EXPLICATION DÉTAILLÉE DU PROJET ## TRAVAUX PRATIQUES 2 : PRÉDICTION DU CHURN CLIENT --- ## 🎯 OBJECTIF DU PROJET Développer un **système complet de Machine Learning** pour prédire le **churn client** (désabonnement) dans une entreprise de télécommunications. ### Qu'est-ce que le Churn ? Le **churn** (ou attrition client) désigne le phénomène par lequel un client arrête d'utiliser les services d'une entreprise. C'est un KPI critique pour toute entreprise, particulièrement dans les secteurs : - Télécommunications - Banques et assurances - Services par abonnement (streaming, SaaS) - E-commerce ### Pourquoi est-ce important en entreprise ? **Coût d'acquisition vs rétention :** - Acquérir un nouveau client coûte **5 à 25 fois plus cher** que de conserver un client existant - Une amélioration de 5% du taux de rétention peut augmenter les profits de **25% à 95%** - Les clients fidèles dépensent en moyenne **67% de plus** que les nouveaux clients **Actions possibles avec la prédiction du churn :** 1. **Rétention proactive** : Identifier les clients à risque avant qu'ils partent 2. **Offres personnalisées** : Proposer des promotions ciblées 3. **Amélioration du service** : Comprendre les facteurs de départ 4. **Optimisation des ressources** : Concentrer les efforts sur les clients à forte valeur --- ## 📊 COMPRÉHENSION DU DATASET ### Variables du dataset Télécommunications Le dataset contient des informations sur environ **7 000 clients** avec les caractéristiques suivantes : #### **1. Informations Démographiques** ``` - gender : Sexe du client (Male/Female) - SeniorCitizen : Senior (1) ou non (0) - Partner : A un conjoint (Yes/No) - Dependents : A des personnes à charge (Yes/No) ``` **Utilité :** Ces variables permettent de segmenter les clients et d'identifier des patterns démographiques. #### **2. Informations sur le Compte** ``` - tenure : Ancienneté en mois (combien de temps le client est resté) - Contract : Type de contrat (Month-to-month, One year, Two year) - PaperlessBilling : Facturation électronique (Yes/No) - PaymentMethod : Méthode de paiement (Electronic check, Mailed check, etc.) - MonthlyCharges : Montant facturé mensuellement - TotalCharges : Montant total facturé depuis le début ``` **Hypothèses business :** - Les clients avec un contrat mensuel ont plus de chances de partir - Les clients avec une longue ancienneté sont plus fidèles - Le montant des charges influence la décision de rester/partir #### **3. Services Souscrits** ``` - PhoneService : Service téléphonique (Yes/No) - MultipleLines : Plusieurs lignes (Yes/No/No phone service) - InternetService : Type d'Internet (DSL, Fiber optic, No) - OnlineSecurity : Sécurité en ligne (Yes/No/No internet service) - OnlineBackup : Sauvegarde en ligne (Yes/No/No internet service) - DeviceProtection : Protection des appareils - TechSupport : Support technique - StreamingTV : Streaming TV - StreamingMovies : Streaming films ``` **Hypothèses business :** - Plus un client a de services, plus il est "engagé" et moins il risque de partir - La qualité des services (notamment Internet) influence le churn #### **4. Variable Cible** ``` - Churn : Le client est-il parti ? (Yes/No) ``` --- ## 🔧 ÉTAPES DU PROJET DÉTAILLÉES ### **ÉTAPE 1 : EXPLORATION DES DONNÉES (EDA)** ```python # Charger et examiner les données df_churn = pd.read_csv(url) print(df_churn.shape) # (7043, 21) - 7043 clients, 21 variables print(df_churn.head()) ``` **Questions à se poser :** - Combien de clients ont quitté l'entreprise ? - Quelle est la distribution des variables ? - Y a-t-il des valeurs manquantes ? - Y a-t-il des valeurs aberrantes ? **Analyse du Churn :** ```python print(df_churn['Churn'].value_counts(normalize=True)) # Résultat typique : # No 0.734 → 73.4% de clients fidèles # Yes 0.266 → 26.6% de clients partis ``` **Insight :** Le dataset est **déséquilibré** (imbalanced). Il faudra en tenir compte lors de la modélisation. --- ### **ÉTAPE 2 : NETTOYAGE DES DONNÉES** #### Problème : TotalCharges contient des espaces ```python # TotalCharges est une chaîne de caractères au lieu de numérique df_churn['TotalCharges'] = pd.to_numeric(df_churn['TotalCharges'], errors='coerce') ``` **Pourquoi ?** Certaines valeurs sont des espaces vides, ce qui empêche les calculs. #### Gérer les valeurs manquantes ```python # Imputation par la médiane pour TotalCharges df_churn['TotalCharges'].fillna(df_churn['TotalCharges'].median(), inplace=True) ``` **Alternatives possibles :** - Supprimer les lignes (si peu nombreuses) - Imputer par la moyenne - Imputer par un modèle prédictif #### Supprimer les colonnes inutiles ```python df_churn.drop('customerID', axis=1, inplace=True) ``` **Pourquoi ?** L'ID client est unique et n'apporte aucune information prédictive. --- ### **ÉTAPE 3 : FEATURE ENGINEERING** Créer de nouvelles variables pour améliorer les prédictions. #### **Feature 1 : ChargePerMonth** ```python df_churn['ChargePerMonth'] = df_churn['TotalCharges'] / (df_churn['tenure'] + 1) ``` **Logique :** - Cette variable capture le **montant moyen par mois** - Un ratio élevé peut

indiquer un service perçu comme cher - Le `+1` évite la division par zéro pour les nouveaux clients #### **Feature 2 : HasMultipleServices** ```python df_churn['HasMultipleServices'] = ((df_churn['OnlineSecurity'] == 'Yes') | (df_churn['OnlineBackup'] == 'Yes') | (df_churn['DeviceProtection'] == 'Yes')).astype(int) ``` **Logique :** - Mesure l'**engagement** du client - Plus de services = plus de raisons de rester - Variable binaire : 1 si au moins un service, 0 sinon #### **Autres features possibles (à explorer) :** ```python # Tenure en catégories df['TenureGroup'] = pd.cut(df['tenure'], bins=[0, 12, 24, 48, 72], labels=['0-1 an', '1-2 ans', '2-4 ans', '4+ ans']) # Ratio charges vs ancienneté df['ChargeRatio'] = df['MonthlyCharges'] / df['TotalCharges'] # Nombre total de services service_cols = ['PhoneService', 'InternetService', 'OnlineSecurity', ...] df['TotalServices'] = df[service_cols].apply(lambda x: (x == 'Yes').sum(), axis=1) ``` --- ### **ÉTAPE 4 : ENCODAGE DES VARIABLES** Les algorithmes de ML ne comprennent que les nombres, il faut donc encoder les variables catégorielles. #### **A. Label Encoding (variables binaires)** ```python binary_cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn'] le = LabelEncoder() for col in binary_cols: df_churn[f'{col}_encoded'] = le.fit_transform(df_churn[col]) ``` **Exemple :** ``` gender → gender_encoded Male → 1 Female → 0 ``` **Quand l'utiliser ?** Pour des variables à **2 catégories** avec une relation ordinale (ex: No=0, Yes=1). #### **B. One-Hot Encoding (variables multi-classes)** ```python multi_cols = ['InternetService', 'Contract', 'PaymentMethod'] df_churn = pd.get_dummies(df_churn, columns=multi_cols, drop_first=True) ``` **Exemple :** ``` Contract → Contract_One year Contract_Two year Month-to-month → 0 0 One year → 1 0 Two year → 0 1 ``` **Pourquoi drop_first=True ?** - Évite la **multicolinéarité** - Si Contract_One_year=0 et Contract_Two_year=0, alors c'est forcément Month-to-month --- ### **ÉTAPE 5 : PRÉPARATION POUR LA MODÉLISATION** #### **Séparer X (features) et y (cible)** ```python X = df_churn[feature_cols] # Variables indépendantes y = df_churn['Churn_encoded'] # Variable à prédire ``` #### **Split Train/Test** ```python X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y) ``` **Paramètres importants :** - `test_size=0.2` : 80% train, 20% test - `random_state=42` : Reproductibilité - `stratify=y` : **CRUCIAL** - maintient la proportion du churn dans train et test **Sans stratify :** ``` Train: 70% No, 30% Yes Test: 80% No, 20% Yes ← Problème ! ``` **Avec stratify :** ``` Train: 73.4% No, 26.6% Yes Test: 73.4% No, 26.6% Yes ← Proportions respectées ``` #### **Standardisation** ```python scaler = StandardScaler() X_train_scaled = scaler.fit_transform(X_train) X_test_scaled = scaler.transform(X_test) ``` **Pourquoi standardiser ?** **Avant :** ``` tenure : 1 à 72 mois MonthlyCharges : 18 à 120 dollars TotalCharges : 18 à 8700 dollars ← Échelle très différente ! ``` **Après standardisation :** ``` Toutes les variables : moyenne = 0, écart-type = 1 ``` **Important :** - `fit_transform` sur train (apprend les paramètres) - `transform` seulement sur test (utilise les paramètres du train) - **Ne JAMAIS standardiser avant le split** (data leakage !) --- ### **ÉTAPE 6 : ENTRAÎNEMENT DE PLUSIEURS MODÈLES** Tester différents algorithmes pour trouver le meilleur. #### **Modèle 1 : Régression Logistique** ```python log_reg = LogisticRegression(max_iter=1000, random_state=42) log_reg.fit(X_train_scaled, y_train) ``` **Avantages :** - Rapide et simple - Interprétable (coefficients) - Bon pour les relations linéaires **Inconvénients :** - Ne capture pas les interactions complexes - Sensible aux features non standardisées #### **Modèle 2 : Arbre de Décision** ```python dt = DecisionTreeClassifier(random_state=42) dt.fit(X_train, y_train) ``` **Avantages :** - Très interprétable (visualisation) - Pas besoin de standardisation - Capture les interactions non-linéaires **Inconvénients :** - Tendance au surapprentissage - Instable (petit changement de données → gros changement d'arbre) #### **Modèle 3 : Forêt Aléatoire** ```python rf = RandomForestClassifier(n_estimators=100, random_state=42) rf.fit(X_train, y_train) ``` **Avantages :** - Très performant - Résistant au surapprentissage - Gère bien les données manquantes - Fournit l'importance des variables **Inconvénients :** - Moins interprétable - Plus lent à entraîner #### **Modèle 4 : Gradient Boosting** ```python gb = GradientBoostingClassifier(random_state=42) gb.fit(X_train, y_train) ``` **Avantages :** - Souvent le plus performant - Excellent pour les compétitions Kaggle - Gère bien les déséquilibres **Inconvénients :** - Plus lent - Risque de surapprentissage - Beaucoup d'hyperparamètres --- ### **ÉTAPE 7 : ÉVALUATION ET COMPARAISON** #### **Métriques à analyser** Pour un problème de **churn** (classification déséquilibrée), on privilégie : **1. ROC-AUC (Area Under the Curve)** - Mesure la capacité du modèle à discriminer les classes - **0.5** : modèle aléatoire - **1.0** : modèle parfait - **> 0.8** : très bon modèle **2. Recall

(Sensibilité)** - Proportion de clients partis correctement identifiés - **Crucial pour le churn** : on veut détecter le maximum de départs Formule : Recall = TP / (TP + FN) **Exemple :** ``` 100 clients partis réellement Modèle détecte 80 → Recall = 80% 20 clients partis non détectés ← Perte ! ``` **3. Precision** - Proportion de prédictions "va partir" qui sont correctes - Important pour ne pas déranger les bons clients Formule : Precision = TP / (TP + FP) **4. F1-Score** - Moyenne harmonique de Precision et Recall - Équilibre entre les deux **Trade-off business :** ``` Recall élevé → Détecter tous les départs (mais faux positifs) Precision élevée → Cibler uniquement les vrais départs (mais manquer certains) ``` **Décision business :** Privilégier le **Recall** car le coût de perdre un client est élevé. --- ### **ÉTAPE 8 : INTERPRÉTATION ET RECOMMANDATIONS** #### **Importance des variables** ```python # Avec Random Forest feature_importance = pd.DataFrame({ 'Feature': X.columns, 'Importance': rf.feature_importances_ }).sort_values('Importance', ascending=False) ``` **Exemple de résultats (hypothétiques) :** ``` 1. tenure (ancienneté) → 0.25 ← Variable la plus importante 2. MonthlyCharges → 0.18 3. Contract_Month-to-month → 0.15 4. TotalCharges → 0.12 5. InternetService_Fiber → 0.10 ... ``` #### **Insights business** **1. Ancienneté (tenure)** - Les nouveaux clients (< 6 mois) ont un taux de churn de 50%+ - **Action :** Programme d'onboarding renforcé pour les 6 premiers mois **2. Type de contrat** - Contrats mensuels : 42% de churn - Contrats 1 an : 11% de churn - Contrats 2 ans : 3% de churn - **Action :** Inciter à passer sur des contrats longue durée (réductions) **3. Montant mensuel** - Clients payant > 70€/mois : risque élevé - **Action :** Offres personnalisées pour les "gros" clients **4. Services additionnels** - Clients sans services de sécurité : +30% de risque - **Action :** Promouvoir les services de valeur ajoutée --- ## 💼 APPLICATION PRATIQUE EN ENTREPRISE ### **Pipeline de production** ```python # 1. Scorer quotidiennement tous les clients clients_actifs = get_active_customers() X_clients = preprocess(clients_actifs) churn_probabilities = model.predict_proba(X_clients)[:, 1] # 2. Identifier les clients à risque clients_risque = clients_actifs[churn_probabilities > 0.7] # 3. Segmenter par niveau de risque clients_risque['risk_level'] = pd.cut(churn_probabilities, bins=[0.7, 0.8, 0.9, 1.0], labels= ['Medium', 'High', 'Critical']) # 4. Envoyer aux équipes de rétention for client in clients_risque[clients_risque['risk_level'] == 'Critical']: send_to_retention_team(client) propose_special_offer(client) ``` ### **Actions de rétention par segment** **Clients à risque CRITIQUE (90%+) :** - Appel téléphonique personnalisé - Offre exclusive -30% pendant 6 mois - Upgrade gratuit de service **Clients à risque ÉLEVÉ (80-90%) :** - Email personnalisé - Offre -20% pendant 3 mois - Consultation gratuite **Clients à risque MOYEN (70-80%) :** - Communication automatisée - Offre sur services additionnels - Programme de fidélité ### **ROI de la prédiction du churn** **Exemple chiffré :** ``` Base clients : 100 000 Taux de churn : 26% → 26 000 départs/an Valeur vie client (CLV) : 500€ Sans prédiction : Perte = 26 000 × 500€ = 13 000 000€ Avec prédiction (Recall 80%, taux rétention 40%) : Clients sauvés = 26 000 × 0.80 × 0.40 = 8 320 Gain = 8 320 × 500€ = 4 160 000€ Coût du programme : 500 000€ ROI = (4 160 000 - 500 000) / 500 000 = 732% ``` --- ## 📈 AMÉLIORATIONS POSSIBLES ### **1. Gérer le déséquilibre des classes** ```python from imblearn.over_sampling import SMOTE # Sur-échantillonner la classe minoritaire smote = SMOTE(random_state=42) X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train) ``` ### **2. Optimiser les hyperparamètres** ```python param_grid = { 'n_estimators': [100, 200, 300], 'max_depth': [10, 20, 30], 'min_samples_split': [2, 5, 10] } grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=5) grid_search.fit(X_train, y_train) ``` ### **3. Ajuster le seuil de décision** ```python # Au lieu de 0.5, utiliser un seuil optimisé pour le Recall threshold = 0.3 # Plus de clients détectés y_pred = (y_pred_proba > threshold).astype(int) ``` ### **4. Feature Engineering avancé** - Tendance des charges (augmentation/diminution) - Fréquence de contact avec le support - Score de satisfaction (si disponible) - Interactions entre variables --- ## 🎓 COMPÉTENCES ACQUISES À la fin de ce projet, vous maîtriserez : ✅ **Analyse métier** : Comprendre un problème business réel ✅ **Préparation de données** : Nettoyage, encodage, feature engineering ✅ **Modélisation ML** : Entraîner et comparer plusieurs algorithmes ✅ **Évaluation** : Choisir les bonnes métriques selon le contexte ✅ **Interprétation** : Traduire les résultats en actions business ✅ **Déploiement** : Pipeline de production et monitoring --- ## 📝 EXERCICES COMPLÉMENTAIRES **Exercice 1 :** Créer une nouvelle feature "EngagementScore" combinant plusieurs services **Exercice 2 :** Analyser l'impact du type

d'Internet (Fiber vs DSL) sur le churn **Exercice 3 :** Construire un modèle de scoring de 0 à 100 pour faciliter la communication **Exercice 4 :** Calculer le ROI d'une campagne de rétention basée sur vos prédictions **Exercice 5 :** Créer un dashboard Power BI/Tableau pour visualiser les résultats