

Rapport du Mini Projet du module écosystème hadoop

Réalisé par :

- YENDOUBOUAM ALEXANDRE LALLE
- ABDELGHAFFOUR MOUHSINE

MASTER SDSI 2022-2023

INTRODUCTION

L'Organisation mondiale de la santé a estimé que 12 millions de décès surviennent dans le monde chaque année en raison de maladies cardiaques. La moitié des décès aux États-Unis et dans d'autres pays développés sont dus à des maladies cardio-vasculaires. Le pronostic précoce des maladies cardiovasculaires peut aider à prendre des décisions sur les changements de mode de vie chez les patients à haut risque et à son tour réduire les complications. Cette recherche vise à identifier les facteurs de risque les plus pertinents des maladies cardiaques et à prédire le risque global à l'aide de la régression logistique en prétraitant des données.

Objectif :

La tâche est de prédire si le patient a un risque de 10 ans de maladie coronarienne CHD ou non. En outre, les participants ont également demandé à créer une visualisation des données sur les données pour obtenir des informations exploitables sur le sujet.

Source :

La base de données est accessible au public sur le site Web de Kaggle et provient d'une étude cardiovasculaire en cours sur des résidents de la ville de Framingham, Massachusetts. L'objectif de la classification est de prédire si le patient présente un risque de maladie coronarienne (CHD) à 10 ans. L'ensemble de données fournit des informations sur les patients. Il comprend plus de 4 000 observations et 15 variables.

<https://www.kaggle.com/datasets/dileep070/heart-disease-prediction-using-logistic-regression>

Variables :

Chaque variable est un facteur de risque potentiel. Il existe à la fois des facteurs de risque démographiques, comportementaux et médicaux. Description des données Démographique :

- Sex: homme ou femme ("M" ou "F")
- Age: Âge du patient (Continu - Bien que les âges enregistrés aient été tronqués en nombres entiers, le concept d'âge est continu) Comportementale
- is_smoking: si le patient est ou non un fumeur actuel ("YES" ou "NO")
- CigsPerDay: le nombre de cigarettes que la personne a fumé en moyenne en une journée (peut être considéré comme continu car on peut avoir n'importe quel nombre de cigarettes, même une demi-cigarette). Antécédents médicaux)
- BPMeds: que le patient prenne ou non des médicaments contre l'hypertension (valeur nominale)
- preleventStroke: si le patient a déjà eu un accident vasculaire cérébral (nominal)
- prevalentHyp: si le patient était ou non hypertendu (nominal)
- Diabetes: si le patient était ou non diabétique (nominal)

Médical (actuel)

- totChol: taux de cholestérol total (en continu)
- sysBP: tension artérielle systolique (continue)
- diaBP: tension artérielle diastolique (continue)
- BMI: indice de masse corporelle (en continu)
- heartRate: fréquence cardiaque (continue - Dans la recherche médicale, des variables telles que la fréquence cardiaque sont en fait discrètes, mais sont considérées comme continues en raison du grand nombre de valeurs possibles.)
- Glucose: niveau de glucose (en continu)

Variable de prédiction (cible souhaitée)

- TenYearCHD : 10 ans de risque de maladie coronarienne CHD (binaire: «1», signifie «Yes», «0» signifie «No»).

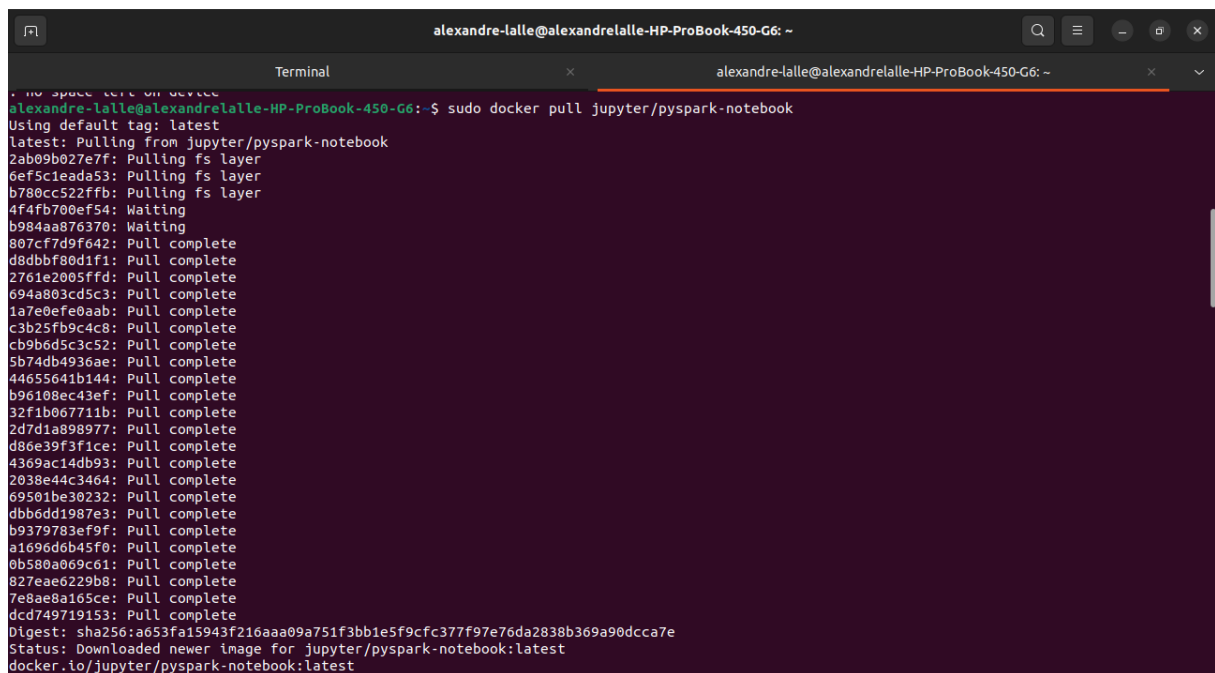
Réalisation

Nous avons réalisé un projet de régression logistique avec PySpark dans le but de prédire si un patient présente un risque de maladie coronarienne CHD dans les 10 prochaines années. En outre, nous avons également créé des visualisations pour obtenir des informations exploitables sur les données.

1. Installation de PySpark

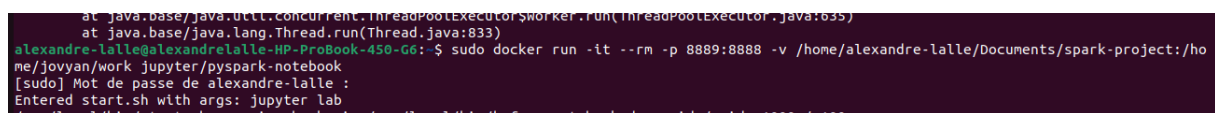
Nous avons utilisé le conteneur Docker jupyter/pyspark-notebook, qui offre un environnement de travail basé sur Jupyter Notebook et contient PySpark préinstallé.

- Téléchargement de l'image docker



```
alexandre-lalle@alexandre-lalle-HP-ProBook-450-G6: ~  
$ sudo docker pull jupyter/pyspark-notebook  
Using default tag: latest  
latest: Pulling from jupyter/pyspark-notebook  
2ab09b027e7f: Pulling fs layer  
6ef5c1eada53: Pulling fs layer  
b780cc522ffb: Pulling fs layer  
4f4fb700ef54: Waiting  
b984aa876370: Waiting  
807cf7d9f642: Pull complete  
d8dbbf80d1f1: Pull complete  
2761e2005ffd: Pull complete  
694a803cd5c3: Pull complete  
1a7e0efe0aab: Pull complete  
c3b25fb9c4c8: Pull complete  
cb9b6d5c3c52: Pull complete  
5b74db4936ae: Pull complete  
44655641b144: Pull complete  
b96108ec43ef: Pull complete  
32f1b067711b: Pull complete  
2d7d1a898977: Pull complete  
d86e39f3fice: Pull complete  
4369ac14db93: Pull complete  
2038e44c3464: Pull complete  
69501be30232: Pull complete  
dbb6dd1987e3: Pull complete  
b9379783ef9f: Pull complete  
a1696d6b45f0: Pull complete  
0b580a069c61: Pull complete  
827eae6229b8: Pull complete  
7e8ae8a165ce: Pull complete  
dcd749719153: Pull complete  
Digest: sha256:a653fa15943f216aaa09a751f3bb1e5f9cfc377f97e76da2838b369a90dcca7e  
Status: Downloaded newer image for jupyter/pyspark-notebook:latest  
docker.io/jupyter/pyspark-notebook:latest
```

- Exécution du conteneur



```
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)  
at java.base/java.lang.Thread.run(Thread.java:833)  
alexandre-lalle@alexandre-lalle-HP-ProBook-450-G6: ~$ sudo docker run -it --rm -p 8889:8888 -v /home/alexandre-lalle/Documents/spark-project:/home/jovyan/work jupyter/pyspark-notebook  
[sudo] Mot de passe de alexandre-lalle :  
Entered start.sh with args: jupyter lab  
/usr/local/bin/start.sh: running books in /usr/local/bin before notebook does uid: 1000 / 1000
```

La documentation officiel et le lien GitHub de l'image docker sont accessibles via ce lien : <https://hub.docker.com/r/jupyter/pyspark-notebook>

2. Exploration des données :

Nous avons commencé par explorer les données pour comprendre leur structure et identifier les variables pertinentes pour notre modèle. Nous avons effectué des analyses univariées et bivariées pour comprendre les relations entre les variables.

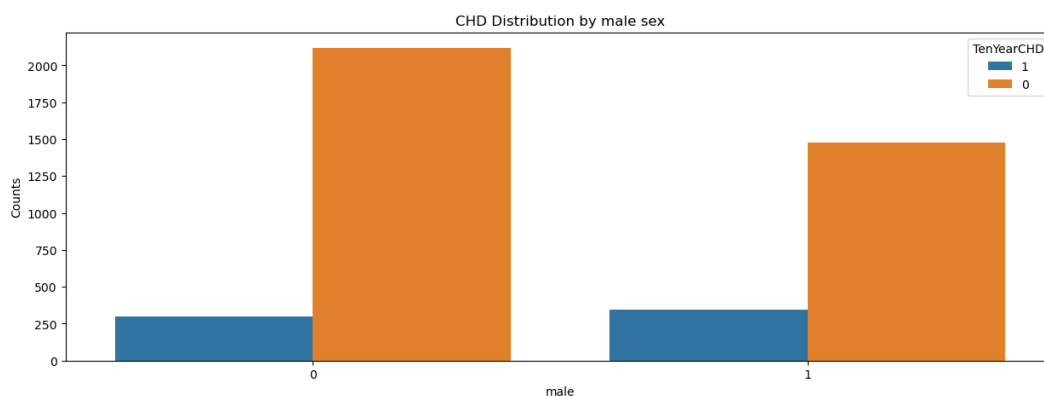
	Sex_male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	0	0	0	0	0	0	195	106	70	26.97	80	77	0
1	0	46	0	0	0	0	0	0	250	121	81	28.73	95	76	0
2	1	48	1	20	0	0	0	0	245	127.5	80	25.34	75	70	0
3	0	61	1	30	0	0	1	0	225	150	95	28.58	65	103	1
4	0	46	1	23	0	0	0	0	285	130	84	23.1	85	85	0

- Nombre de personnes ayant la maladie :

```
In [71]: heart_df.groupby('TenYearCHD').count().show()
```

```
+-----+-----+
| TenYearCHD | count |
+-----+-----+
|           0 | 3594 |
|           1 |  644 |
+-----+-----+
```

- Nombre de personnes ayant la maladie par rapport au sexe :



3. Sélection des variables :

Après avoir exploré les données et effectué un prétraitement initial, nous avons sélectionné les variables les plus pertinentes pour notre modèle avec la meilleure corrélation avec la variable de prédiction. Cela a inclus la suppression de variables redondantes ou peu informatives.

Vérification de la corrélation avec TenYearCHD

```
In [72]: import pyspark.sql.functions as F

correlations = heart_df.select([F.corr(F.col('TenYearCHD'), F.col(c)).alias(c) for c in heart_df.columns]).toPandas()
correlations = correlations.T*100
correlations
```

```
Out[72]:
```

	0
Sex_male	8.842757
age	22.525610
education	-5.405896
currentSmoker	1.945627
cigsPerDay	5.788426
BPMeds	8.748858
prevalentStroke	6.180995
prevalentHyp	17.760273
diabetes	9.731651

Par exemple la variable « education » à une corrélation négative donc elle n'est pas utile à l'étude.

4. Prétraitement des données :

Nous avons effectué des opérations de nettoyage des données pour éliminer les valeurs manquantes, les doublons et les valeurs aberrantes. Nous avons également effectué des transformations de données telles que la normalisation ou la standardisation pour améliorer la performance de notre modèle.

- Type des données avant prétraitement

```
In [68]: heart_df.printSchema()

root
 |-- Sex_male: string (nullable = true)
 |-- age: string (nullable = true)
 |-- education: string (nullable = true)
 |-- currentSmoker: string (nullable = true)
 |-- cigsPerDay: string (nullable = true)
 |-- BPMeds: string (nullable = true)
 |-- prevalentStroke: string (nullable = true)
 |-- prevalentHyp: string (nullable = true)
 |-- diabetes: string (nullable = true)
 |-- totChol: string (nullable = true)
 |-- sysBP: string (nullable = true)
 |-- diaBP: string (nullable = true)
 |-- BMI: string (nullable = true)
 |-- heartRate: string (nullable = true)
 |-- glucose: string (nullable = true)
 |-- TenYearCHD: string (nullable = true)
```

```
In [73]: # Suppression de La colonne 'education'
heart_df = heart_df.drop("education")

# Conversion des colonnes string en double
for col_name in ["age", "cigsPerDay", "totChol", "sysBP", "diaBP", "glucose", "TenYearCHD", "prevalentHyp"]:
    heart_df = heart_df.withColumn(col_name, col(col_name).cast("double"))
|

# Indexation et encodage des colonnes catégorielles
indexer = StringIndexer(inputCol='Sex_male', outputCol='Sex_maleIndex')
indexed = indexer.fit(heart_df).transform(heart_df)

encoder = OneHotEncoder(inputCols=['Sex_maleIndex'], outputCols=['Sex_maleIndexVec'])
encoded = encoder.fit(indexed).transform(indexed)

# Assemblage des colonnes en un vecteur de caractéristiques
assembler = VectorAssembler(inputCols=["age", "Sex_maleIndexVec", "cigsPerDay", "totChol", "sysBP", "diaBP", "glucose", "prevalentHyp", "diabetes", "BMI", "heartRate", "TenYearCHD"], outputCol='features')
output = assembler.transform(encoded)

# Sélection des colonnes à utiliser pour l'entraînement
model_df = output.select(['features', 'TenYearCHD'])
```

- Type des données après prétraitement

```
In [74]: encoded.printSchema()

root
 |-- Sex_male: string (nullable = true)
 |-- age: double (nullable = true)
 |-- currentSmoker: string (nullable = true)
 |-- cigsPerDay: double (nullable = true)
 |-- BPMeds: string (nullable = true)
 |-- prevalentStroke: string (nullable = true)
 |-- prevalentHyp: double (nullable = true)
 |-- diabetes: string (nullable = true)
 |-- totChol: double (nullable = true)
 |-- sysBP: double (nullable = true)
 |-- diaBP: double (nullable = true)
 |-- BMI: string (nullable = true)
 |-- heartRate: string (nullable = true)
 |-- glucose: double (nullable = true)
 |-- TenYearCHD: double (nullable = true)
 |-- Sex_maleIndex: double (nullable = false)
 |-- Sex_maleIndexVec: vector (nullable = true)
```

Les variables sélectionnés pour l'entraînement ont été convertie en type double pour pouvoir être traiter par le modèle.

5. Modélisation :

Nous avons utilisé PySpark pour ajuster un modèle de régression logistique à nos données. Nous avons également effectué répartition des données en ensemble d'entraînement et ensemble de teste.

```
In [75]: # Division des données en ensembles d'entraînement et de test
train_data, test_data = model_df.randomSplit([0.75, 0.25], seed=42)

# Construction du modèle de régression logistique
lr = LogisticRegression(featuresCol="features", labelCol="TenYearCHD")
lr_model = lr.fit(train_data)
```

6. Évaluation du modèle :

Après avoir ajusté notre modèle, nous avons évalué sa performance à l'aide de métriques telles que l'exactitude et la précision. Nous avons également examiné la matrice de confusion pour évaluer la capacité de notre modèle à détecter les vrais positifs et les faux positifs.

- Données réelles de l'ensemble de test :

```
In [80]: test_data.groupBy('TenYearCHD').count().show()
```

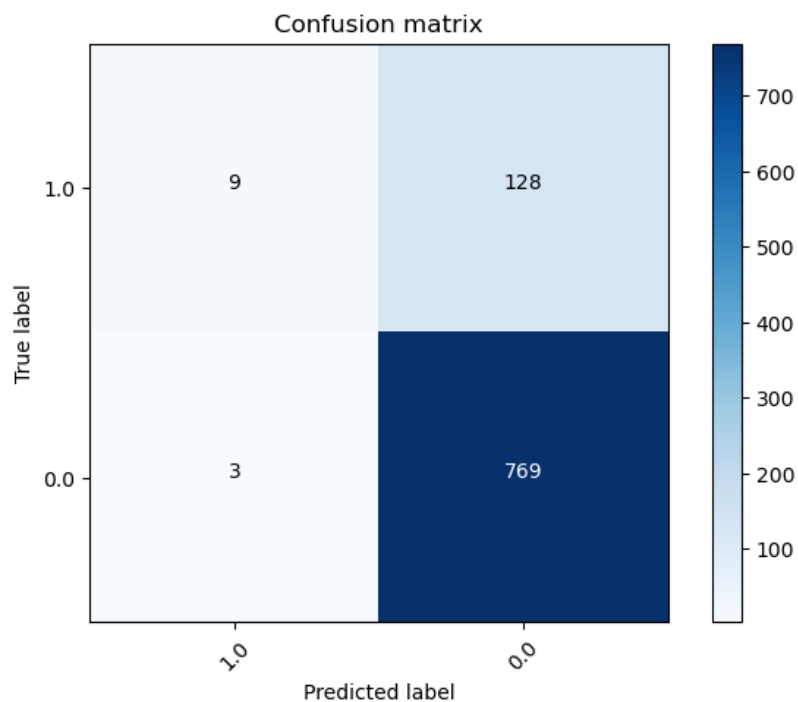
TenYearCHD	count
0.0	772
1.0	137

- Prédictions du modèle sur l'ensemble de test :

TenYearCHD	prediction	count
0.0	0.0	769
0.0	1.0	3
1.0	0.0	128
1.0	1.0	9

Accuracy: 0.8558855885588559

Nous avons obtenu une précision de 85%.



Conclusion

En conclusion, notre modèle de régression logistique a démontré une précision de 85% dans la prédiction du risque de maladie coronarienne CHD chez les patients pour les 10 prochaines années, ce qui témoigne de son efficacité. De plus, nous avons pu identifier les variables les plus pertinentes pour la prédiction, ce qui peut fournir des informations importantes pour la prévention et le traitement de la maladie.