Rapport

Projet Machine Learning avec PySpark Prédiction du Churn Télécom

---

Environnement technique

- Python 3 avec PySpark
- VS Code
- pyspark.ml, pandas,numpy
- Docker Desktop avec cluster Spark configuré

Description du modèle ML choisi

Objectif :

Le but du modèle est de prédire si un client d'un opérateur télécom va résilier son abonnement (churn) ou non. Cette prédiction permet d'anticiper les départs clients et de mettre en place des actions ciblées pour réduire le churn.

Algorithme :

Nous avons choisi d'utiliser un classificateur Random Forest implémenté avec PySpark MLlib. Random Forest est un ensemble d'arbres de décision qui réduit le surapprentissage et améliore la précision globale. C'est un algorithme robuste et efficace pour les données tabulaires hétérogènes.

Métriques :

Pour évaluer la performance du modèle, nous utilisons :

- Accuracy (précision globale)

Description du dataset utilisé

- Nom du fichier : Data4.csv
- Taille : 7043 lignes , 21 colonnes
- Format : CSV, avec une colonne cible Churn (valeurs binaires 0 ou 1)
- Source : Dataset fictif / téléchargeable sur un référentiel pédagogique (à adapter selon source réelle)
- Colonnes : Informations clients comme sexe, Client senior, contrat, Partenaire, Service téléphonique, etc. Certaines colonnes catégorielles doivent être encodées.

Détails sur l'adaptation du modèle à PySpark

- **SparkSession:** point d'entré de Spark.
- **StringIndexer:** Transforme du texte → indice numerique.
  EXEMPLE: Gender (Male/Female) → Gender_indexer (0/1)
- **VectorAssembler :** assemble plusieurs colonne en une seule features.
- **RandomForestClassifier:** Algorithme de classification binaire avancé.
- **MulticlassClassificationEvaluator:** Evaluation du modéle (calculer les métriques d'accuracy).

Fichier Spark churn_model.py

Initialisation du session Spark

```
spark = SparkSession.builder.appName("Churn-RF").getOrCreate()
```

créer une session Spark avec un nom "churn-RF" → obligatiore pour executer n'importe quel commande Spark.

Chargement du dataset

```
data = spark.read.csv("/app/Data4.csv", header=True, inferSchema=True)
```

charger Dataset : → header → 1ere ligne contient les noms du colonnes.
→ inferSchema → Spark detecte les types des colonnes automatiquement.

feature → est utilisée pour que le modéle va apprendre.

Data Split

```
train_data, test_data = final_data.randomSplit([0.8, 0.2], seed=42)
```

80 % → train-data
20% → test-data
seed=42 → decoupage toujours le meme.

Entrainer le modele de classification par RandomForrest

```
rf = RandomForestClassifier(numTrees=50)
model = rf.fit(train_data)
results = model.transform(test_data)
```

entrainer le modele par l'algorithme RandomForrest **.fit()**
transform() : applique le modele aux data du teste.

Evaluation du performance avec Accuracy

```
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
accuracy = evaluator.evaluate(results)
print(f" Accuracy du Random Forest : {accuracy:.2f}")
```

MulticlassClassificationEvaluator: Evaluation du modéle (calculer les métriques d'accuracy).

Fermeture de la session

```
spark.stop()
```

libére les ressources Spark a fin de l'execution.

Scripts d'exécution des données avec Spark

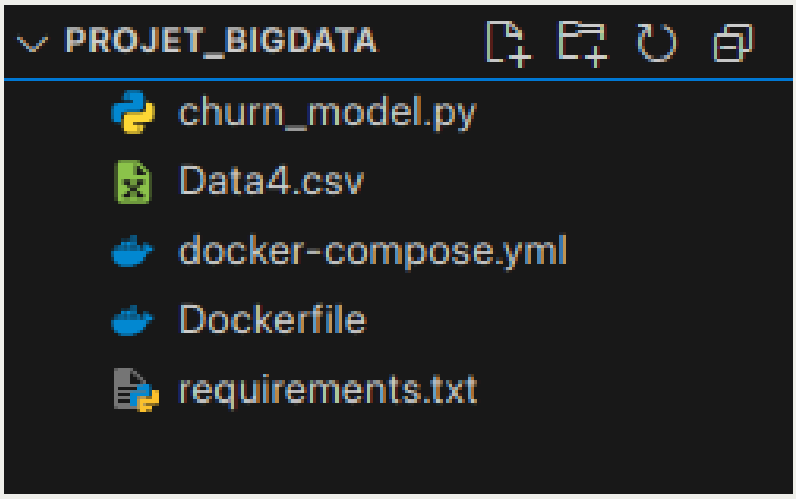
spark-submit churn_pyspark.py

Interprétation des résultats

Une accuracy de 80% montre que le modèle est capable de bien différencier les clients churners des non-churners.
Le random forest permet d'identifier les variables les plus importantes (feature importance), souvent la durée du contrat (tenure), le type de contrat, ou les charges mensuelles.
Les métriques de précision et rappel sont importantes pour équilibrer le coût des faux positifs (client non churner prédit churner) et faux négatifs (client churner non détecté).

```
25/05/15 08:33:17 INFO TaskSchedulerImpl: Killing al
nished
25/05/15 08:33:17 INFO DAGScheduler: Job 42 finished
cala:61, took 0.548507 s
 Accuracy du Random Forest : 0.80
25/05/15 08:33:17 INFO SparkContext: SparkContext is
25/05/15 08:33:17 INFO SparkUI: Stopped Spark web UI
25/05/15 08:33:17 INFO MapOutputTrackerMasterEndpoin
pped!
25/05/15 08:33:17 INFO MemoryStore: MemoryStore clea
25/05/15 08:33:17 INFO BlockManager: BlockManager st
```

Partie Docker

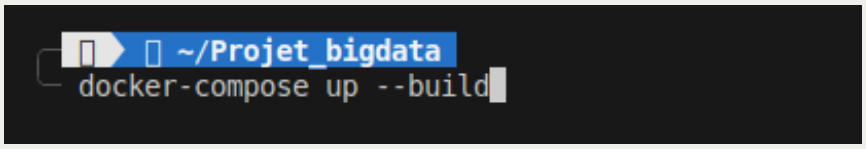


Transferer la dataset et le PySprak dans Docker

```
docker cp projet_final.py spark-master:/app/projet_final.py
```

```
docker cp Data4 spark-master:/app/data/Data4.py
```

Construire le docker-compose



Executer le Bash Spark Master