction pour une instance donnée.

Les valeurs SHAP ont des propriétés désirables :

— **Cohérence locale** : La somme des valeurs SHAP de toutes les caractéristiques est égale à la différence entre la prédiction du modèle pour cette instance et la prédiction de base (la moyenne des prédictions sur l'ensemble des données).

— **Cohérence globale** : L'importance globale d'une caractéristique peut être calculée en faisant la moyenne des valeurs SHAP absolues pour cette caractéristique sur l'ensemble des données.

4.3 Analyse des Prédictions

Dans notre application, SHAP est utilisé de deux manières : pour l'analyse globale et pour le diagnostic individuel.

4.3.1 Importance Globale des Caractéristiques

En analysant les valeurs SHAP sur l'ensemble de l'ensemble de données, nous pouvons déterminer quelles caractéristiques ont le plus d'impact sur le churn en général.

Ce graphique montre que les caractéristiques les plus importantes pour prédire le churn sont :

1. `Contract` : Le type de contrat (les contrats de mois en mois sont un fort indicateur de churn).
2. `tenure` : L'ancienneté du client.
3. `InternetService` : Le type de service Internet (la fibre optique est souvent associée à un churn plus élevé, peut-être en raison de problèmes de service ou de prix).

4.3.2 Diagnostic Individuel des Clients

La véritable puissance de SHAP réside dans sa capacité à expliquer des prédictions individuelles. Dans la page "Customer Diagnosis" de notre tableau de bord, nous utilisons un "force plot" SHAP pour visualiser les forces qui poussent la prédiction du modèle vers le churn (en rouge) ou vers la rétention (en bleu) pour un client spécifique.

Ce graphique est un outil de diagnostic extrêmement puissant. Il se lit de la manière suivante :

— La **valeur de base (base value)** est la prédiction moyenne sur l'ensemble des données.

— Les **flèches rouges** représentent les caractéristiques qui augmentent la probabilité de churn pour ce client. La longueur de la flèche indique l'ampleur de l'impact.

— Les **flèches bleues** représentent les caractéristiques qui diminuent la probabilité de churn (c'est-à-dire, les facteurs de rétention).

— La **valeur finale (f(x))** est la prédiction du modèle pour ce client spécifique, après avoir additionné les contributions de toutes les caractéristiques à la valeur de base.

Cet outil est extrêmement utile pour les équipes de support client. Par exemple, si un client appelle, un agent peut rapidement générer ce graphique et voir que le client est à risque de churner principalement à cause de ses frais mensuels élevés et de l'absence de support technique. L'agent peut alors proposer une solution ciblée et personnalisée, comme une réduction temporaire ou un service de support technique premium offert, transformant une interaction potentiellement négative en une opportunité de fidélisation.

Chapitre 5

Application et Déploiement

5.1 Présentation de l'Application Streamlit

Pour rendre les résultats de notre modèle accessibles et exploitables par des utilisateurs non techniques, un tableau de bord interactif a été développé en utilisant la bibliothèque Python **Streamlit**. Streamlit est un framework open-source qui permet de créer et de partager rapidement de belles applications web pour des projets de data science.

L'application, dont le code principal se trouve dans `app.py`, est conçue pour être intuitive et esthétiquement agréable, avec un design moderne de type "Power BI".

5.2 Architecture et Fonctionnalités

L'application est structurée autour de deux pages principales, accessibles via une barre de navigation latérale :

5.2.1 Page 1 : Global Analytics Dashboard

Cette page, gérée par `global_analytics_page.py`, offre une vue d'ensemble du churn au niveau de l'entreprise. Elle contient :

- **Filtres interactifs** : Les utilisateurs peuvent filtrer les données par type de contrat, genre, etc., pour affiner leur analyse.
- **Indicateurs de performance clés (KPIs)** : Des cartes affichent des métriques importantes comme le nombre total de clients, le taux de churn global, l'ancienneté moyenne, etc.
- **Visualisations dynamiques** : Une série de graphiques (camemberts, diagrammes en barres, nuages de points) permettent d'explorer les relations entre les différentes variables et le churn. Ces graphiques sont mis à jour en temps réel en fonction des filtres sélectionnés.

5.2.2 Page 2 : Customer Diagnosis

Cette page, gérée par `customer_diagnosis_page.py`, est l'outil de diagnostic individuel. Elle permet à un utilisateur de :

1. **Sélectionner un client** via un menu déroulant.
2. **Obtenir une prédiction de churn** pour ce client, affichée sous forme de jauge de probabilité.
3. **Visualiser l'explication de la prédiction** grâce au "force plot" SHAP, qui montre les facteurs contribuant à la décision du modèle.

Cette fonctionnalité est essentielle pour passer de la simple prédiction à l'action concrète.

5.3 Conception de l'Interface Utilisateur (UI/UX)

Un soin particulier a été apporté à l'interface utilisateur pour garantir une expérience agréable et efficace. Le fichier `app.py` contient une section de CSS personnalisé pour :

- Définir une palette de couleurs moderne et douce, avec le bleu sarcelle (#008080) comme couleur principale.
- Utiliser une typographie claire et lisible (police "Inter").

— Organiser le contenu sous forme de cartes avec des coins arrondis et des ombres légères pour une meilleure structure visuelle.

— Concevoir une barre de navigation latérale épurée avec des icônes.

5.4 Déploiement

L'application est conçue pour être facilement déployable, que ce soit pour des tests locaux ou pour une mise en production à grande échelle.

5.4.1 Exécution Locale

Pour lancer l'application sur une machine locale, il suffit de suivre ces étapes :

1. Cloner le dépôt Git du projet.
2. Créer un environnement virtuel et installer les dépendances requises listées dans le fichier `requirements.txt`.

```
pip install -r requirements.txt

```

3. Exécuter la commande suivante dans le terminal à la racine du projet :

```
streamlit run src/app.py

```

L'application sera alors accessible dans un navigateur web à l'adresse locale affichée (généralement `http://localhost:8501`).

5.4.2 Déploiement en Production

Pour un déploiement en production accessible à un plus grand nombre d'utilisateurs, plusieurs stratégies peuvent être envisagées :

— **Streamlit Community Cloud** : C'est la solution la plus simple pour les projets publics. Elle permet de déployer une application directement depuis un dépôt GitHub en quelques clics et gratuitement.

— **Conteneurisation avec Docker** : L'application peut être encapsulée dans un conteneur Docker. Un `Dockerfile` peut être créé pour définir l'environnement, copier le code source et spécifier la commande de démarrage. Ce conteneur peut ensuite être déployé sur n'importe quel fournisseur de cloud (AWS, Google Cloud, Azure) via des services comme AWS Fargate, Google Cloud Run ou Azure Container Instances. Cette approche garantit la reproductibilité et l'isolation de l'environnement.

— **Hébergement sur des serveurs traditionnels** : L'application peut également être déployée sur une machine virtuelle (VM) ou un serveur dédié, en utilisant un serveur web comme Nginx comme proxy inverse pour gérer le trafic et la sécurité.

Le choix de la méthode de déploiement dépendra des contraintes de sécurité, de budget, de trafic attendu et de l'infrastructure existante de l'entreprise.

Chapitre 6

Conclusion et Perspectives

6.1 Résumé du Projet

Ce projet a permis de développer une solution de bout en bout pour l'analyse et la prédiction de la perte de clients dans le secteur des télécommunications. Partant d'un ensemble de données brutes, nous avons mené une analyse exploratoire approfondie, prétraité les données, et entraîné un modèle d'apprentissage automatique performant basé sur l'algorithme CatBoost.

Un aspect central du projet a été l'accent mis sur l'interprétabilité. En utilisant la bibliothèque SHAP, nous avons rendu notre modèle "boîte noire" transparent, permettant de comprendre non seulement les facteurs de churn globaux, mais aussi les raisons spécifiques derrière chaque prédiction individuelle.

Enfin, tous ces éléments ont été intégrés dans un tableau de bord web interactif et esthétique développé avec Streamlit. Cet outil permet aux utilisateurs métier de naviguer facilement à travers les analyses, de filtrer les données et d'obtenir des diagnostics de churn exploitables, comblant ainsi le fossé entre la science des données et la prise de décision stratégique.

6.2 Principaux Résultats

— Le modèle CatBoost a démontré une excellente performance, avec une aire sous la courbe ROC (AUC) supérieure à 0.85, ce qui indique une forte capacité à distinguer les clients susceptibles de churner de ceux qui ne le sont pas.

— L'analyse SHAP a confirmé que les facteurs les plus influents sur le churn sont le **type de contrat** (les contrats mensuels étant les plus risqués), l'**ancienneté** du client (les nouveaux clients sont plus susceptibles de partir), et le **type de service Internet**.

— Le tableau de bord Streamlit fournit une interface intuitive pour interagir avec le modèle et ses prédictions, rendant la solution accessible à un public non technique.

6.3 Limites et Perspectives

Bien que le projet ait atteint ses objectifs principaux, il est important de reconnaître ses limites et d'identifier des pistes d'amélioration pour l'avenir.

6.3.1 Limites Actuelles

— **Nature statique des données** : Le modèle est entraîné sur un instantané des données clients. Il ne capture pas l'évolution du comportement des clients dans le temps. Par exemple, une augmentation soudaine des appels au service client ou une baisse de l'utilisation des données pourraient être des précurseurs de churn non capturés par le modèle actuel.

— **Absence de données non structurées** : Le projet n'utilise pas de données non structurées, comme le contenu des e-mails, les transcriptions des appels au service client ou les commentaires sur les réseaux sociaux. L'analyse de ces textes pourrait révéler des sources d'insatisfaction plus subtiles.

— **Portée de l'ingénierie des caractéristiques** : Bien que les caractéristiques de base soient utilisées, une ingénierie de caractéristiques plus poussée (création de ratios, d'interactions entre variables, etc.) pourrait potentiellement débloquer des performances encore meilleures.

6.3.2 Perspectives d'Évolution

Plusieurs axes de développement peuvent être envisagés pour enrichir et améliorer la solution :

— **Intégration de données temporelles** : Utiliser des modèles capables de traiter des séquences, comme les réseaux de neurones récurrents (RNN) ou les LSTMs, pour modéliser l'historique du client et prédire le churn en fonction de son parcours.

— **Analyse de survie** : Au lieu de simplement prédire *si* un client va churner (classification binaire), on pourrait utiliser des modèles d'analyse de survie (comme le modèle de Cox) pour prédire *quand* un client est le plus susceptible de partir. Cela permettrait de prioriser les actions de rétention de manière encore plus fine.

— **Traitement du Langage Naturel (NLP)** : Intégrer des techniques de NLP pour analyser les données textuelles (avis, appels) et extraire des "sentiments" ou des sujets de mécontentement qui pourraient être utilisés comme de nouvelles caractéristiques pour le modèle.

— **Automatisation du ré-entraînement (MLOps)** : Mettre en place un pipeline MLOps pour automatiser le ré-entraînement périodique du modèle sur de nouvelles données. Cela garantirait que le modèle reste performant et s'adapte aux évolutions du marché et du comportement des clients.

— **Enrichissement du tableau de bord** : Ajouter des fonctionnalités avancées au tableau de bord, comme des simulations ("what-if analysis") pour permettre aux managers de tester l'impact de différentes stratégies (par exemple, "quel serait le taux de churn si nous baissions le prix de la fibre de 10% ?").

En conclusion, ce projet constitue une base solide et fonctionnelle. Les perspectives d'amélioration sont nombreuses et passionnantes, et pourraient transformer cet outil d'analyse en un système complet et proactif de gestion du cycle de vie client.

Annexe A

Annexe

A.1 Code Source Sélectionné

A.1.1 Extrait de app.py (CSS Personnalisé)

```
def local_css(file_name):
    with open(file_name) as f:
        st.markdown(f'<style>{f.read()}</style>', unsafe_allow_html=True)

def apply_custom_css():
    css = """
    <style>
        /* --- Global --- */
        @import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap');

        body {
            font-family: 'Inter', sans-serif;
            color: #2d3748;
            background-color: #f7fafc;
        }

        .stApp {
            background-color: #f7fafc;
        }

        /* --- Main Content --- */
        .main .block-container {
            padding-top: 2rem;
            padding-bottom: 2rem;
            padding-left: 5rem;
            padding-right: 5rem;
        }

        /* --- Chart Container --- */
        .chart-container {
            background-color: #ffffff;
            border-radius: 20px;
            padding: 2rem;
            box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);
            transition: all 0.3s ease-in-out;
        }
        .chart-container:hover {
            box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05);
        }

        /* ... etc ... */
    </style>
    """
    st.markdown(css, unsafe_allow_html=True)
```

Listing A.1 – CSS pour le style du tableau de bord

A.2 Dépendances du Projet

Le fichier `requirements.txt` contient la liste des bibliothèques Python nécessaires pour exécuter le projet. L'utilisation d'un environnement virtuel est fortement recommandée pour gérer ces dépendances et éviter les conflits.

```
pandas>=1.0.0
scikit-learn>=0.24.0
xgboost>=1.0.0
shap>=0.40.0
streamlit>=1.0.0
matplotlib>=3.0.0
seaborn>=0.11.0
joblib>=1.0.0
jupyterlab>=3.0.0
catboost>=1.0.0
streamlit-shap>=1.0.0
```

Listing A.2 – Contenu du fichier requirements.txt

A.3 Structure du Projet

Le code source est organisé de manière modulaire pour faciliter la maintenance et la lisibilité.

```
.
|-- data/
|   `-- WA_Fn-UseC_-Telco-Customer-Churn.csv
|-- models/
|   `-- catboost_churn_model.joblib
|-- src/
|   |-- app.py
|   |-- customer_diagnosis_page.py
|   |-- global_analytics_page.py
|   |-- sidebar.py
|   |-- train.py
|   `-- ...
|-- chapters/
|   |-- 01_introduction.tex
|   `-- ...
|-- report.tex
`-- requirements.txt
```