Master 2 Intelligence Artificielle, Sciences Des Données Et Systèmes Cyber Physiques (IA2S)

Rapport sur le jeu de données UF Retail HRI Dataset en s'appuyant sur l'article « **Human mobile robot interaction in the retail environment** »

IA, Robotique et CPS

Réalisé par :
Soufian CHABOU – Rahma Imene BENGAMRA

Sous l'encadrement de :
Pr. SAMER Mohammed

2022-2023

Table des Matières

Introduction

Récemment, les recherches scientifiques se sont fortement intéressées à l'utilisation de l'intelligence artificielle pour modéliser avec précision la reconnaissance d'activités humaines. Cette avancée est importante dans de nombreux domaines, tels que la conduite autonome, les robots collaboratifs ou encore la surveillance de la sécurité publique. Les humains ont acquis au fil des millénaires l'instinct de comprendre les intentions et les réactions des autres agents autour d'eux, grâce à quoi ils peuvent prendre des décisions éclairées dans des environnements complexes. Les robots doivent également développer une perception précise du mouvement humain et une planification comportementale pour fonctionner de manière sûre et appropriée, en particulier lorsqu'ils travaillent aux côtés des humains. C'est pourquoi l'interaction homme-robot est de plus en plus importante avec l'avancement de la technologie. Cependant, pour permettre aux robots de travailler en collaboration étroite avec les humains, il est nécessaire d'avoir une surveillance et une prédiction précises du mouvement humain. Dans ce rapport, nous présentons le travail d'un laboratoire de l'Université de Floride qui se concentre sur la navigation sociale entre les humains et les robots dans un environnement commercial de détail et de gros.

Définition du jeu de données

Les ensembles de données qui comprennent les interactions entre les humains et les robots dans un espace commun sont de plus en plus recherchés en raison de leur grand apport à alimenter les recherches en robotique. Cependant, les ensembles de données qui suivent l'HRI en fournissant des informations riches autres que des images et/ou vidéos sont rares. Ce travail consiste à créer un dataset qui se concentre sur la navigation sociale entre les humains et les robots dans un environnement de vente. Le domaine WRT (Wholesale and Retail trade) est choisi comme le focus et le banc d'essai de ce travail en raison de l'augmentation significative de l'utilisation des robots dans ce secteur industriel, qui a été amplifiée par l'émergence de la pandémie, suscitant un besoin crucial pour l'intégration sûre de cette technologie dans l'industrie. De plus, comparé à d'autres environnements courants de déploiement de robots (usines de fabrication, centres de distribution de grande échelle, etc.), les environnements de vente au détail offrent des opportunités de HRI plus directes et fréquentes, ce qui rend les données de mouvement humain plus précieuses.

Description de la collecte des données

La collecte de données a été effectuée avec 8 participants en bonne santé, 5 hommes et 3 femmes. Tous les participants ont été recrutés parmi la population d'étudiants de l'Université de Floride et ont signalé être en bonne santé. Leur âge moyen, leur taille avec des chaussures et leur poids corporel étaient de 19,4 ans avec un écart type de 2 ans, 176,7 cm un écart type de 10,2 cm et 66,0 kg un écart type de 10,1kg.
7 des participants se sont auto-déclarés droitiers, le 8ème a été déclaré ambidextre. Les participants ont volontairement accepté d'être enregistrés et ont été informés que les données collectées dans l'étude seront rendues publiques.

Environnement

L'expérience a été menée dans une installation de recherche WRT orientée vers l'avenir. Ce laboratoire permet aux chercheurs et aux praticiens de mener des études et de pratiquer des protocoles d'interaction avec les méthodes WRT émergentes. Ceux-ci comprennent des systèmes de verrouillage à faible ou sans contact, des dispositifs anti-vol, des écrans de protection, des caméras jour/nuit avec AI en périphérie et des moniteurs de vision publique spéciaux qui offrent une publicité ou un message personnalisé. Il comprend également des murs et des étagères configurables et des unités multifonctionnelles, permettant une capacité de réglage de la disposition physique très flexible

Figure L'environnement

Matériel

Robot

Le robot utilisé dans l'expérience est une plateforme de robot mobile personnalisée constituée d'une base de fret Fetch et d'un manipulateur de robot UR5.

Figure 2 Fret Fetch et UR5

Équipements du robot :

- Lidar 2D intégré
- Webcam Logitech C920
- Centrale inertielle 6D (IMU)
- 2 codeurs de roue

Le participant utilise :

- Xsens (IMU MOCAP) avec 17 capteurs
- Tobii Pro Glasses 2 avec une caméra.

Participant

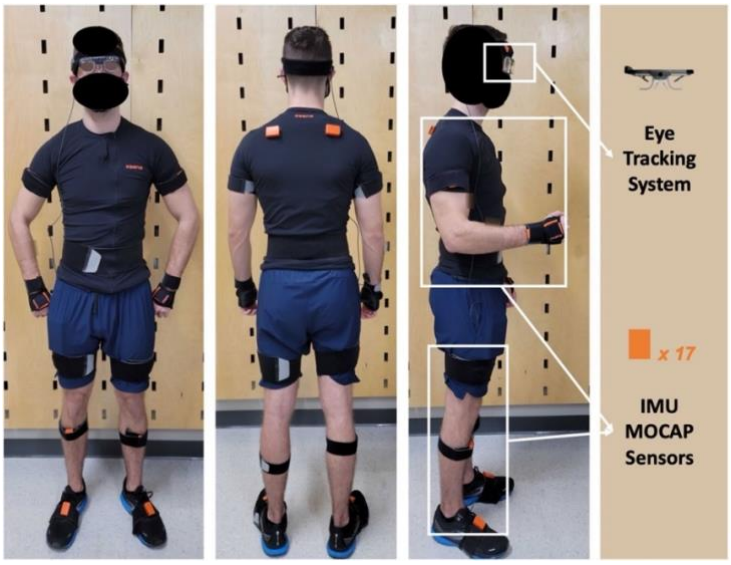


Figure 3 : Photo d'un participant de l'équipe de capteurs IMU.

Les étapes de l'expérimentation

Chaque participant a été bien préparé avant de commencer l'expérience, comme le montre le tableau suivant. Le participant a été introduit à l'environnement et au matériel de l'expérience. Ensuite, une installation et un calibrage des équipements ont été effectués avant de démarrer les tâches à effectuer. Les tâches à réaliser ont été bien expliquées à chaque participant.

Ordre des tâches	Description des tâches	Durée approximative
1	Introduction à l'environnement et aux instruments	5 mins
2	Recueillir des informations démographiques.	2 mins
3	Fixation des capteurs IMU MOCAP.	10 mins
4	Calibrage du système MOCAP.	5 mins
5	Mise en place de l'eye-tracker.	2 mins
6	Calibrage du système de suivi oculaire.	1 min
7	Introduction des tâches.	5 mins
8	Essais de prélèvement et de tri des commandes.	2min × 10 répétions = 20 mins
9	Essais de vérification de l'inventaire.	3min × 4 répétions = 12 mins

La tâche I consistait en une sélection de commandes et un tri. Le participant devait pousser un chariot d'achat et récupérer huit articles différents, un de chaque étagère. Ensuite, le participant est retourné à la caisse automatique pour trier les quatre premiers articles dans un bac et les quatre autres dans un autre bac. Le chariot d'achat pesait 16,78 kg, et pour simuler les conditions réelles, son poids a été ajouté à 45,36 kg et contrôlé pour être le même pour tous les participants. Il y avait deux expériences différentes de la tâche :

1. sélection et tri avec le robot
2. sélection et tri sans le robot.

Chaque expériences avait cinq essais.

La tâche II consistait en une vérification de l'inventaire. Après les tâches de ramassage et de tri des commandes, chaque participant a effectué quatre essais de vérification de l'inventaire. Comme pour la tâche I, il y avait également deux expériences : la vérification de l'inventaire avec le robot et la vérification de l'inventaire sans le robot. Dans chaque essai, le participant recevait une liste de contrôle montrant les articles à vérifier parmi les huit étagères Le participant devait compter les articles de la liste dans l'ordre. Le chariot n'était utilisé à aucun moment tout au long de cette tâche. Le participant a soit expérimenté l'environnement avec le robot en premier, soit sans le robot, et chaque expérience a eu deux répétitions (c'est-à-dire 4 essais de vérification de l'inventaire). Les articles de chaque étagère étaient sélectionnés aléatoirement et les mêmes deux listes de contrôle ont été utilisées dans les deux conditions. Aucune formation n'a été donnée avant cette tâche.

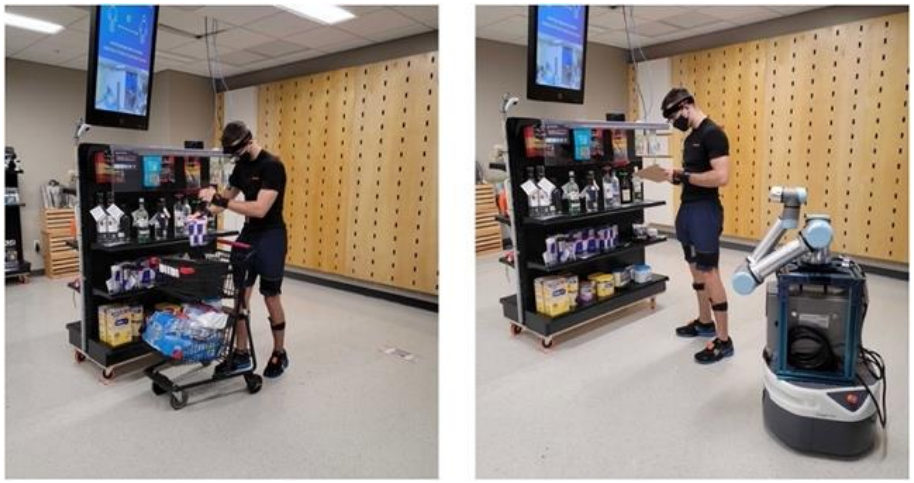
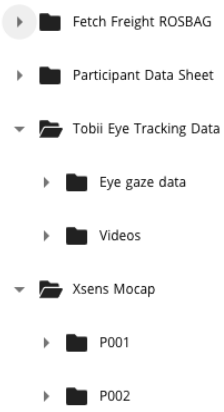


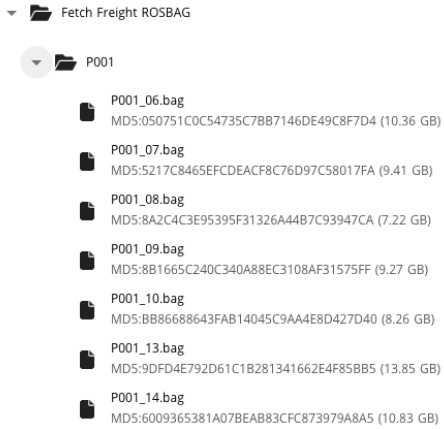
Figure 4 : Photos des expériences mené, celle de gauche représente la sélection et le tri, la photo de droite représente l'expérience du contrôle de l'inventaire .

A la fin de l'expérience plus de 260 minutes de données ont été enregistrées, y compris les données des capteurs de robot, de la capture de l'activité humaine et de la mesure du regard. Afin de fournir une accessibilité gratuite au public, les données ont été téléchargées sur Science Data Bank (https://doi.org/10.11922/sciencedb.01351), un référentiel de données généraliste ouvert développé et maintenu par le Centre de informatique de l'Académie des sciences de Chine.

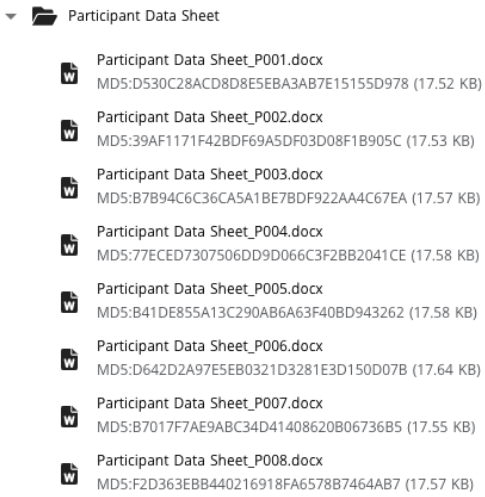
L’architecture de Dataset à télécharger se présente comme suite :



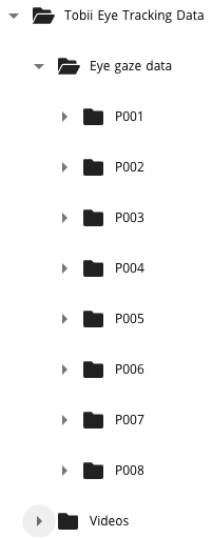
Le dossier Fetch Freight ROSBAG: Contient toutes les données de capteur du robot, y compris les images de la caméra, les nuages de points du Lidar, les mesures d'accélération et de taux angulaire de l'IMU et les états des joints des encodeurs de roues, ont été enregistrées sous forme de données de message sérialisées ROS dans le format rosbag. tant donné que seulement la moitié des essais impliquaient le robot, 56 (40 essais de ramassage et de tri + 16 contrôles d'inventaire) fichiers bag sont disponibles dans le jeu de données.



Le dossier Participant Data Sheet : Les conditions d'essai et la durée des tâches ont été enregistrées dans le "Feuillet de données des participants.docx".



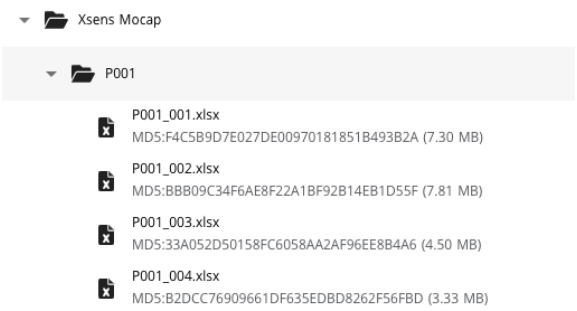
Le dossier Tobii Eye Tracking Data: Les données de suivi des yeux, telles que les directions du regard et les mouvements des yeux, ont été enregistrées dans 112 fichiers Excel (8 participants × 14 essais). Les données collectées ont été exportées avec des horodatages afin qu'elles puissent être analysées plus en détail avec d'autres mesures. De plus, un total de 112 vidéos (.MP4) enregistrées par la caméra de scène intégrée sont également disponibles dans ce jeu de données.



Le dossier Xsens Mocap: Il y a un total de 112 fichiers Excel disponibles dans le dossier de données MOCAP. Quatre essais ont été jugés de mauvaise qualité de données (c'est-à-dire, hors axe) et les fichiers correspondants ont été étiquetés comme "hors axe".
Les mouvements humains suivants ont été fréquemment observés pendant l'expérience :

1. prendre et poser ce qu'il y a sur la liste d'articles.
2. pousser et tirer le chariot d'achat.
3. parcourir les rayons.
4. se pencher et s'accroupir.
5. prendre des articles sur les rayons.
6. mettre des articles dans le bac à tri.
7. écrire sur la liste les articles contrôlés.
8. marcher entre les rayons.
9. compter les articles.
10. éviter le robot si nécessaire.

Avec des enregistrements vidéo et des données de suivi de mouvement portables (tous accessibles), cette information sémantique peut être extraite et étiquetée par le public. Dans chaque dossier de participant, les fichiers d'essai contiennent les informations de temps et les données de position de 23 articulations, y compris le bassin, L5, L3, T12, T8, le cou, la tête, l'épaule droite et gauche, le bras supérieur droit et gauche, le avant-bras droit et gauche, la main droite et gauche, la cuisse supérieure droite et gauche, la jambe inférieure droite et gauche, le pied droit et gauche, le doigt de pied droit et gauche.



Exploitation du jeu de données

Aucune méthode d'apprentissage n'a été appliquée par les chercheurs dans cet article, en revanche ils ont utilisé des modèles pré-entraînés (YOLO V5) pour baliser les vidéos enregistrées par le robot, comme le montre l'image suivante.

Figure 5 : démonstration de l'image vidéo annotée par YOLO V5.

De notre coté nous avons essayé de faire la même chose mais avec les vidéos enregistrer par les Tobbi Eye Tracking ou nous avons réussi a trouvez des résultats similaires de détection des objets comme le montre l'image suivante :

Figure 6 : démonstration de l'image vidéo du Tobbi Eye Tracker annotée par YOLO V5.

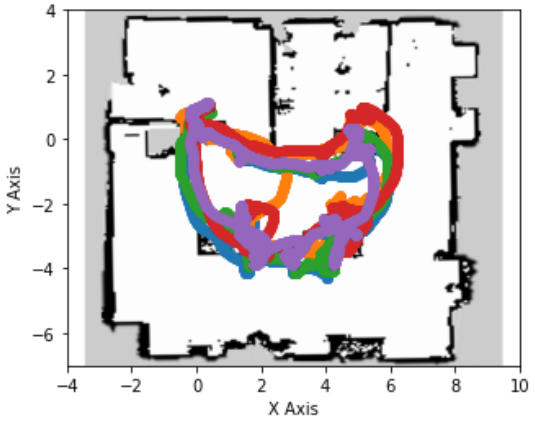
Figure 7 : démonstrations des images vidéo du Tobbi Eye Tracker annotée par YOLO V5.

Nous avons remarqué que la détection des objets a été plus ou moins bonne, bien que le modèle n'ait pas pu détecter l'ensemble des objets dans le WRT. Parmi ces objets, il y a le robot. Ce phénomène est tout à fait normal, car l'ensemble d'entraînement du modèle pré-entraîné ne contient pas de robots. En revanche, il a bien détecté les humains, même s'il ne s'agit que de leurs parties comme les mains par exemple.

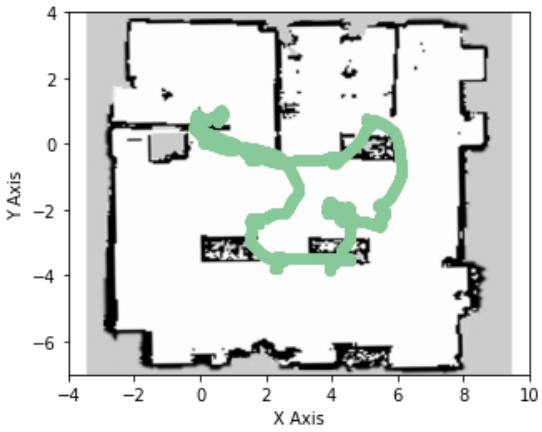
Analyse des trajectoires

Nous avons décidé d'utiliser les données de Xsens pour tracer toutes les trajectoires des participants dans le WRT. Nous avons analysé les résultats pour déterminer s'il y a des différences significatives entre les trajectoires avec et sans un robot dans le WRT. Pour cela, nous avons utilisé les données du premier participant, P1. La première figure représente la trajectoire de P1 dans le WRT lors de l'exécution de la tâche numéro 1, qui consiste en une sélection de commandes et un tri en l'absence du robot lors de la première tentative.

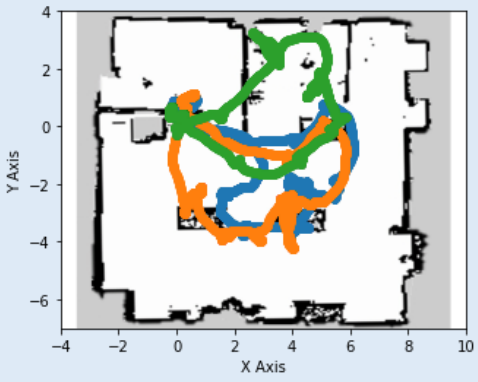
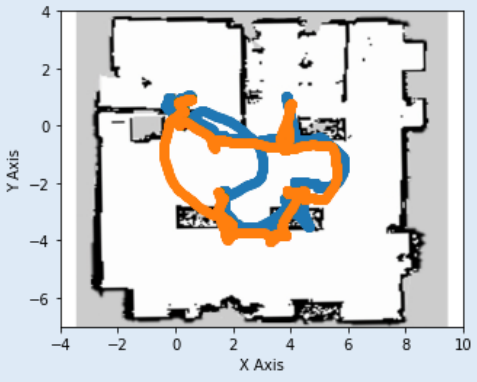
La figure suivante représente la trajectoire de toutes les tentatives de P1 dans la tâche 1.



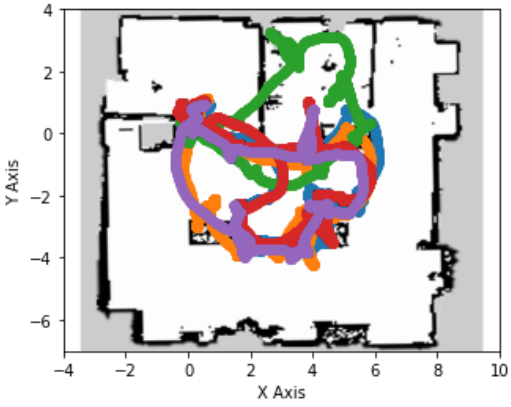
Chaque participant s'est vu attribuer une liste d'articles pour faire une sélection puis un tri (voir la tâche 1). Notez qu'il a dû effectuer cette tâche à 5 reprises, ce qui explique donc les divergences de trajectoire représentées sur la figure. Dans le cas où le robot est actif, nous avons obtenu les résultats qui sont représentés plus bas. La grande divergence des trajectoires est due au fait que même s'ils ont utilisé les mêmes 5 listes, ils ont fait une affectation aléatoire des listes.



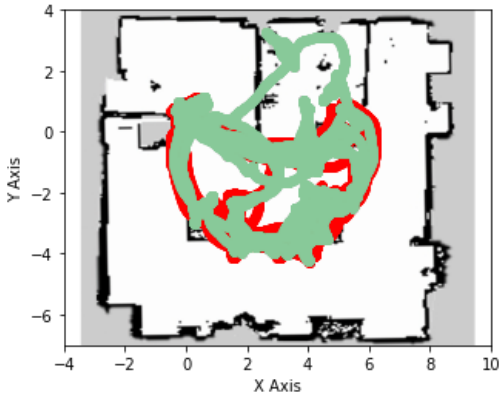
Parmi les 5 tentatives qui ont été réalisées avec la présence du robot, nous avons 3 d'entre elles qui se sont déroulées sans encombre, c'est-à-dire aucun risque de collision entre l'humain et le robot n'était à signaler. Cependant, pour les deux tentatives restantes, il y avait un risque élevé de collision, ce qui a poussé l'équipe à réagir en intervenant sur la trajectoire du robot (avec une manette) afin de dévier la trajectoire du robot, permettant ainsi d'éviter cette éventuelle collision.

Tentative normal	Tentative avec intervention
	

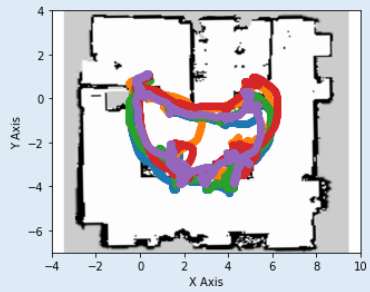
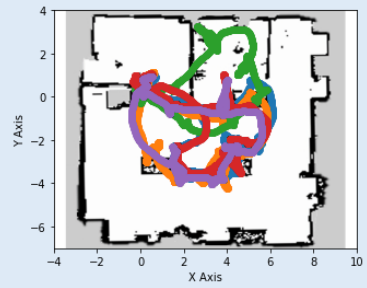
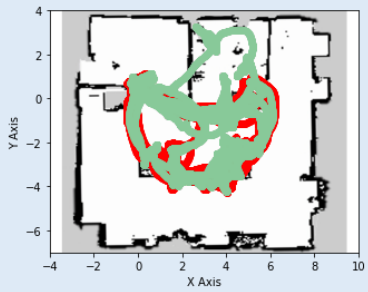
Les 5 trajectoires qui se sont déroulées avec la présence du robot sont représentées dans la figure suivante :



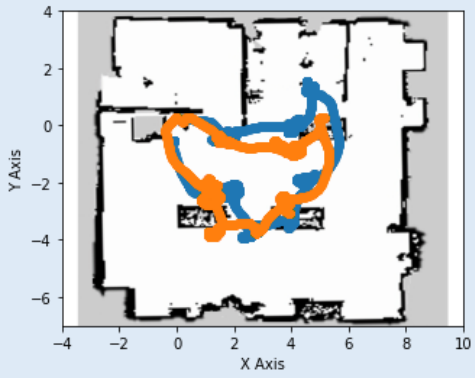
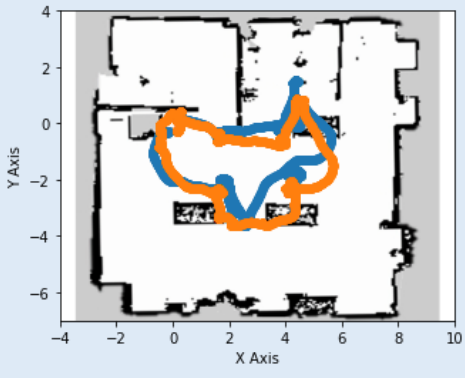
En analysant visuellement la figure, nous pouvons remarquer qu'il y a une différence entre les expériences qui se sont déroulées avec et sans la présence du robot. Pour appuyer notre analyse, nous avons décidé de visualiser les 10 trajectoires dans une seule et même figure, avec les trajectoires avec robot en vert et celles sans robot en rouge. Le résultat est donc représenté ci-dessous.



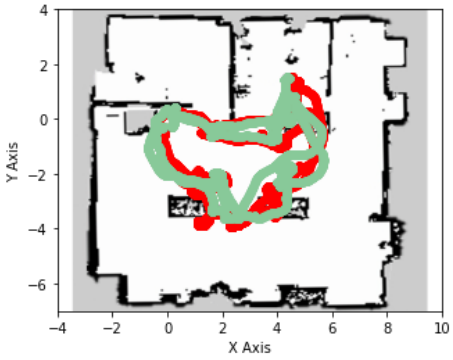
Le tableau suivant comprend le récapitulatif des visualisations que nous avons réalisées.

Sans Robot	Avec Robot	Superposition des trajectoires
		

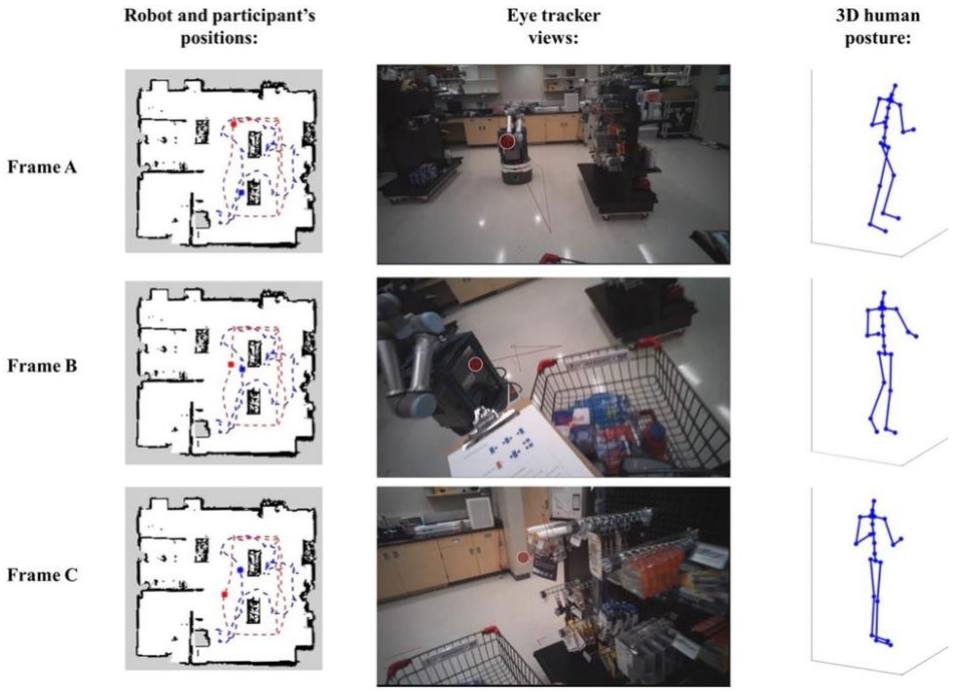
La deuxième tâche est similaire à la première. Une liste a été fournie aux participants pour qu'ils puissent la suivre. La trajectoire du participant sans et en présence du robot est la suivante :

Sans Robot	Avec Robot
	

On peut remarquer qu'il y a une différence entre les trajectoires, ce qui prouve que la présence du robot influence la trajectoire du participant. Sur la figure suivante, nous avons superposé les deux trajectoires. En vert, les trajectoires en présence du robot, et en rouge, la trajectoire sans le robot.



La visualisation suivante a été présentée dans l'article. La première colonne, celle de gauche, représente le mappage des positions des humains et des robots à différents instants au cours de l'expérience. Dans la colonne du centre se trouvent les captures d'écran prises à l'aide de l'eye tracker. Quant à la colonne de droite, elle comprend les figures du modèle humain 3D créé à l'aide des données de MOCAP Xsens. La frame "A" représente donc le premier regard du participant sur le robot. La frame "B" représente le moment où le participant et le robot se trouvent au même niveau (tout en s'évitant pour ne pas causer de collision). La frame C montre le participant prenant une posture lui permettant d'éviter le robot, tout en cherchant un article sur les rayons.



Méthodes d'apprentissage automatique proposées pour ce jeu de données

Chargement des données avec python dans Colab

Lors du téléchargement de tous les dossiers du jeu de données, nous nous sommes aperçus que la taille de ce jeu de données est de 535,86 Go, ce qui est colossal et dépasse la capacité de stockage de nos ordinateurs. C'est la raison pour laquelle nous avons décidé de travailler sur les données Excel qui se trouvent dans le dossier Xsens MOCAP, qui contient toutes les informations relatives aux capteurs de toutes les expériences des 8 participants. Chaque participant a son propre dossier qui contient les données des 14 expériences qu'il a effectuées. Nous avons utilisé le langage Python sous Google Colab car il offre de nombreux outils et fonctionnalités qui facilitent la programmation ainsi que la visualisation des données en ayant recours à des bibliothèques telles que Pandas et Numpy pour le

traitement des données dans des tableaux ou des listes, la bibliothèque Scikit-Learn qui propose plusieurs algorithmes d'apprentissage automatique, ainsi que Matplotlib pour la visualisation des données et plein d'autres fonctionnalités

Architecture de modèle

Après une longue analyse des données de capteurs Xsens, nous avons décidé de construire deux modèle de Machine Learning qui vont prédire le type de tâche qu'un participant est en train d'effectuer (order picking and sorting ou bien inventory checking).

Modèle 1 (prédiction de tâche) :

Afin de réduire la complexité de notre modèle, nous avons décidé d'utiliser uniquement les données des participants en l'absence du robot. Ainsi, nous avons pris comme données d'entraînement celles des participants 1 et 2, et avons testé notre modèle avec les données des autres participants. Les classes que nous avons utilisées étaient au nombre de 2 : la première, que nous avons étiquetée "1", correspond à la tâche 1 (order picking and sorting), et la deuxième, étiquetée "2", correspond à la tâche 2 (inventory checking).

Modèle 2 (prédiction de tâche sans et avec robot) :

Les mêmes données que pour le premier modèle ont été utilisées. Cependant, cette fois-ci, nous avons non pas 2, mais 4 classes (1, 2, 3 et 4) qui correspondent respectivement à order picking and sorting sans le robot, order picking and sorting avec le robot, inventory checking sans le robot et inventory checking avec le robot.

Pour construire notre modèle, nous avons suivi les étapes suivantes énumérées ci-dessous

Étape 1 : Construction et Normalisation des données d'entrainement

Dans cette première partie, nous avons collecté les données pertinentes pour l'entraînement de notre modèle. Durant le prétraitement des données, nous avons constaté que la normalisation des données prend beaucoup de temps en raison de la grande taille du jeu de données. C'est pourquoi nous avons décidé d'enregistrer les données après ce traitement dans de nouveaux fichiers pour les utiliser dans les étapes suivantes, afin d'optimiser le temps d'exécution futur.

```
import pandas as pd
import numpy as np

df_P1_1 = pd.read_excel("P001_001.xlsx")
df_P1_2 = pd.read_excel("P001_002.xlsx")
df_P1_3 = pd.read_excel("P001_003.xlsx")
df_P1_4 = pd.read_excel("P001_004.xlsx")
df_P1_5 = pd.read_excel("P001_005.xlsx")
df_P1_6 = pd.read_excel("P001_006.xlsx")
df_P1_7 = pd.read_excel("P001_007.xlsx")
df_P1_8 = pd.read_excel("P001_008_offaxis.xlsx")
df_P1_9 = pd.read_excel("P001_009.xlsx")
df_P1_10 = pd.read_excel("P001_010.xlsx")
df_P1_11 = pd.read_excel("P001_011.xlsx")
df_P1_12 = pd.read_excel("P001_012.xlsx")
```

```
df_P1_13 = pd.read_excel("P001_013.xlsx")
df_P1_14 = pd.read_excel("P001_014.xlsx")
df_P2_13 = pd.read_excel("P002_013.xlsx")
df_P2_14 = pd.read_excel("P002_014.xlsx")

# Normalization
def Normalization(df):

  List = list(df)
  for i in List:
    max = df[i].max()
    min = df[i].min()
    for j in range(len(df[i])):
      df[i].iloc[j] = (df[i].iloc[j]-min)/(max-min)
  return df


df_P1_1 = Normalization(df_P1_1)
df_P1_2 = Normalization(df_P1_2)
df_P1_3 = Normalization(df_P1_3)
df_P1_4 = Normalization(df_P1_4)
df_P1_5 = Normalization(df_P1_5)
df_P1_6 = Normalization(df_P1_6)
df_P1_7 = Normalization(df_P1_7)
df_P1_8 = Normalization(df_P1_8)
df_P1_9 = Normalization(df_P1_9)
df_P1_10 = Normalization(df_P1_10)
df_P1_11 = Normalization(df_P1_11)
df_P1_12 = Normalization(df_P1_12)
df_P1_13 = Normalization(df_P1_13)
df_P1_14 = Normalization(df_P1_14)
df_P2_13 = Normalization(df_P2_13)
df_P2_14 = Normalization(df_P2_14)

frame = [df_P1_1, df_P1_2, df_P1_3, df_P1_4, df_P1_5 ]
peckinge_dataset = pd.concat(frame)
frame2 = [df_P1_11, df_P1_12, df_P2_13, df_P2_14]
inventory_dataset = pd.concat(frame2)

del inventory_dataset['Timestamp']
del peckinge_dataset['Timestamp']

inventory_dataset.to_csv("inventory_dataset.csv", index=False)
peckinge_dataset.to_csv("peckinge_dataset.csv", index=False)
```

Étape 2 : Prétraitement des données

Dans cette étape, nous avons pris les données de l'étape 1 et avons préparé l'ensemble d'entraînement en attribuant la classe 1 à la tâche 1 et 2 à la tâche 2. Ensuite, nous avons divisé l'ensemble des données en sous ensemble d'entraînement et sous ensemble de test.

```
def import_data(dataframe, classe):
  train_data = np.empty((0,69))
  train_classe = np.empty((0,1))
  for i in range(len(dataframe)):
    train_data=np.concatenate((train_data,dataframe.iloc[[i]]))
    train_classe=np.concatenate((train_classe,[[classe]]))
  return train_data, train_classe

x_pecking_data, y_pecking_data = import_data(peckinge_dataset,1)
x_inventory_data, y_inventory_data = import_data(inventory_dataset,2)

X = np.concatenate((x_pecking_data,x_inventory_data), axis = 0)
y = np.concatenate((y_pecking_data,y_inventory_data), axis = 0)


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, train_size=0.7, random_state=1)
```

Étape 3 : Entrainement des modelés de classification

Nous avons entraîné plusieurs modèles de classification qui sont : Naïve Bayes, KNN, K-means et SVM.

L'algorithme Naïve Bayes est un algorithme d'apprentissage automatique utilisé dans la classification de données. Il est basé sur le théorème de Bayes, qui est une formule pour calculer la probabilité d'un événement en fonction de sa probabilité conditionnelle et de la probabilité des événements qui peuvent en découler. En utilisant cet algorithme, les données sont classées en fonction de la probabilité que chaque donnée d'entrée corresponde à une classe prédite.

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import (precision_score, recall_score, f1_score,
accuracy_score, confusion_matrix)
#fit the model with naive Bayes algorithm
model_nb = GaussianNB()
model_nb.fit(X_train, y_train)
#Prédire les données de test
y_pred_nb = model_nb.predict(X_test)
```

L'algorithme KNN (k-nearest neighbors) est un algorithme d'apprentissage automatique qui est utilisé pour la classification et la régression. Il s'agit d'un algorithme d'apprentissage supervisé qui utilise une

approche de voisinage pour classer ou prédire de nouvelles données en fonction de leur proximité avec les données existantes.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn import model_selection
model_knn = KNeighborsClassifier(10)
# entrainer le modéle en utilisant X_train
model_knn.fit(X_train, y_train)
#10_fold Cross validation
succes = model_selection.cross_val_score(model_knn, X, y,
cv=10,scoring='accuracy')
print('succes : ', succes)
#evaluate the model
y_pred_knn = model_knn.predict(X_test)
```

L'algorithme K-means est un algorithme de clustering non supervisé, ce qui signifie qu'il ne nécessite pas de données étiquetées pour regrouper les données en clusters. Dans l'apprentissage non supervisé, l'algorithme essaie de trouver des structures ou des modèles intrinsèques dans les données sans l'aide de la supervision d'un expert.

```
from sklearn.cluster import KMeans
from sklearn import model_selection
model_km = KMeans(2)
#Entrainer le modéle K-means en utilisant X_train
model_km.fit(X_train, y_train)
#Cross validation 10_fold
succes = model_selection.cross_val_score(model_km, X, y,
cv=10,scoring='accuracy')
print('Résultat : ', succes)
#Prédire les données de test
y_pred_km = model_km.predict(X_test)
```

L'algorithme SVM (Support Vector Machine) est un algorithme d'apprentissage automatique utilisé pour la classification et la régression. Il est principalement utilisé pour la classification de données linéairement séparables ou non linéairement séparables.

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clmodel_svm = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clmodel_svm.fit(X_train, y_train)

#Predict the response for test dataset
y_pred_svm = clmodel_svm.predict(X_test)
```

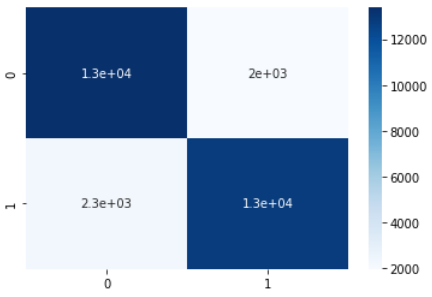
Étape 4 : Comparaison et analyse de la performance des modelés

```
import matplotlib.pyplot as plt
import seaborn as sns

def resultat(y_test, y_pred):
  #Matrice de confusion
  confusion_m= confusion_matrix(y_test, y_pred)
  sns.heatmap(confusion_m,  cmap='Blues', annot=True)
  print("Matrice de confusion \n", confusion_m)
  acc = accuracy_score(y_test,y_pred)
  print("Accuracy : ",acc)
  #Precision
  precision = precision_score(y_test,y_pred,
pos_label='positive',average='micro')
  print("precision : ",precision)
  #Rappel
  rappel =
recall_score(y_test,y_pred,pos_label='positive',average='micro')
  print("Rappel : ",rappel)
  #F-mesure
  f_mesure=f1_score(y_test, y_pred, labels=None, pos_label=1,
average='micro',sample_weight=None,zero_division='warn')
  print("f_mesure : ",f_mesure)
```

Algorithme Naive Bayes :

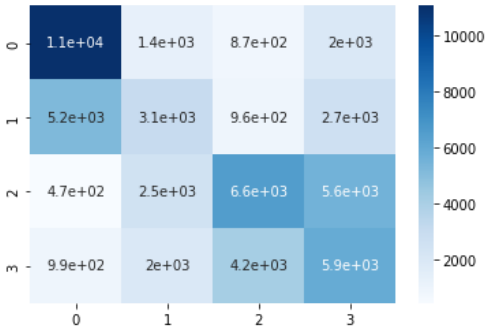
```
resultat(y_test, y_pred_nb)
```

Modèle 1




```
Matrice de confusion
 [[13374  1969]
 [ 2322 12799]]
Accuracy :  0.8591452205882353
precision :  0.8591452205882353
Rappel :  0.8591452205882353
f_mesure :  0.8591452205882353
```

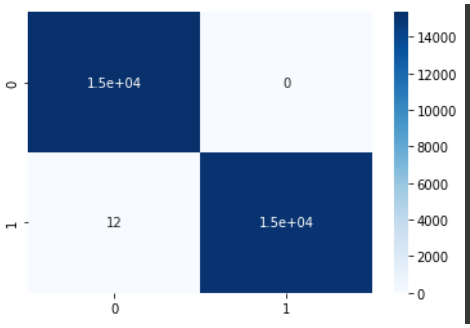
Modèle 2




```
Matrice de confusion
 [[11053  1368   870  2016]
 [ 5246  3100   956  2689]
 [  471  2544  6582  5584]
 [  994  2033  4172  5909]]
Accuracy :  0.4793207044812636
precision :  0.4793207044812636
Rappel :  0.4793207044812636
f_mesure :  0.4793207044812636
```

Algorithme KNN :

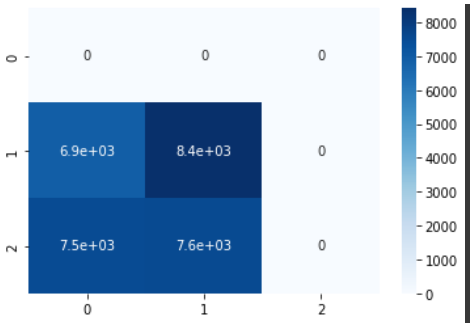
```
resultat(y_test, y_pred_knn)
```




```
Matrice de confusion
 [[15343     0]
 [   12 15109]]
Accuracy :  0.9996060924369747
precision :  0.9996060924369747
Rappel :  0.9996060924369747
f_mesure :  0.9996060924369747
```

Algorithme K-means :

```
resultat(y_test, y_pred_km)
```

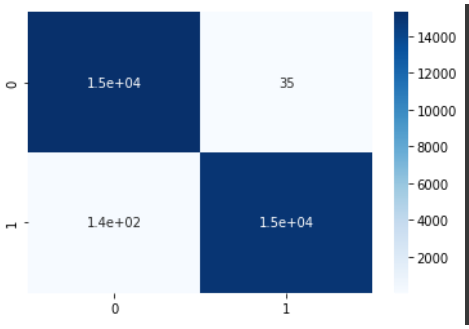



```
Matrice de confusion
 [[   0    0    0]
 [6928 8415    0]
 [7545 7576    0]]
Accuracy :  0.27622767857142855
precision :  0.27622767857142855
Rappel :  0.27622767857142855
```

Algorithme SVM :

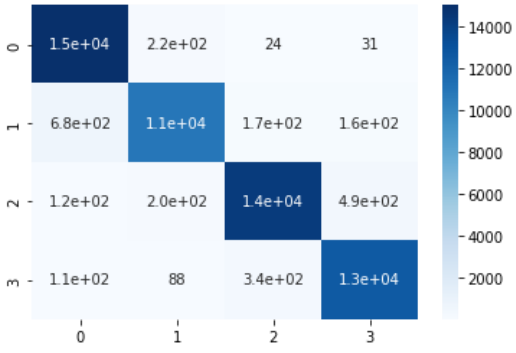
```
resultat(y_test, y_pred_svn)
```

Modèle 1




```
Matrice de confusion
 [[15308    35]
 [  141 14980]]
Accuracy :  0.9942226890756303
precision :  0.9942226890756303
Rappel :  0.9942226890756303
f_mesure :  0.9942226890756303
```

Modèle 2




```
Matrice de confusion
 [[15031   221    24    31]
 [  679 10978   172   162]
 [  120   205 14366   490]
 [  112    88   335 12573]]
Accuracy :  0.9525248709230576
precision :  0.9525248709230576
Rappel :  0.9525248709230576
f_mesure :  0.9525248709230576
```

Nous remarquons que les résultats de l'apprentissage supervisé ont été très performants pour le premier modèle et ont correctement prédit les classes des données de test. En revanche, la méthode d'apprentissage non supervisé K-Means n'a pas réussi à classifier les données de manière satisfaisante.

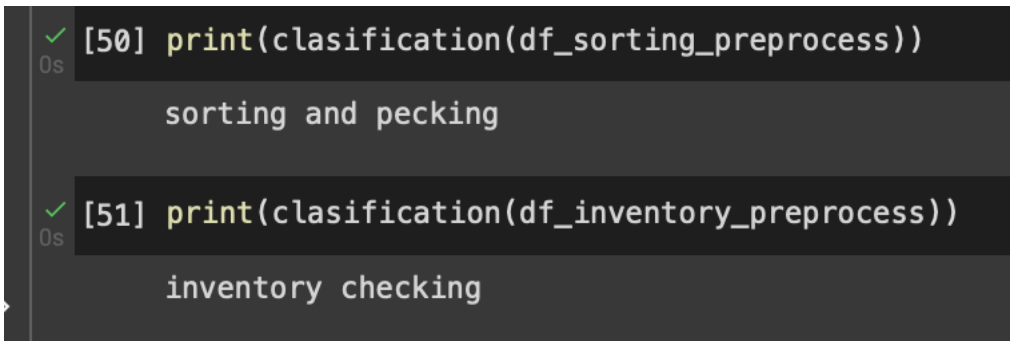
Pour le modèle 2, la méthode d'apprentissage supervisé SVM a donné des résultats très intéressants. Cependant, lors des tests avec de nouvelles données de participants, nous avons remarqué qu'il y avait des confusions pour déterminer si le robot était présent ou non.

Étape 5 : Test du modèle

Suite à la construction du modèle, nous avons procédé à une phase de test en utilisant les données des participants 6, 7 et 8. Cette démarche avait pour objectif de vérifier que notre modèle est capable de fournir des classifications précises.

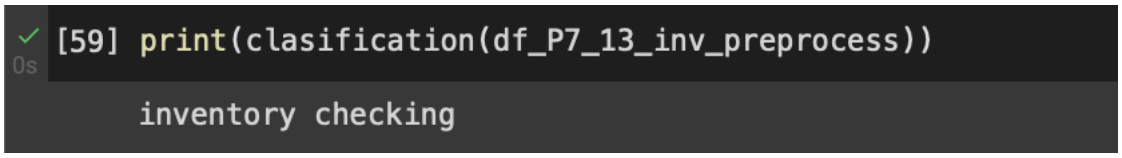
```
def preprosess_data(dataframe):
  train_data = np.empty((0,69))
  for i in range(len(dataframe)):
    train_data=np.concatenate((train_data,dataframe.iloc[[i]]))
  return train_data

def clasification(X):
  y_pred_nb = model_nb.predict(X)
  # Convert the predictions to a 1D array of class indices
  predictions = y_pred_nb.astype('int')
  # Get the class with the majority of occurrences
  majority_class = np.bincount(predictions).argmax()
  if(majority_class == 1):
    return "sorting and pecking"
  elif(majority_class == 2):
    return "inventory checking"


# Normalization
def Normalization(df):

  List = list(df)
  for i in List:
    max = df[i].max()
    min = df[i].min()
    for j in range(len(df[i])):
      df[i].iloc[j] = (df[i].iloc[j]-min)/(max-min)
  return df

df_sorting = pd.read_excel("P006_006.xlsx")
df_inventory = pd.read_excel("P006_014.xlsx")
df_sorting = Normalization(df_sorting)

df_inventory = Normalization(df_inventory)

del df_sorting['Timestamp']
del df_inventory['Timestamp']

df_sorting_preprocess = preprosess_data(df_sorting)
df_inventory_preprocess = preprosess_data(df_inventory)
```

Nous avons débuté en testant les données du participant 6 en utilisant les données de la tâche "order sorting and pecking" ainsi que celles de "inventory checking". Nous avons prétraité les données pour les tester. Suite à ce test, le modèle a été capable de classifier les données et de déterminer la tâche correspondante avec précision.

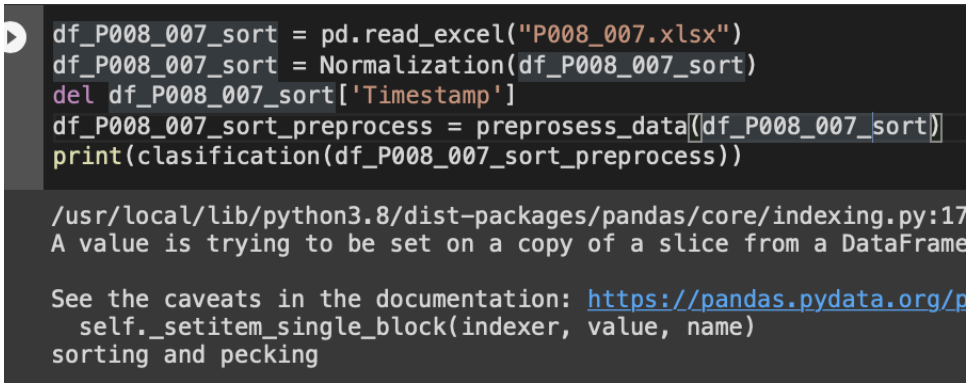


Nous avons ensuite utilisé les données du participant 7 pour tester notre modèle. Nous avons extrait les données de la tâche "order sorting and pecking" ainsi que les données de "inventory checking" et les avons pré-traitées pour la classification. Après le test, notre modèle a pu classifier les données avec précision et déterminer à quelle tâche elles correspondaient.

```
df_P7_13_inv = pd.read_excel("P007_013.xlsx")
df_P7_13_inv = Normalization(df_P7_13_inv)
del df_P7_13_inv['Timestamp']
df_P7_13_inv_preprocess = preprosess_data(df_P7_13_inv)
```




```
df_P008_007_sort = pd.read_excel("P008_007.xlsx")
df_P008_007_sort = Normalization(df_P008_007_sort)
del df_P008_007_sort['Timestamp']
df_P008_007_sort_preprocess = preprosess_data(df_P008_007_sort)
print(clasification(df_P008_007_sort_preprocess))
```



Nous avons effectué la même procédure de test pour le modèle SVM et les résultats ont été identiques.

```
def clasification_clf(X):
```

```
  y_pred_nb = model_svn.predict(X)
  # Convert the predictions to a 1D array of class indices
  predictions = y_pred_nb.astype('int')
  # Get the class with the majority of occurrences
  majority_class = np.bincount(predictions).argmax()
  if(majority_class == 1):
    return "sorting and pecking"
  elif(majority_class == 2):
    return "inventory checking"
```

```
[62] print(clasification_clf(df_P008_007_sort_preprocess))
     sorting and pecking

[63] print(clasification_clf(df_P7_13_inv_preprocess))
     inventory checking

     print(clasification_clf(df_inventory_preprocess))
     inventory checking

[65] print(clasification_clf(df_sorting_preprocess))
     sorting and pecking
```

Modèle 2 :

```
def clasification_svn(X):
  y_pred_nb = model_svn.predict(X)
  # Convert the predictions to a 1D array of class indices
  predictions = y_pred_nb.astype('int')
  # Get the class with the majority of occurrences
  majority_class = np.bincount(predictions).argmax()
  if(majority_class == 1):
    return "sorting and pecking"
  elif(majority_class == 2):
    return "sorting and pecking with robot"
  elif(majority_class == 3):
    return "inventory checking"
  elif(majority_class == 4):
    return "inventory checking with robot"
```

```
print(clasification_svn(df_sorting))
print(clasification_svn(df_inventory))
print(clasification_svn(df_P7_13_inv))
print(clasification_svn(df_P008_007_sort))
print(clasification_svn(df_P56sr))
print(clasification_svn(df_P57sr))
print(clasification_svn(df_P514ir))
```

```
sorting and pecking
inventory checking with robot
inventory checking
sorting and pecking
inventory checking with robot
sorting and pecking with robot
sorting and pecking
```

Conclusion

Dans le cadre de ce travail, nous avons présenté une étude du jeu de données UF Retail HRI Dataset, basée sur l'article "Human mobile robot interaction in the retail environment". Nous avons analysé ce jeu de données dans le but de trouver des applications pour l'apprentissage automatique. Nous avons choisi de nous concentrer sur les données des capteurs Xsens pour créer des modèles capables de prédire avec précision les tâches effectuées par un humain dans un environnement WRT. Nous avons utilisé les méthodes d'apprentissage supervisé (Naïve Bayes, SVM et KNN) ainsi que l'apprentissage non supervisé (K-means) pour classifier ces données et analyser les résultats obtenus. Enfin, nous avons utilisé les données d'autres participants pour tester la capacité de nos modèles à déterminer les tâches effectuées par l'être humain.