UNIVERSITÉ TÉLUQ

TRAVAIL NOTÉ 2

PRÉSENTÉ À

FATIMA BENSALMA

COMME EXIGENCE PARTIELLE

DU COURS

PROJET EN SCIENCE DES DONNÉES SCI 1402

PAR

GABRIEL FLEURENT-THIFFAULT

22307023

TRAVAIL NOTÉ 2

22 MAI 2025

Table des matières

Rencontre avec les données

Formulation de l'hypothèse

L'hypothèse de départ avait été formulée au moment de la validation du projet, mais n'avait pas été expliquée en détail, d'où le pourquoi de cette partie.

Donc, voici un rappel : plus une personne est âgée et que celle-ci a un haut taux de cholestérol, plus elle a de risques de développer des maladies cardiaques.

Cette hypothèse se veut être de type statistique et elle se décline en deux hypothèses :

- L'hypothèse nulle (H_0) qui stipule que l'âge et le taux de cholestérol n'ont aucun effet sur les risques de maladies cardiaques.
- Tandis que l'hypothèse alternative (H_a) démontre que ces paramètres ont un effet significatif sur les dangers de développer des maladies du cœur (Mazerolle, 2019).

Analyser et comprendre l'ensemble des colonnes, pour bien saisir leur rôle

Selon l'hypothèse de départ, en théorie, il fallait uniquement se baser sur l'âge et le cholestérol. Cependant, en visionnant les autres colonnes, la colonne « num » est apparue aussi capitale que les deux autres. Il sera question de cette colonne, plus loin dans ce travail.

Âge

En se basant sur le site du gouvernement du Québec, il a été question de déterminer à quel âge une personne devenait âgée. Mais la réponse à cette question n'a pas été explicitement donnée. Il a donc été conclu que l'âge d'une personne âgée est de 65 ans et plus ; puisqu'ils se basent majoritairement sur cet âge pour les représenter (Institut de la statistique du Québec, 2023).

Cholestérol

Le cholestérol est constitué en deux catégories : lipoprotéine de basse densité (LDL) et lipoprotéine de haute densité (HDL).

Le HDL, le bon cholestérol qui permet d'éliminer la surabondance de cholestérol dans les artères

Le LDL est celui qui est considéré comme mauvais, puisque formé principalement de cholestérol. Quand il y en a trop, cela peut causer de l'athérosclérose. L'athérosclérose est l'accumulation de dépôts graisseux dans les artères et leurs parois. Celles-ci peuvent entrainer une obstruction des artères ou des caillots sanguins.

De cette façon, s'il y a présence d'un taux de cholestérol supérieur à 200 mg / litre, nous sommes devant un cas d'hypercholestérolémie et un niveau dangereux pour la santé (Elsan, 2025).

Collecte de données

Pour la collecte des données, le répertoire GitHub a été créé d'avance et le fichier téléchargé depuis kaggle a été déposé. Cette option a été préconisée, puisque Kaggle permettait uniquement le chargement en .zip.

De plus, le dossier « donnees » a été créé, dans Google Colab, afin d'avoir une bonne structure de dossiers.

```
# Création des dossiers donnees/csv et ajout de la base de données UCI

!wget -P donnees/csv/ https://raw.githubusercontent.com/thiffaultg-teluq/sci1402/main/donnes/brutes/heart_disease_uci.csv
```

Dans la même lignée, un dossier présent par défaut, soit celui « sample_data » a été supprimé :

```
# Suppression du dossier inutile

!rm -rf sample_data/
```

Avec les données maintenant accessibles via Google Colab (notebook Jupyter), les traitements suivants ont pu être apportés : supprimer les rangées ayant des cellules vides (« age » et « chol ») et conserver que les données fiables, donc de soustraire les données aberrantes.

Une erreur dans le travail noté 1 a été identifiée. En effet, il était mentionné à l'étape 2.3.1 que la normalisation des colonnes « Cp », « Origin », « Thal », « Restecg » et « Slope » allait être faite. Mais finalement, ce sera une conversion en valeurs numériques.

Traitement des données manquantes

Dans le cadre du cours SCI 1017, nous n'avions pas abordé directement la suppression de colonnes ayant des cellules vides. Nous avions abordé la transformation *drop*, mais sans plus.

En effectuant des recherches sur internet, de l'information pertinente a été trouvée, aidant à élucider le questionnement (Sparkbyexamples, 2024)

CODE :

```
# Ménage | Suppression des cellules vides :
uci="donnees/csv/heart_disease_uci.csv"
df = spark.read.options(header='true', inferSchema='true').csv(uci)
df_apres_supp = df.na.drop(subset=["age", "chol"])

# Vérification avant/après
print("Avant:", df.count())
print("Après:", df_apres_supp.count())
```

À noter que la suppression des cellules vides ont été faites uniquement en se fiant à l'hypothèse initiale, soit: « age » et « chol ». Puisqu'avec la supposition émise, seulement ces deux colonnes sont importantes. Qu'il y ait des données manquantes dans la colonne « thal » ne changera rien au résultat final.

Un « avant » et « après » a été ajouté, pour s'assurer qu'il y avait assez de rangées dans les données, à la suite des suppressions.

Avant:	920

Après:	890

Après le traitement des données manquantes, uniquement 30 colonnes avaient été supprimées : soit 3.26% des données initiales

Traitement des données aberrantes

Par la suite, les données aberrantes ont pu être traitées. Encore une fois, nous avions couvert cette transformation dans le cours SCI 1016 (Université TÉLUQ, 2021), mais pas concrètement dans un vrai projet. Or, quelques recherches sur internet ont été effectuées et une bonne solution a été trouvée : la méthode Écart interquartile. Celle-ci ressemble énormément à ce que nous avons pu voir au cours SCI 1016

Ce script a permis de déterminer les bornes inférieures et supérieures des données aberrantes pour les colonnes « age » et « chol » (Machinelearningplus, 2025) :

CODE :

```
# Gestion des données aberrantes

# CHOL

def iqr_outlier_treatment(df, columns=["chol"], factor=1.5):
  for column in columns:
    quantiles = df_apres_supp.approxQuantile(column, [0.25, 0.75], 0.01)
    q1, q3 = quantiles[0], quantiles[1]
    iqr = q3 - q1
    lower_bound = q1 - factor * iqr
    upper_bound = q3 + factor * iqr
    print("Lower : ", lower_bound)
    print("Upper : ", upper_bound)
    df = df.filter((col(column) >= lower_bound) & (col(column) <=
upper_bound))
  return df

df_fiable = iqr_outlier_treatment(df_apres_supp)

print(df_fiable.count())
```

Où le facteur à été fixé à 1.5, lui qui aurait pu être entre 1.5 et 33 (Université TÉLUQ, 2021).

Pour éviter la duplication de code, l'exemple ci-dessus représente uniquement « chol ». Soit la variable surlignée en vert, la seule différence avec « age ».

Ainsi, il a été possible d'émettre que la borne inférieure était égale à 32 et la borne supérieure égale à 408 pour « chol ».

Pour l'« age », la borne inférieure était de 27.5 et celle supérieure à 79.5. Comme nous pourrons le voir avec les boites à moustaches, les données aberrantes de cette colonne ne seront pas traitées ; puisqu'il n'y avait rien à transformer.

Pour confirmer ces transformations, deux diagrammes en boite ont été créés : un pour les âges et l'autre pour le cholestérol.

En partant du code donné au point « Calling the iqr_outlier_treatment() function » (Machinelearningplus, 2025) , l'erreur suivante est apparue : `AttributeError: 'Series' object has no attribute 'join'.` En analysant le code, la déduction a été faite, il fallait créer un nouveau dataframe. En suivant l'exemple donné « Creating a DataFrame from Another DataFrame », le code a pu être adapté (Geeksforgeeks, 2025). Pour décortiquer et assimiler l'information, le site Pandas a été consulté (Pandas, 2025).

CODE:

```
# Diagramme à moustache

  # AGE

pdDF = df_apres_supp.toPandas()

df_modifie = df_fiable.toPandas()

df_age = pd.DataFrame({

    'age_original': pdDF['age'],

    'age_traité': df_modifie['age'],

})

df_age.boxplot(column =['age_original','age_traité'], grid = False,
figsize=(16,9))


# Diagramme à moustache (Suite)

# CHOL

df_chol = pd.DataFrame({

    'chol_original': pdDF['chol'],

    'chol_traité': df_modifie['chol']

})

df_chol.boxplot(column =['chol_original','chol_traité'], grid = False,
figsize=(16,9))
```

Donc, après analyse, décision de ne traiter que la colonne « chol » a été prise, puisqu'« age » n'apportait aucun changement significatif.

Avec les données aberrantes traitées de « chol ». Cela a permis d'exclure les personnes ayant un cholestérol à 0.

Les résultats du diagramme en boite pour les âges sont disponibles à l'Annexe A et ceux du cholestérol à l'Annexe B.

Conversions des catégories en valeurs numériques

Il est important de convertir les colonnes contenant des catégories en valeurs numériques, puisque les ordinateurs n'assimilent pas les descriptions, mais comprennent les chiffres (Medium, 2024).

Pour chacune des colonnes représentant des catégories, la même procédure a été appliquée :

1. Premièrement, la liste complète de la catégorie était regardée avec la fonction **distinct** :

```
df_fiable.select("slope").distinct().show()
```

Deuxièmement, avec les fonctions **withColumn** et **when**, un chiffre était attribué pour chaque description (Stackoverflow, 2017) :

```
df_fiable = df_fiable.withColumn(
    "slope",
    when(df_fiable.slope == "flat", 1)
    .when(df_fiable.slope == "downsloping", 2)
    .when(df_fiable.slope == "upsloping", 3)
    .otherwise(None)
)
```

Donc, les colonnes suivantes ont subi des conversions, puisqu'elles représentaient toutes des catégories.

1. thal;
2. sex;
3. dataset;
4. cp;
5. restecg;
6. slope .

Produit final et présentation

Analyser et apprendre

Analyse partielle de l’hypothèse de départ

En sélectionnant les personnes âgées de 65 et plus et ayant un cholestérol de plus de 200 mg / litre, soit le seuil dangereux à ne pas atteindre. 49.23% (32 personnes sur 65) de l’échantillon avait un seuil supérieur ou égal à 1 dans la prédiction d’avoir des problèmes cardiaques.

La décision de considérer tout ce qui est plus grand ou égal à 1 pour la colonne « num » a été faite : en se basant sur cette grille (Kaggle, 2025) :

0 = no heart disease
1 = mild heart disease
2 = moderate heart disease
3 = severe heart disease
4 = critical heart disease

De cette façon, oui, les maladies cardiaques légères sont prises en compte, mais celles-ci ne sont pas encore à un stade avancé. Donc le stade de la maladie cardiaque est encore précoce et ce choix permet également d’avoir une plus grande portée.

Il y a une diminution marquée comparée à l’échantillon initial, mais n’en demeure pas moins que celle-ci est considérée comme étant grande : puisqu’elle a nombre supérieur à 30 éléments (Université TÉLUQ, 2022).

Devant ce constat, il n’est pas possible de rejeter l’hypothèse nulle de : puisque le pourcentage (49.23%) n’est pas suffisant.

Avec le code ci-dessous, il est possible de voir les filtres qui ont été utilisés pour venir chercher les personnes âgées de plus 65 ans, ayant un taux de cholestérol supérieur à 200 mg / litre et un diagnostic supérieur ou égal à 1 (Chikhaoui, 2025). De plus, il est aussi possible de constater la création d’un fichier .csv avec un nom fixe (Stackoverflow, 2017).

CODE :

```
# Application des filtres

df_fiable_filtre = df_fiable.filter((col("age") >= 65) & (col("chol") > 200) & (col("num") >= 1))
```

```
#Création du fichier .csv avec un nom unique

import os

import shutil

TEMPORARY_TARGET="/content/donnees/csv/temp"

DESIRED_TARGET="/content/donnees/csv/donnees-filtrees.csv"

df_fiable_filtre.coalesce(1).write.option("header",
"true").csv(TEMPORARY_TARGET)

part_filename = next(entry for entry in os.listdir(TEMPORARY_TARGET) if
entry.startswith('part-'))

temporary_csv = os.path.join(TEMPORARY_TARGET, part_filename)

shutil.copyfile(temporary_csv, DESIRED_TARGET)
```

Étapes d’estimation en appliquant un algorithme d'apprentissage machine (régression logistique)

La régression logistique a été choisie plutôt que la régression linéaire, puisque dans ce travail, il était question de pouvoir prédire si un patient était plus propice à avoir des maladies cardiaques en prenant uniquement son âge et son taux de cholestérol. Dans le but, de pouvoir catégoriser les patients en deux groupes : atteint de problèmes cardiaques et sans problème cardiaque (Chikhaoui, 2024).

Pour la création du modèle, celui présenté au cours SCI 1017 en exemple 2 : Classification avec Régression logistique a été grandement utilisé (Chikhaoui, 2024).

Les différences à noter :

1. Au lieu d’utiliser des données aléatoires, les données de la base de données ont été utilisées. Ce afin de déterminer à partir des données réelles si oui ou non l’âge et le cholestérol ont un impact significatif sur les maladies cardiaques.
2. Pour la création des deux groupes : entrainement et test le module K-fold de l’extension sklearn a été utilisé. En ayant K égalant 3 (n_splits) : le groupe d’entrainement était de 30 personnes et celui de test de 14. La variable `random_state` a été mis à 49, nombre choisi entièrement de façon aléatoire, pour que le résultat soit toujours le même (Geeksforgeeks, 2024)

CODE :

```
from pyspark.ml.feature import VectorAssembler

from pyspark.ml.classification import LogisticRegression

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

from sklearn.model_selection import KFold

import pandas as pd

df_filtre = spark.read.csv("/content/donnees/csv/donnees-filtrees.csv", header=True, inferSchema=True)

filtre_pd = df_filtre.toPandas()

# Paramètrage de KFOLD :

  #n_splits = 3 => Entrainement 2/3 | Test 1/3

kf = KFold(n_splits=3, shuffle=True, random_state=49)

# Initiation des deux groupes :

for train_index, test_index in kf.split(filtre_pd):

  train_pd = filtre_pd.iloc[train_index]

  test_pd = filtre_pd.iloc[test_index]

# Charger les données d'entraînement

train_df = spark.createDataFrame(train_pd)
```

```
# Charger les données de test

test_df = spark.createDataFrame(test_pd)

# Préparer l'assembleur de vecteurs

cols = ["age", "chol"]

assembler = VectorAssembler(inputCols=cols, outputCol="features")

# Appliquer l'assembleur de vecteurs

train_df = assembler.transform(train_df)

test_df = assembler.transform(test_df)

# Renommer la colonne de label

train_df = train_df.withColumnRenamed("num", "label")

test_df = test_df.withColumnRenamed("num", "label")

# Entraîner le modèle de classification par régression logistique

lr = LogisticRegression(labelCol="label", featuresCol="features", maxIter=10)

model = lr.fit(train_df)

# Faire des prédictions et évaluer le modèle

predictions = model.transform(test_df)

# Afficher les prédictions

predictions.select("chol","age","label", "prediction").show()

# Calculer la précision du modèle

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")

accuracy = evaluator.evaluate(predictions)

print(f"Accuracy: {accuracy}")
```

Évaluation des performances en utilisant la métrique précision et la validation croisée.

Afin de déterminer si la précision du modèle concorde avec l'hypothèse ou non, l'application de cette formule a été préconisée (Chikhaoui, 2024) :

```
# Faire des prédictions et évaluer le modèle

predictions = model.transform(test_df)

# Afficher les prédictions

predictions.select("label", "prediction").show()

# Calculer la précision du modèle

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")

accuracy = evaluator.evaluate(predictions)

print(f"Accuracy: {accuracy}")
```

Avec le résultat retourné : Accuracy: 0.42857142857142855. Ce résultat est encore plus bas que l'analyse partielle. Donc, il est de plus en plus évident que l'hypothèse nulle ne pourra être exclue : 42.86% d'exactitude de précision n'est définitivement pas suffisant, pour rejeter l'hypothèse nulle.

En étant sur le site pour faire le K-Fold, le sujet du *cross-validation* a été abordé (Geeksforgeeks, 2024). Mais l'importance de ce concept n'a pas été prise en compte dès le départ. C'est par après que le désir d'en apprendre davantage est survenu. Ceci explique en grande partie le retard dans la planification.

La validation croisée permet d'avoir des « modèles performants et solides ». Ceci est possible avec la division des données en plusieurs parts (plis) et par des cycles d'apprentissage et des tests enchaînés avec des variations dans les agencements de ces plis.

Plusieurs types existent, mais celui « Stratified K-Fold Cross-Validation » a été choisi : pour avoir une répartition égale de chacune des classes dans chaque pli. De ce fait, aucune sous-représentation n'est possible (Intelligence-artificielle-school, 2025).

Ainsi, après plusieurs recherches, une solution fonctionnelle a été trouvée (Medium, 2024).

CODE :

```
# import libs

from sklearn.preprocessing import MinMaxScaler

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import cross_val_score

import numpy as np
```

```
X = filtre_pd[["age", "chol", "cp"]]
Y = filtre_pd["num"]
scaler = MinMaxScaler()
XX = scaler.fit_transform(X)
clf = LogisticRegression(random_state=4)
cv_scores = cross_val_score(clf, XX, Y, cv = 3)
print('Scores de validation croisée (3 plis) :', cv_scores)
print('Précision moyenne :', np.mean(cv_scores))
```

La moyenne de la précision des trois plis est de 43.17 %, toujours dans le même ordre d'idée que la précision précédente.

En se basant sur ces précisions, il est possible de dire que l'hypothèse nulle ne peut être repoussée : puisqu'elles sont en deçà de 70%, soit le standard utilisé généralement pour établir une bonne précision (Fiddler, 2025).

Ainsi, l'âge et le cholestérol à eux seuls ne peuvent expliquer le développement des maladies cardiaques. En effet, d'autres paramètres peuvent expliquer ces maladies : le sexe, la glycémie à jeun (FBS), la tension artérielle au repos, etc.

Avec l'aide du tableau de bord, il sera possible de confirmer le tout.

Objectifs futurs

Déploiement du modèle

Pour donner suite à la modélisation de la solution, la prochaine étape sera de partager la solution via GitHub.

Qu'est-ce qui sera partagé via Github?

1. Comme mentionné précédemment, les données extraites de kaggle.
2. La modélisation de la solution faite à partir de Google Colab sera transmise au format *Jupyter Notebook* (ipynb). De plus, tous les fichiers générés avec *Google Colab* seront également ajoutés.
3. Pour mettre dans le contexte les éventuels visiteurs du répertoire, il sera mis à leur disposition l'ensemble des travaux pratiques faits dans le cadre de ce cours.

Normes et spécifications

Les normes et spécifications à respecter sont les suivantes :

- Utilisation de Git Flow pour la création des branches. Cette utilisation n'est pas nécessaire pour ce projet. En effet, mais simplement par habitude de travail et par souci de qualité et de versionnage potentiels : Git Flow fera partie de mes normes.
- Chaque « commit » sera préfixé par un verbe d'action (Freecodecamp, 2022), ce dans le but de respecter les bonnes pratiques.
- Un fichier README.md sera inclus pour présenter l'essentiel du projet
- Finalement, la structure de dossiers suivante sera respectée :
    1. Données
        a. Brute
    2. Modélisation (Jupyter Notebook)
        a. Ressources
            i. Fichiers .csv
            ii. Images
    3. Travail pratique
        a. Préparation de la recherche
        b. Validation du choix de projet
        c. Plan de projet documenté
        d. Rapport mi-parcours
        e. Rapport du projet présenté

Restitution des résultats

Indicateurs (KPI)

Il serait judicieux de commencer avec l'âge moyen et le taux moyen de cholestérol des patients. Par la suite, le calcul de la précision (nombre de patients où la prédiction est égale au *num*). Finalement, le pourcentage du nombre de patients jugés avec un stade avancé (num 2 et plus).

Graphiques

Pour ce qui a trait aux graphiques, un histogramme sera assurément introduit, lui, qui viendrait exposer le nombre de patients par les diagnostics donnés (*num*).

Pour le moment, ce sont les KPI et les graphiques dont il est question. Néanmoins, il reste encore beaucoup de temps, pour y réfléchir et apporter des modifications.

Bibliographie

Chikhaoui, B. (2025). Module 3 | Spark [Notes fournies dans le cours SCI 1017]. SCI 1017 vm3: Section 3.4 | TELUQ

Elsan. (2025). *Cholestérol*. Repéré le 15 avril 2025 à https://www.elsan.care/fr/pathologie-et-traitement/biologie-medicale/cholesterol-definition

Fiddler. (2025). *Which is more important: model performance or model accuracy?* Repéré le 19 mai 2025 à https://www.fiddler.ai/model-accuracy-vs-model-performance/which-is-more-important-model-performance-or-model-accuracy

Freecodecamp. (2022). *How to Write Better Git Commit Messages – A Step-By-Step Guide*. Repéré le 18 mai 2025 à https://www.freecodecamp.org/news/how-to-write-better-git-commit-messages/

Geeksforgeeks. (2024, 27 mai). *Cross-Validation Using K-Fold With Scikit-Learn*. Repéré le 10 mai 2025 à https://www.geeksforgeeks.org/cross-validation-using-k-fold-with-scikit-learn/

Geeksforgeeks. (2025, 2 janvier). *Different ways to create Pandas Dataframe*. Repéré le 4 mai 2025 à https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/

Institut de la statistique du Québec. (2023). *Portrait des personnes aînées au Québec*. Repéré le 15 avril 2025 à https://statistique.quebec.ca/fr/communique/portrait-personnes-ainees-quebec

Intelligence-artificielle-school. (2025). *Cross-validation : quel est ce processus de validation croisée ?*. Repéré le 14 mai 2025 à https://www.intelligence-artificielle-school.com/ecole/technologies/cross-validation-croisee/

Kaggle. (2025). UCI Heart Disease Data. Repéré le 14 avril 2025 à https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data

Kaggle. (2025). UCI Heart Disease Data. Repéré le 9 mai 2025 à https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data/discussion/485099

Machinelearningplus. (2025). *PySpark Outlier Detection and Treatment – A Comprehensive Guide How to handle Outlier in PySpark.* Repéré le 27 avril 2025 à https://www.machinelearningplus.com/pyspark/pyspark-outlier-detection-and-treatment/

Mazerolle, M J. (2019). *Tests d'hypothèse sur un seul groupe* [Recueil de notes du cours SCI 1018]. Université TÉLUQ.

Medium. (2024, 4 novembre). *Cross Validation with Code Examples*. Repéré le 14 mai 2025 à https://xinqianzhai.medium.com/cross-validation-with-code-examples-eaabc440f61d

Medium. (2024. 21 septembre). *Why, How, and When to Convert Categorical Variables into Numerical in Machine Learning*. Repéré le 4 mai 2024 à https://medium.com/@bharat.panwar1921piemr/why-how-and-when-to-convert-categorical-variables-into-numerical-in-machine-learning-398e0c5764d8

Pandas. (2024). *pandas.DataFrame.boxplot*. Repéré le 28 avril 2025 à https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.boxplot.html

Université TÉLUQ. (2019). *Module 6 Algorithmes de base en apprentissage machine*. https://sci1016.teluq.ca/teluqDownload.php?file=2018/09/SCI1016-Module6-ApprentissageMachine-Oct2022.pdf

Université TÉLUQ. (2021). *Module 3 Analyse exploratoire des données*. https://sci1016.teluq.ca/teluqDownload.php?file=2018/09/SCI1016_M3_Texte.pdf

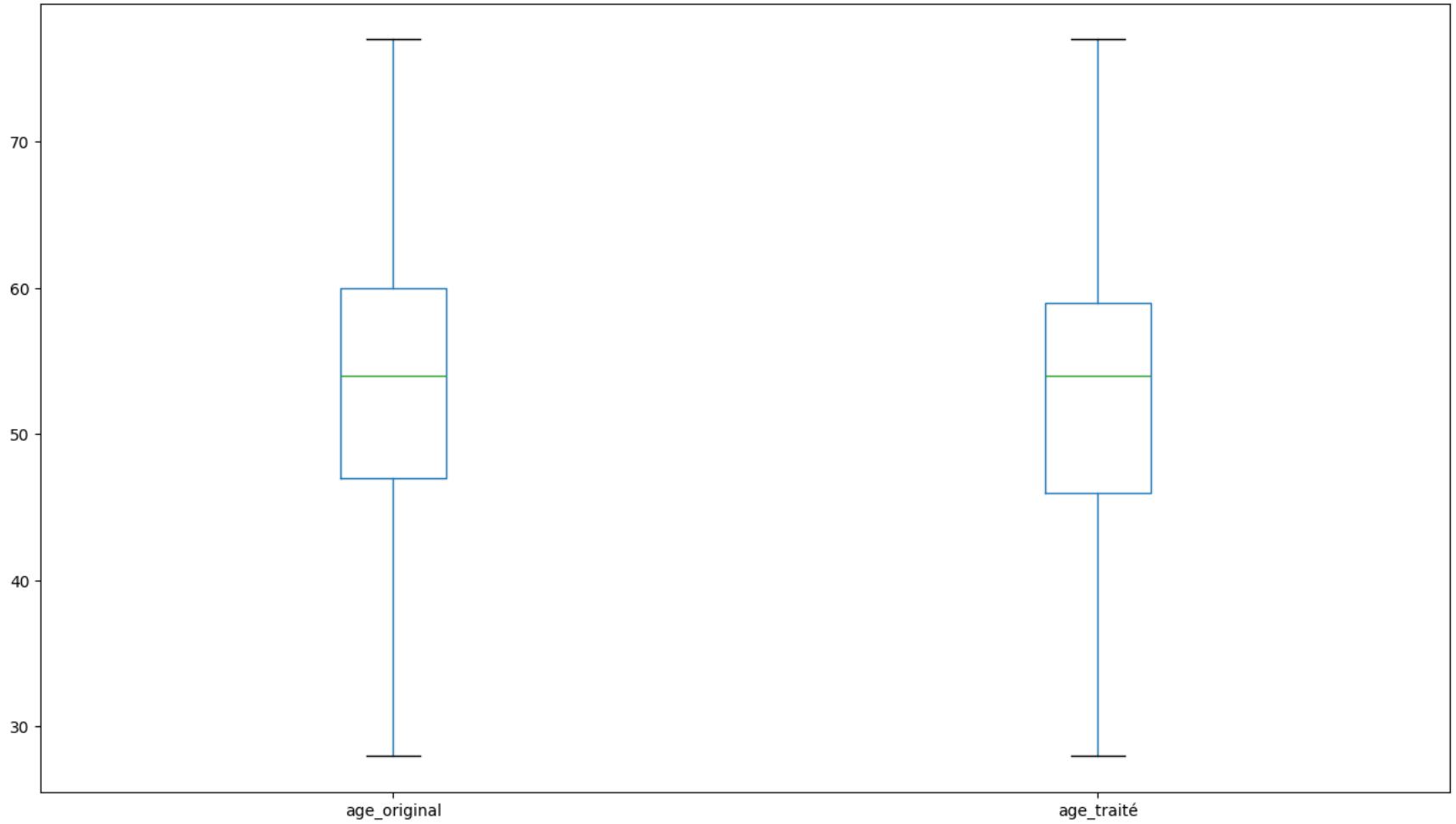
Université TÉLUQ. (2022). *Module 5 Inférence Statistique*. https://sci1016.teluq.ca/teluqDownload.php?file=2020/04/Module5-InferenceStat_Oct2022.pdf

Sparkbyexamples. (2024, 25 septembre). *PySpark Drop Rows with NULL or None Values*. Repéré le 27 avril 2025 à https://sparkbyexamples.com/pyspark/pyspark-drop-rows-with-null-values/#:~:text=In%20order%20to%20remove%20Rows,NULL%20values%20to%20delete%20rows

Stackoverflow. (2017, 27 avril). *Pyspark Dataframe - Map Strings to Numerics.* Repéré le 8 mai 2025 à https://stackoverflow.com/questions/43661660/spark-how-to-write-a-single-csv-file-without-folder

ANNEXE

ANNEXE A



ANNEXE B