

Amélioration des performances de tir

HAKKI Mohamed Anass

SOMMAIRE

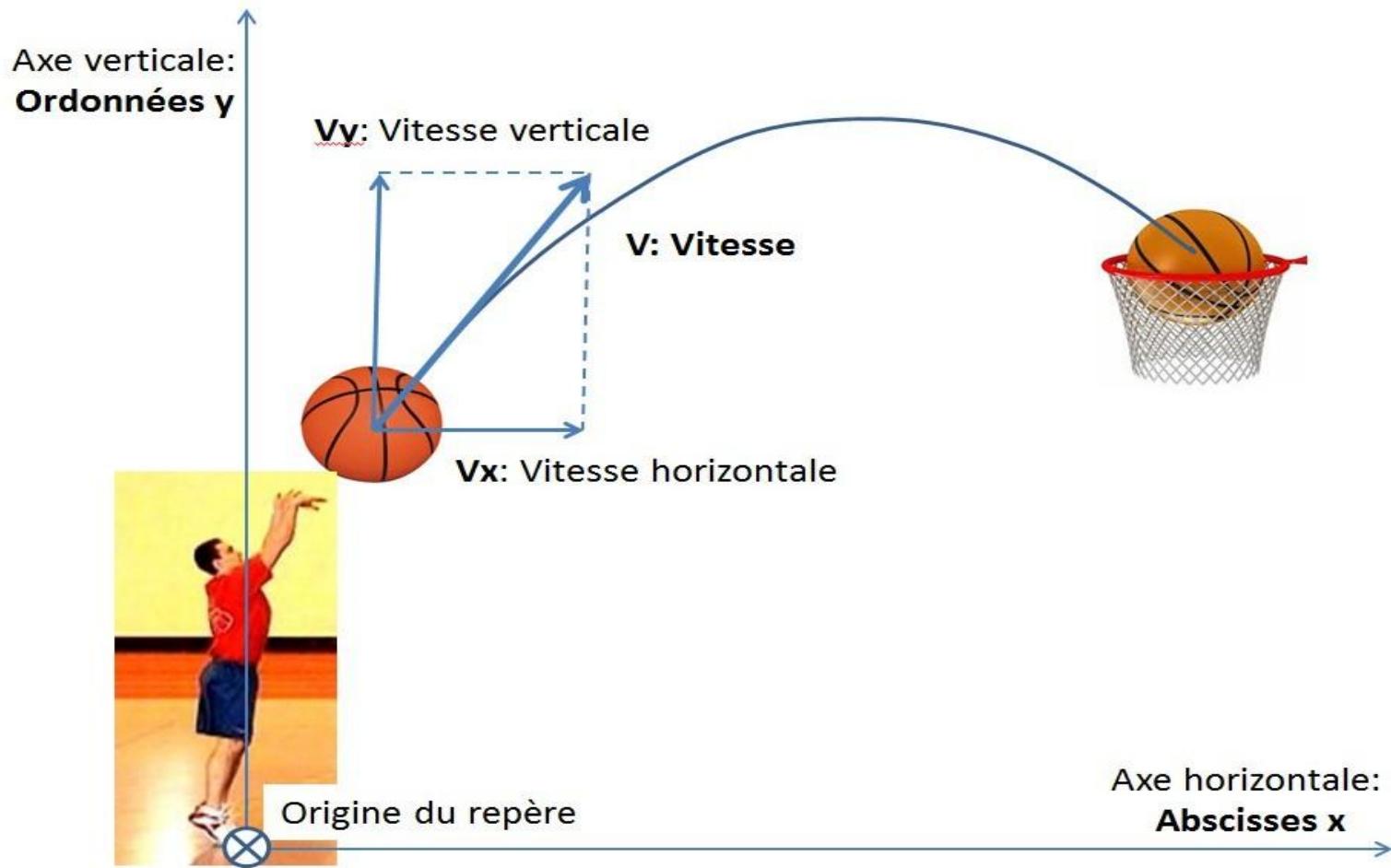
- I. la chute parabolique et ses équations.
- II. Capteur GY-521 et collecte des données.
- III. Préparation des Données pour la Classification.
- IV. Analyse du fonctionnement de la SVM et classement des tirs.
 - i. La machine à vecteurs de support (SVM)
 - ii. Noyau polynomial et noyau linéaire
- IV. Étude comparative des performances .

L'étude théorique et l'expérience pratique sont synchronisées



Chute parabolique

(dans un champ de pesanteur uniforme)



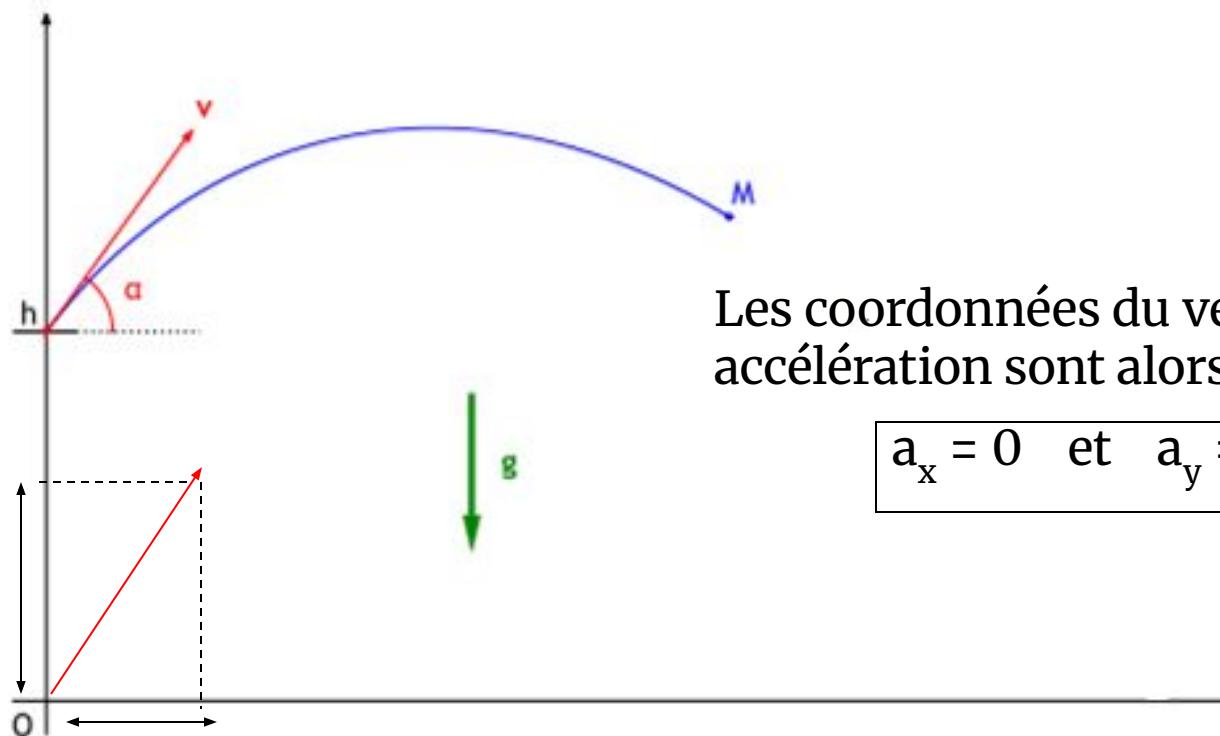
Etablissement des équations horaires

Identification des conditions initiales

$$\Sigma F = ma$$

$$g = a$$

$$v_{0y} = V_0 \sin \alpha$$



Les coordonnées du vecteur accélération sont alors:

$$a_x = 0 \quad \text{et} \quad a_y = -g$$

Les coordonnées du vecteur vitesse à la date $t = 0$ sont:

$$v_{0x} = V_0 \cos \alpha$$

$$v_{0y} = V_0 \sin \alpha$$

Vecteur Vitesse :

$$\vec{V} = \begin{pmatrix} V_x = v_0 \cos \alpha \\ V_y = -gt + v_0 \sin \alpha \end{pmatrix}$$

vecteur
position:

$$\vec{OG} = \begin{pmatrix} x = v_0 \cos \alpha \times t \\ y = -\frac{1}{2}gt^2 + v_0 \sin \alpha \times t + h \end{pmatrix}$$

Nous avons les équations
horizontales: $a_x = 0$
 $a_y = -g$

Pour la vitesse: $v_x = v_0 \cos \alpha$
 $v_y = -gt + v_0 \sin \alpha$

Pour la position: $x(t) = v_0 \cos \alpha \times t$

$$y(t) = -\frac{1}{2}gt^2 + v_0 \sin \alpha \times t + h$$

Etablissement de l'équation de la trajectoire $y = f(x)$:

À partir de la composante de x , on extrait t :

$$t = \frac{x}{v_0 \times \cos \alpha}$$

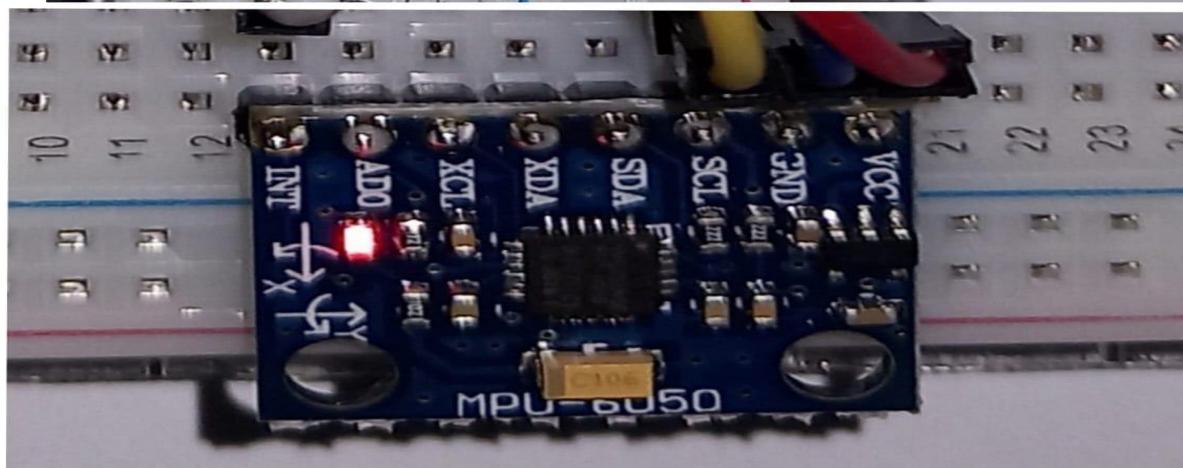
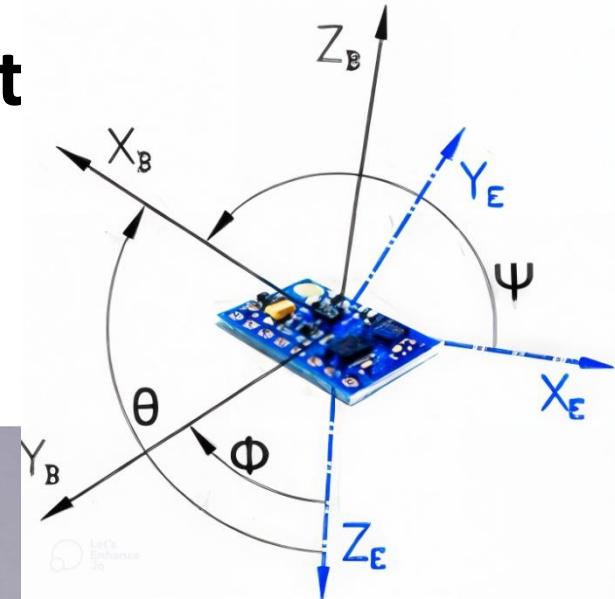
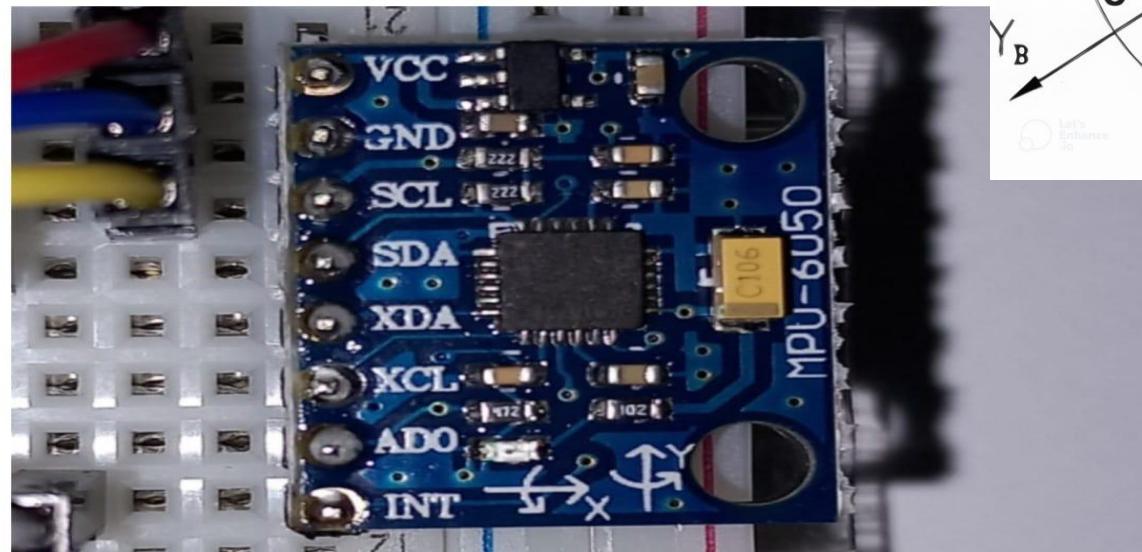
On simplifie et on obtient l'équation d'une parabole:

$$y = -\frac{g \times x^2}{2 \times v_0^2 \times \cos^2 \alpha} + x \times \tan \alpha + h$$

Capteur de mouvement

GY-521 ~MPU 6050

- Le capteur GY-521, également appelé MPU-6050, mesure le mouvement grâce à un accéléromètre triaxial et un gyroscope triaxial intégrés.

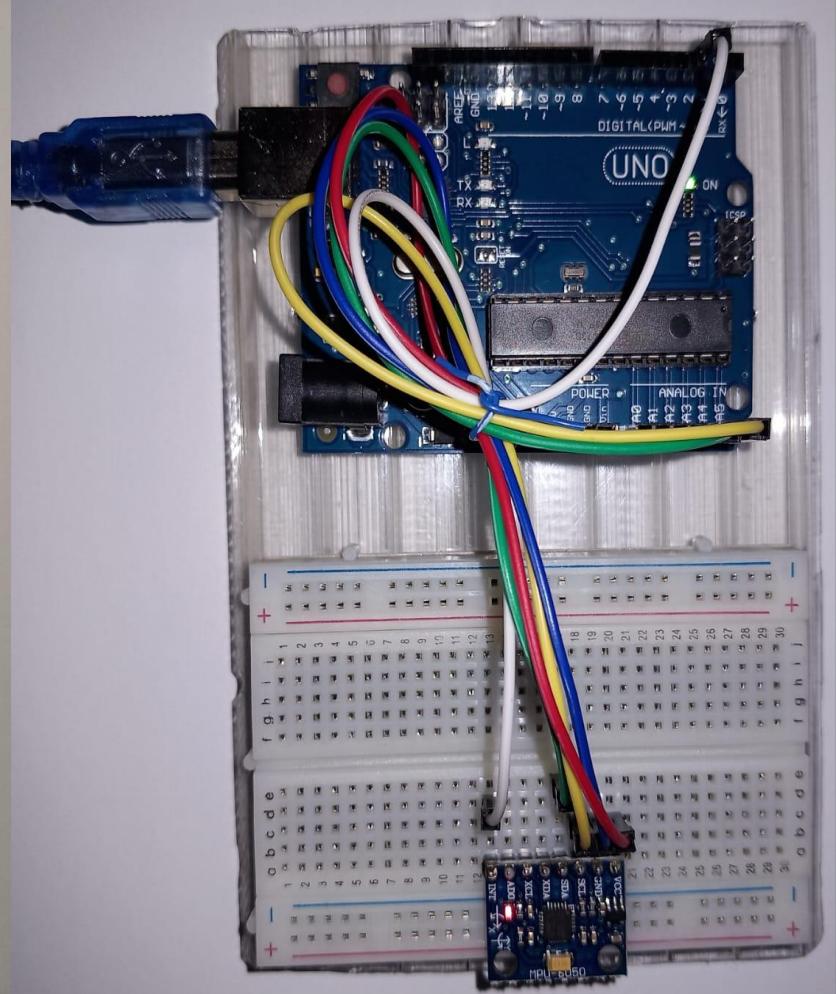


- L'accéléromètre détecte les accélérations linéaires sur les axes x, y et z, généralement mesurées en "g" (gravité standard).
- Le gyroscope mesure les vitesses angulaires de rotation autour des axes x, y et z, souvent exprimées en degrés par seconde ($^{\circ}/s$) ou radians par seconde (rad/s).
- En programmation avec Arduino et le capteur MPU-6050, les données d'accélération peuvent être obtenues par les variables AcX, AcY et AcZ pour les axes x, y et z respectivement.
- De même, les vitesses angulaires du gyroscope peuvent être obtenues par les variables GyX, GyY et GyZ pour les axes x, y et z respectivement.

- Tirs1: 30 tirs **corrects** (professionnels)
- Tirs2: 30 tirs **incorrects** (débutants)
- Transfert vers des fichiers texte (.txt) via Arduino



Acc_Y	Acc_Z	Gyro_X	Gyro_Y	Gyro_Z
-1252	5392	395	357	-686
-1328	5400	664	561	-605



Préparation des Données

Apprentissage automatique (Machine Learning)



- **Chargement des données**
- **Nettoyage des données**
- **Extraction des caractéristiques**
- **Mise à l'échelle des caractéristiques**
- **Division des données**

Data Loading
Data Cleaning
Feature Extraction
Feature Scaling
Data Splitting

CLASSIFICATION



une branche de l'intelligence artificielle qui permet aux ordinateurs d'apprendre à partir de données

Chargement des données (Data Loading)

- **Lecture de fichiers texte :**
- la fonction ‘open’ pour lire les fichiers tirs1.txt et tirs2.txt
- **Parsing des données :**
- Extraction et conversion des valeurs des fichiers texte en entiers.
- **Combinaison des données :**
- Fusion des données des tirs corrects et incorrects en une seule structure de données.
- **Conversion en DataFrame :**
- Utilisation de la bibliothèque pandas pour convertir les données en DataFrame

Nettoyage des données

(Data Cleaning)

- **Correction des erreurs :**
- Modification de la dernière colonne de incorrect_data pour définir les valeurs de Label à 0.

- **Suppression des valeurs manquantes :**
- ‘dropna’ de ‘pandas’ pour supprimer les lignes contenant des valeurs manquantes.

Extraction des caractéristiques

(Feature Extraction)

- + Sélection des caractéristiques pertinentes
- + Calcul des caractéristiques statistiques

Formule de la Variance :

$$\sigma^2 = \frac{\sum_{i=1}^n (v_i - \bar{v})^2}{n - 1}$$

- σ^2 : Variance
- n : Nombre d'échantillons dans le jeu de données
- v_i : Valeur individuelle de l'échantillon
- \bar{v} : Moyenne des valeurs des échantillons

Mise à l'échelle des caractéristiques (Feature Scaling)

- Normalisation
- Égalisation des unités

La normalisation est une technique de mise à l'échelle qui ajuste les valeurs des caractéristiques pour qu'elles se situent toutes dans une plage donnée, typiquement [0, 1].

Formule de Normalisation :

$$x_{\text{new}} = \frac{x - \min(X)}{\max(X) - \min(X)}$$

- x_{new} : Valeur normalisée
- x : Valeur originale
- $\min(X)$: Valeur minimale dans l'ensemble des données X
- $\max(X)$: Valeur maximale dans l'ensemble des données X

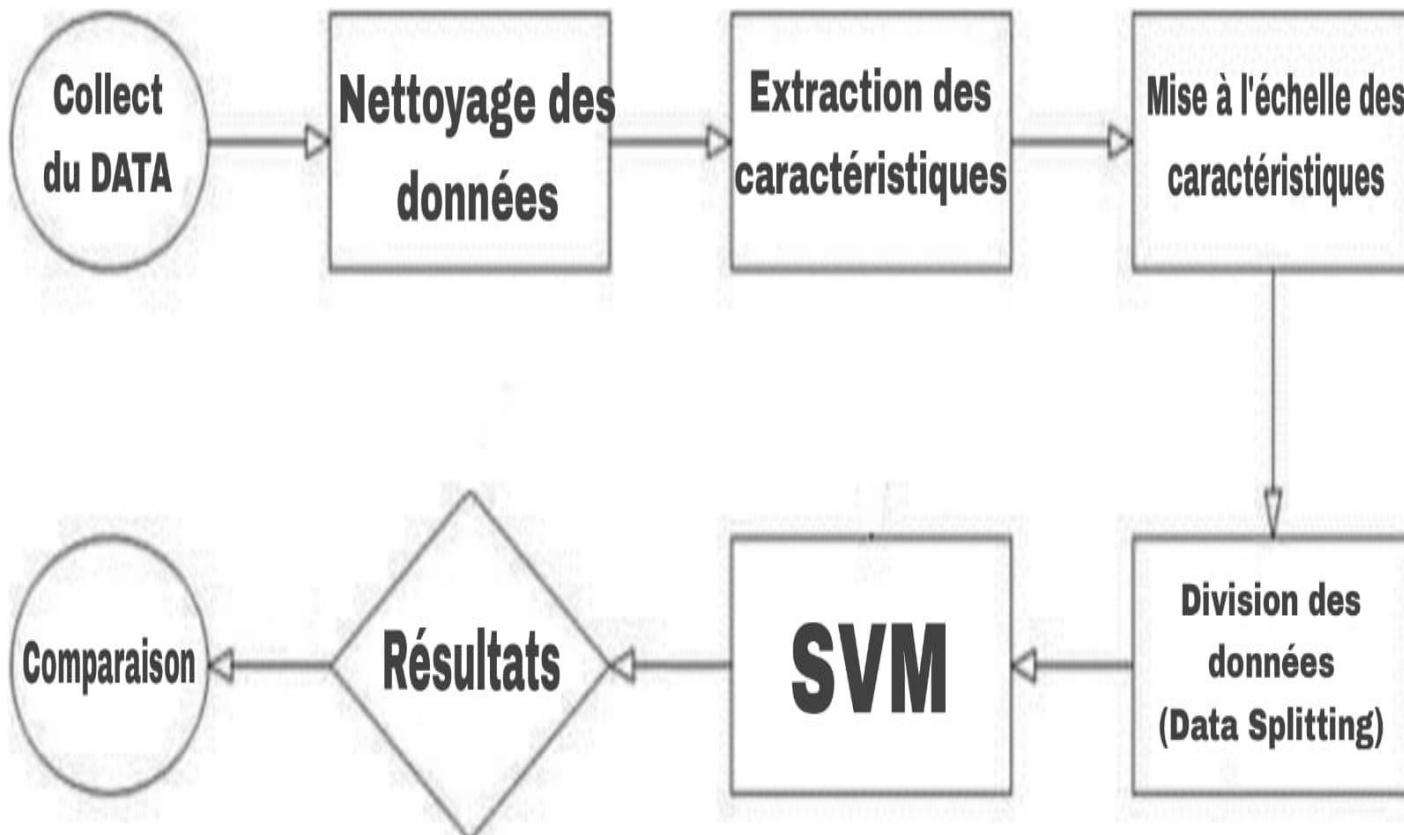
Division des données pour la classification (Data Splitting)

- Division des données:
 - Ensemble d'entraînement
 - Ensemble de test
 - Proportion de division
- la fonction ‘train_test_split’ de scikit-learn permet de diviser les données en ses 3 étapes

Machine à vecteurs de support (SVM)

PLAN :

- Autour de la SVM
- Effectuer le modèle SVM
- Comparaison des noyaux polynomial et linéaire



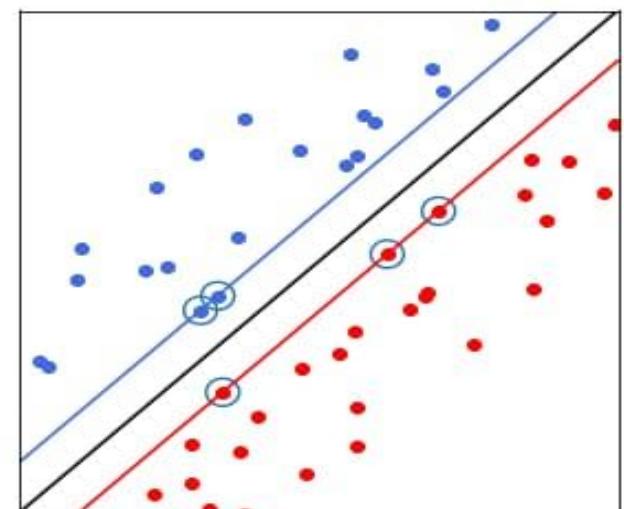
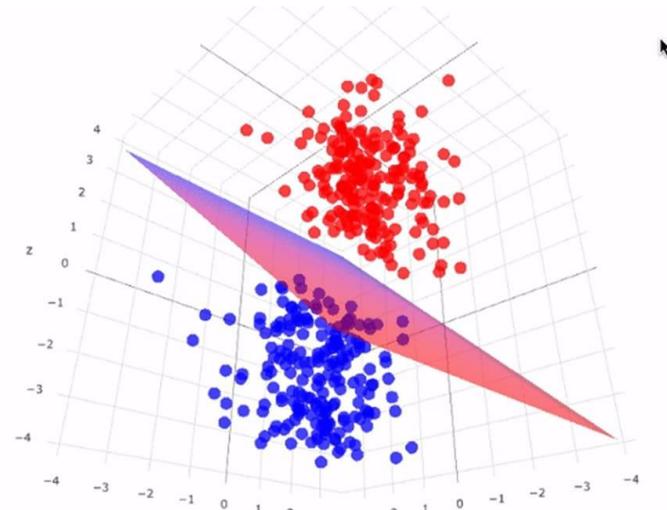
Plan

- Définition
- frontière de séparation
- La solution SVM
- Adaptation aux cas non linéairement séparables: l'astuce des fonctions noyau
- Noyau polynomial
- Noyau linéaire

Machine à vecteurs de support (SVM)

Les Machines à Vecteurs de Support (SVM) sont des algorithmes d'apprentissage supervisé utilisés pour la classification.

Ils visent à trouver l'hyperplan qui maximise la marge de séparation entre les différentes classes dans l'espace des données.

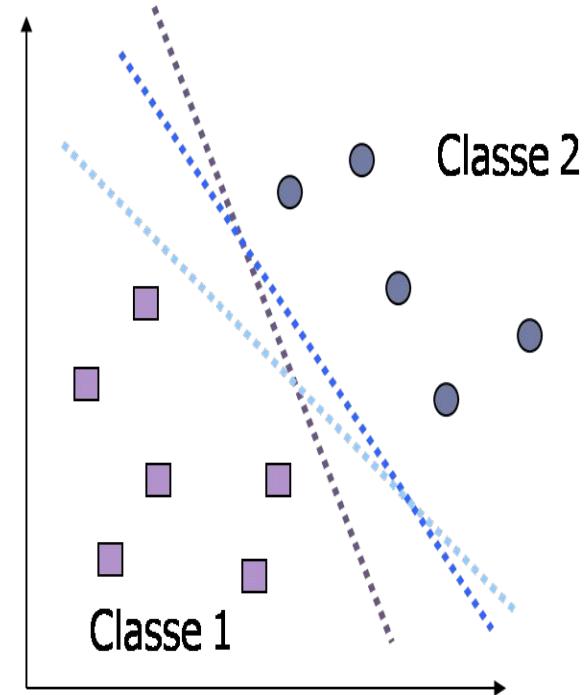


- **Performances de la SVM :**

- **Efficacité dans la séparation des classes** : trouver la frontière de séparation optimale entre les classes dans un espace multidimensionnel.
- **Capacité à gérer les données complexes** : traiter des données complexes et non linéairement séparables grâce à l'utilisation d'hyperplans dans des espaces de grande dimension.
- **Utilisation de vecteurs de support** : Ils identifient les points de données les plus pertinents pour la détermination de l'hyperplan de séparation.
- **Flexibilité grâce aux fonctions noyau** : utilise les différentes fonctions noyau pour transformer l'espace des caractéristiques.

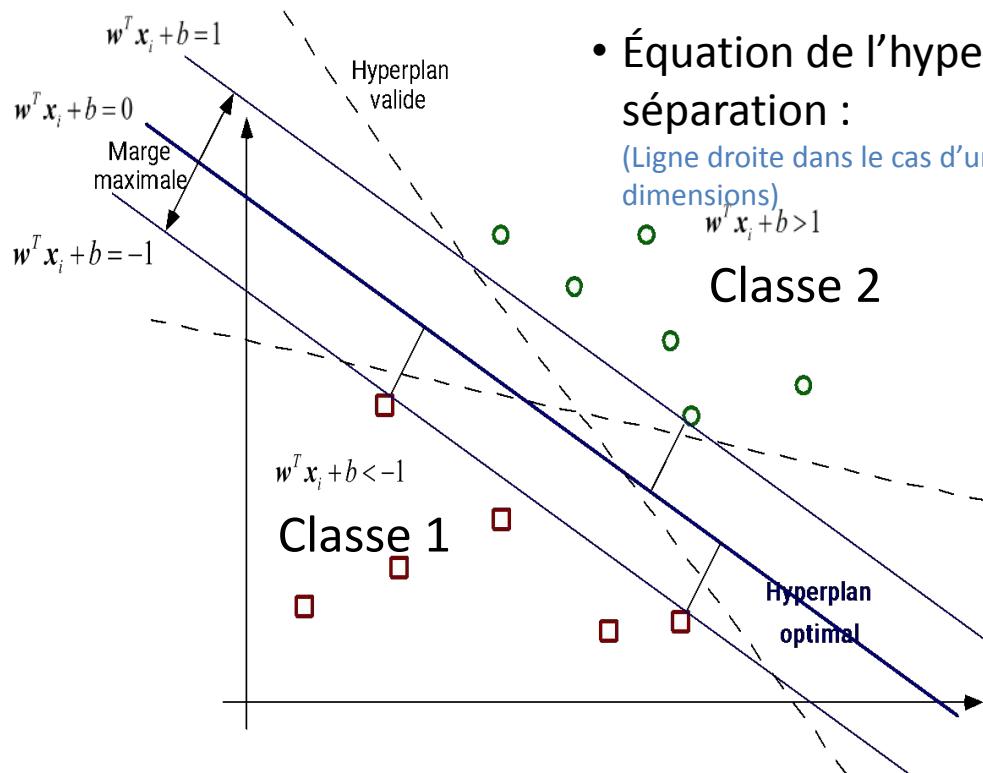
- **Frontière de séparation :**

- L'hyperplan trouvé par les SVM agit comme une frontière de séparation entre les différentes classes.
- Pour les problèmes de classification, cette frontière de séparation est définie par un ensemble de vecteurs de support, qui sont les points les plus proches de l'hyperplan.



- **La solution SVM :**
- optimiser une fonction objectif qui vise à maximiser la marge de séparation tout en minimisant l'erreur de classification.
- en résolvant un problème d'optimisation quadratique.

Hyperplan de plus vaste marge



- Équation de l'hyperplan de séparation : $y = w^T x + b$

(Ligne droite dans le cas d'un espace à deux dimensions)

- Si $\{x_1, \dots, x_n\}$ est l'ensemble des données et $y_i \in \{1, -1\}$ est la classe de chacune, on devrait avoir :

$$y_i(w^T x_i + b) \geq 1, \quad \forall i$$

Optimisation de la marge

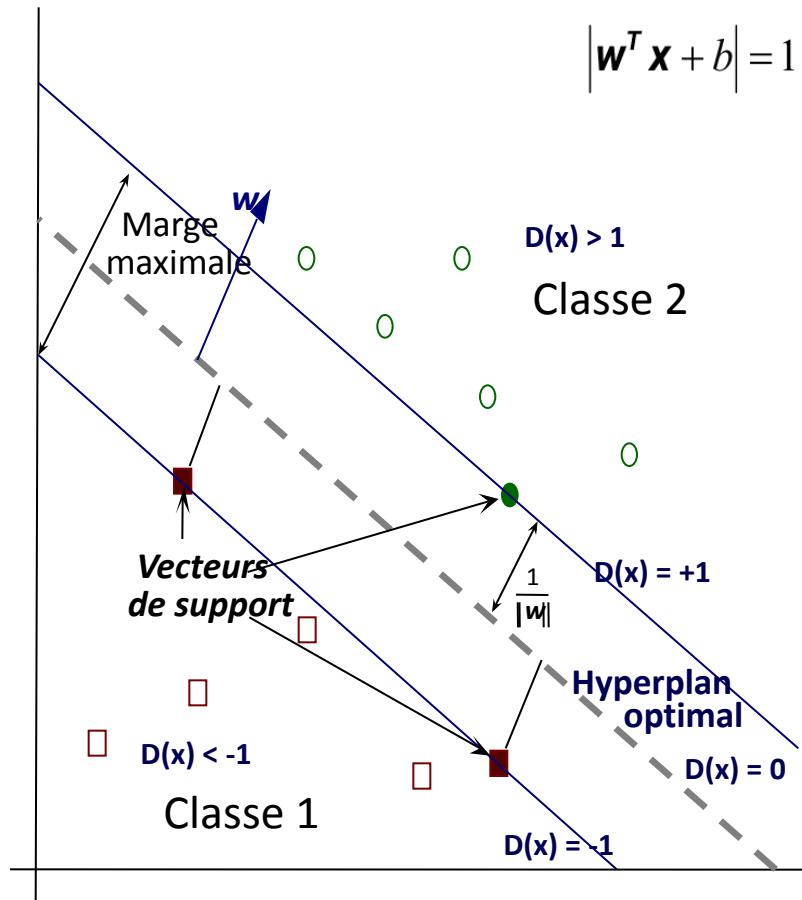
- Distance normale d'un point à l'hyperplan :

$$D(\mathbf{x}) = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|}$$

- Marge max. avant d'atteindre les frontières des deux classes

$$|\mathbf{w}^T \mathbf{x} + b| = 1$$

$$m = \frac{2}{\|\mathbf{w}\|}$$



Optimisation quadratique

□ Maximiser m revient à minimiser $\|w\|$:

$$\min \frac{1}{2} \|w\|^2 \text{ sous la contrainte: } y_i(w^T x_i + b) \geq 1, \forall i$$

□ Maximiser le pouvoir de généralisation du classeur revient à trouver w et b tels que :

$$\frac{1}{2} \|w\|^2 \quad \text{minimum.}$$

_____ et :

$$y_i(w^T x_i + b) \geq 1, \quad i=1, \dots, n$$

l'astuce des fonctions noyau :

transformer l'espace des caractéristiques dans un espace de dimension supérieure où les données peuvent devenir linéairement séparables.

- **Noyau polynomial** :

- transforme les données dans un espace de dimension supérieure en utilisant une fonction polynomiale.

- **Noyau linéaire** :

- ne réalise aucune transformation des données
- calcule le produit scalaire entre les vecteurs d'entrée.
- Bien qu'il soit simple, le noyau linéaire peut être efficace dans les cas où les données sont linéairement séparables dans l'espace d'origine.

Entraînement des modèles SVM

+ Modèle avec noyau polynomial

Accuracy: 0.7715231788079471

Confusion Matrix:

```
[[163  8]
 [ 61 70]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.73	0.95	0.83	171
1	0.90	0.53	0.67	131
accuracy			0.77	302
macro avg	0.81	0.74	0.75	302
weighted avg	0.80	0.77	0.76	302

+ Modèle avec noyau linéaire

Accuracy (Linear Kernel): 0.8576158940397351

Confusion Matrix (Linear Kernel):

```
[[158 13]
 [ 30 101]]
```

Classification Report (Linear Kernel):

	precision	recall	f1-score	support
0	0.84	0.92	0.88	171
1	0.89	0.77	0.82	131
accuracy			0.86	302
macro avg	0.86	0.85	0.85	302
weighted avg	0.86	0.86	0.86	302

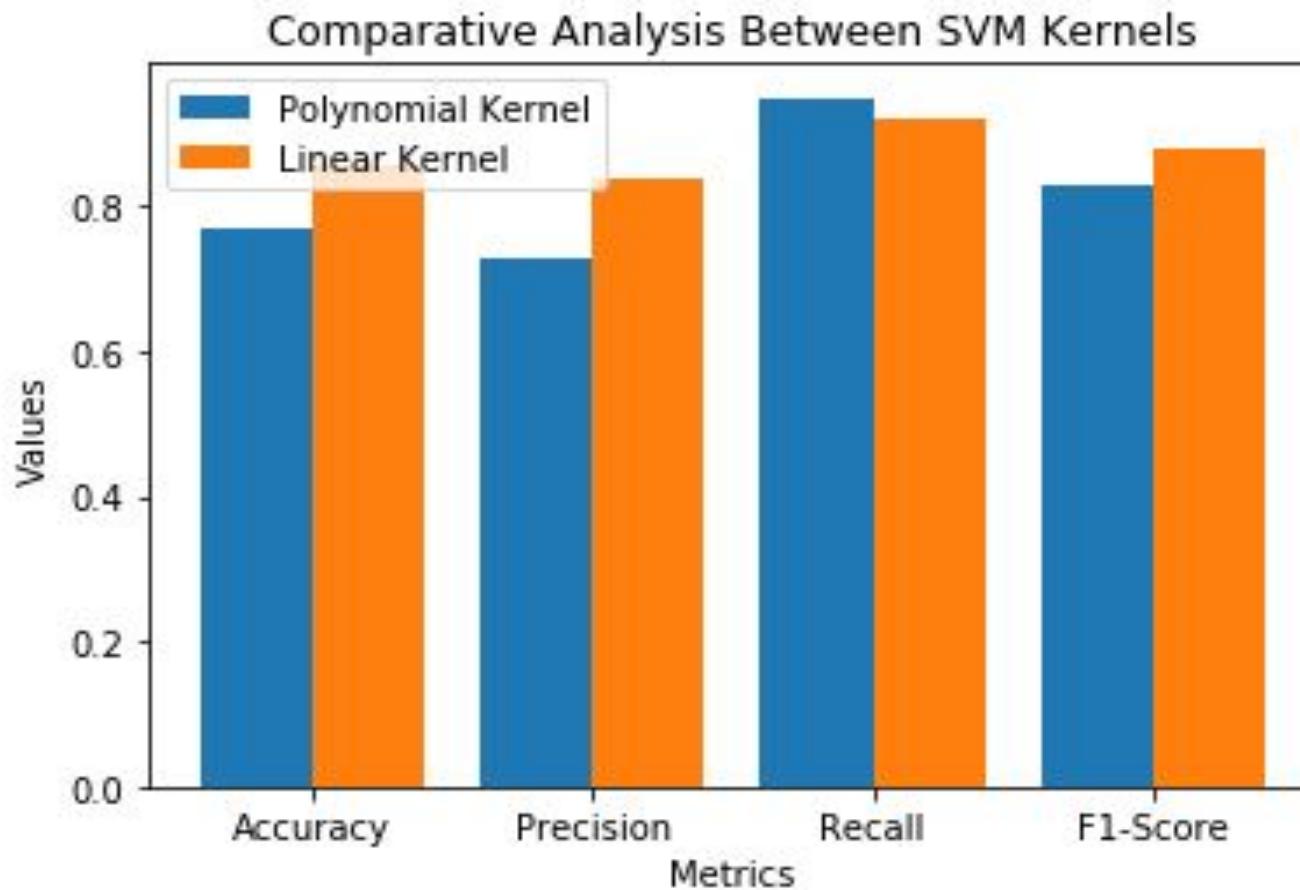
matrices de confusion

Résumé des valeurs TP, FP, FN, TN :

	Linear Kernel		Polynomial Kernel
True Positives (TP)	158		163
False Positives (FP)	13		8
False Negatives (FN)	30		61
True Negatives (TN)	101		70

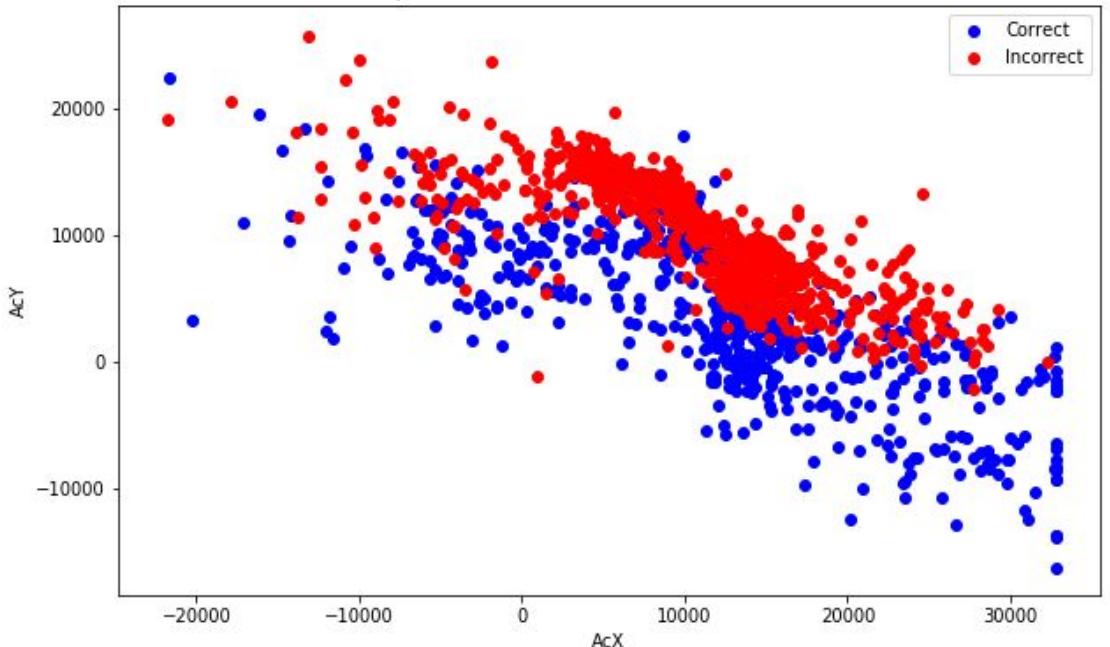
- TP (True Positives) : Nombre de vrais positifs.
- FP (False Positives) : Nombre de faux positifs..
- FN (False Negatives) : Nombre de faux négatifs..
- TN (True Negatives) : Nombre de vrais négatifs..

Comparaison des performances des deux noyaux

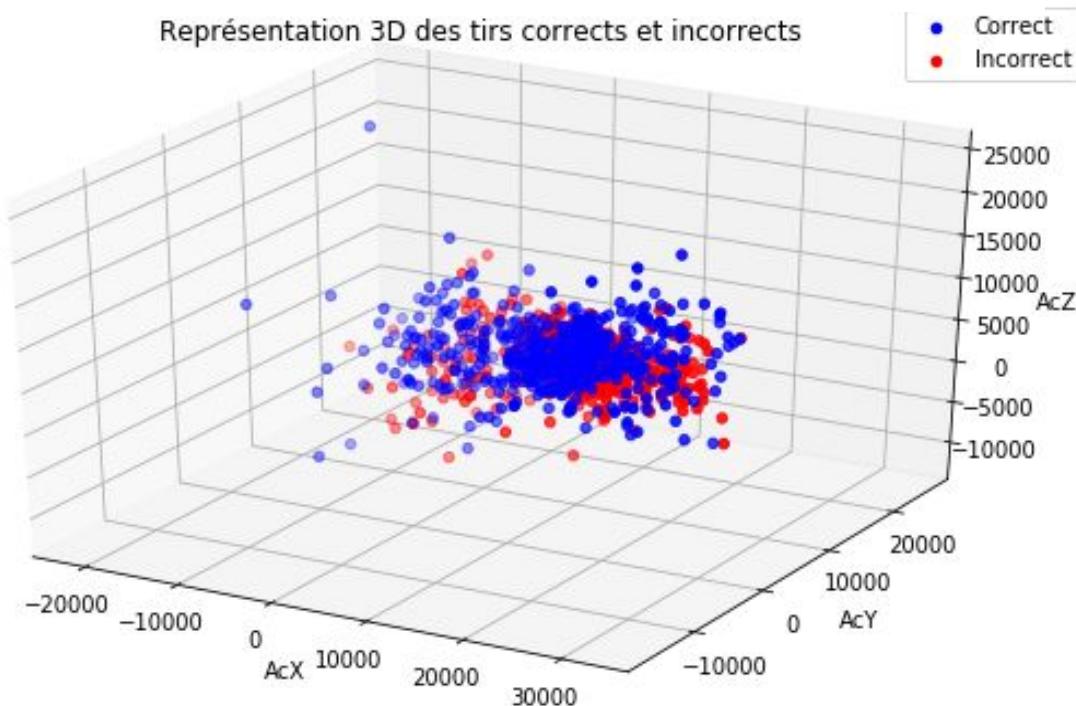


le modèle SVM avec un noyau linéaire offre une performance supérieure par rapport au noyau polynomial

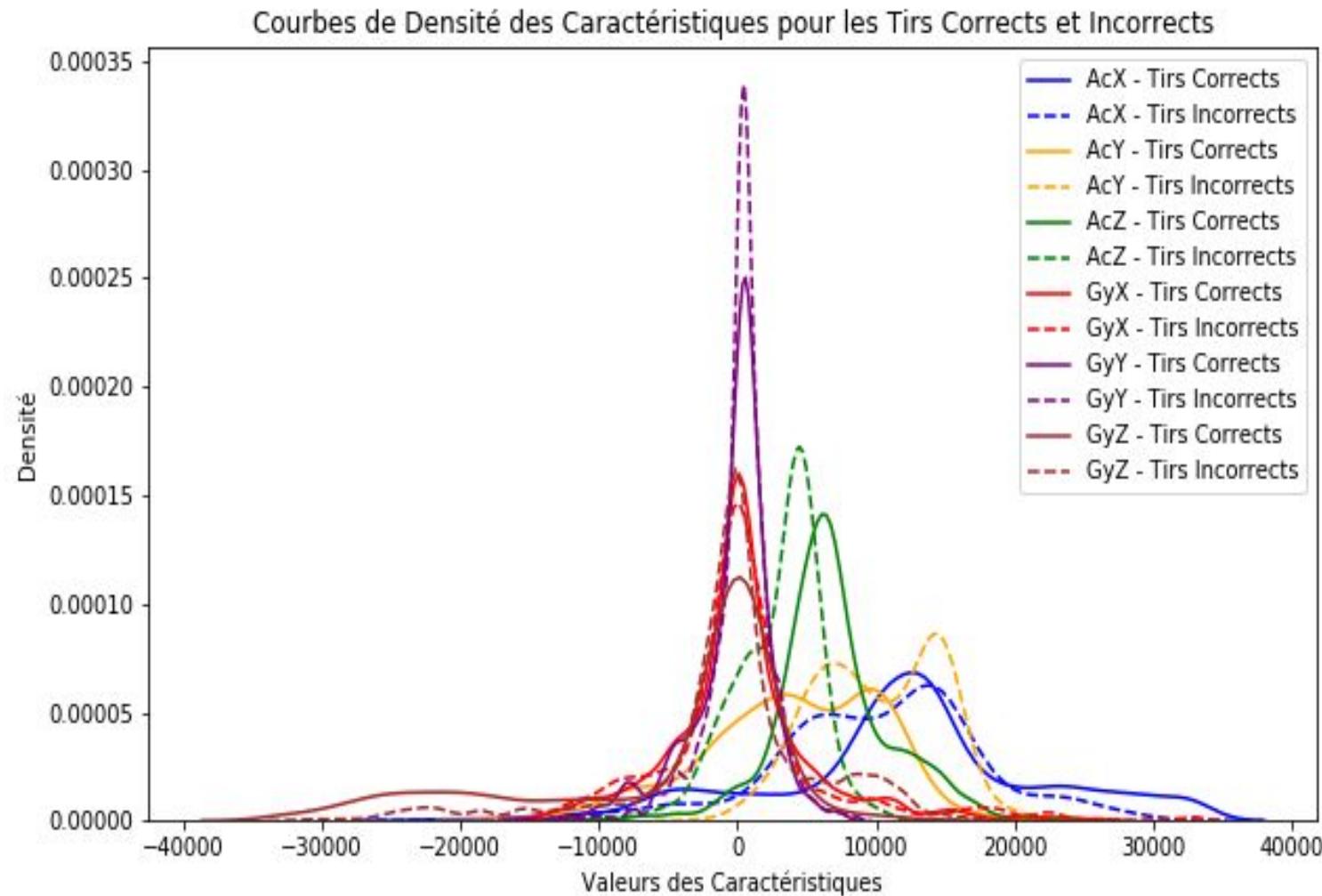
Représentation des tirs corrects et incorrects



Représentation 3D des tirs corrects et incorrects



Courbes de Densité des caractéristiques pour les tirs Corrects et Incorrects



Résultat & Conclusion

- **Précision Supérieure du Noyau Linéaire :**
- Le noyau linéaire a atteint une précision de 85.76%, surpassant le noyau polynomial qui a obtenu 77.15%.
- **Meilleure Classification des Tirs Corrects :**
- Le noyau linéaire a mieux classifié les tirs corrects (84% vs 73%).
- **Classification des Tirs Incorrects :**
- une bonne précision pour les deux (89% vs 90%)
- **Efficacité Générale du Noyau Linéaire :**
- En termes de F1-score, le noyau linéaire a montré une meilleure performance globale (0.88 vs 0.83)

MERCI POUR VOTRE
ATTENTION

Annexe (préparation des données)

Chargement des données (Data Loading) Et Nettoyage des données (Data Cleaning)

```
Entrée [100]: import pandas as pd

# Lire les données à partir des fichiers texte
def read_data(file_path):
    data = []
    with open(file_path, 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            # Vérifier si La Ligne contient exactement 8 parties et si elles peuvent être converties en entiers
            if len(parts) == 8:
                try:
                    # Convertir les parties en entiers
                    parts = [int(part) for part in parts]
                    data.append(parts)
                except ValueError:
                    continue
    return data

# Lire les données des tirs corrects et incorrects
correct_data = read_data('tirs1.txt')
incorrect_data = read_data('tirs2.txt')

# Modifier la dernière colonne de incorrect_data en mettant toutes les valeurs à 0
for row in incorrect_data:
    row[-1] = 0

# Combiner les données et convertir en DataFrame
all_data = correct_data + incorrect_data
df = pd.DataFrame(all_data, columns=['timestape', 'AcX', 'AcY', 'AcZ', 'GyX', 'GyY', 'GyZ', 'Label'])
```

```
Entrée [103]: # Vérifier les valeurs manquantes et supprimer les lignes correspondantes
df = df.dropna()
```

Feature Extraction - Feature scaling

```
from sklearn.preprocessing import MinMaxScaler

# Extraire les caractéristiques et les étiquettes
features = df[['AcX', 'AcY', 'AcZ', 'GyX', 'GyY', 'GyZ']]
labels = df['Label']

# Normaliser les valeurs
scaler = MinMaxScaler()
features_scaled = scaler.fit_transform(features)
```

Division des données (Data Splitting)

```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(features_scaled, labels, test_size=0.2, random_state=42)
```

Annexe modèles svm

Modèle SVM ‘poly’

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# Entrainer un modèle SVM avec un noyau polynomial
clf = SVC(kernel='poly', degree=3)
clf.fit(X_train, y_train)
# Évaluer la performance du modèle
y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Modèle SVM ‘linéaire’

```
# Entrainer un modèle SVM avec un noyau linéaire
clf_linear = SVC(kernel='linear')
clf_linear.fit(X_train, y_train)

# Évaluer la performance du modèle
y_pred_linear = clf_linear.predict(X_test)
print("Accuracy (Linear Kernel):", accuracy_score(y_test, y_pred_linear))
print("Confusion Matrix (Linear Kernel):\n", confusion_matrix(y_test, y_pred_linear))
print("Classification Report (Linear Kernel):\n", classification_report(y_test, y_pred_linear))
```

Annexe comparaisons

Diagramme des performances

```
import matplotlib.pyplot as plt

# Résultats des noyaux polynomial et linéaire
accuracies = [0.7715231788079471, 0.8576158940397351]
precisions = [0.73, 0.84]
recalls = [0.95, 0.92]
f1_scores = [0.83, 0.88]

# Labels pour les métriques
metrics = ['Accuracy', 'Precision', 'Recall', 'F1-Score']

# Valeurs pour les noyaux polynomial et linéaire
polynomial_values = [accuracies[0], precisions[0], recalls[0], f1_scores[0]]
linear_values = [accuracies[1], precisions[1], recalls[1], f1_scores[1]]

# Positions pour les barres
positions = range(len(metrics))

# Tracé des barres
plt.bar(positions, polynomial_values, width=0.4, label='Polynomial Kernel')
plt.bar([p + 0.4 for p in positions], linear_values, width=0.4, label='Linear Kernel')

# Ajout des titres et des étiquettes
plt.title('Comparative Analysis Between SVM Kernels')
plt.xlabel('Metrics')
plt.ylabel('Values')
plt.xticks([p + 0.2 for p in positions], metrics)
plt.legend()

# Affichage du diagramme
plt.show()
```

Courbes de densité

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

# Séparer les données en fonction du Label
correct_shots = df[df['Label'] == 1]
incorrect_shots = df[df['Label'] == 0]

# Créer des courbes de densité pour chaque caractéristique avec les couleurs spécifiées
plt.figure(figsize=(10, 6))
colors = ['blue', 'orange', 'green', 'red', 'purple', 'brown'] # Couleurs pour chaque caractéristique
for i, feature in enumerate(['AcX', 'AcY', 'AcZ', 'GyX', 'GyY', 'GyZ']):
    sns.kdeplot(correct_shots[feature], label=f'{feature} - Tirs Corrects', color=colors[i])
    sns.kdeplot(incorrect_shots[feature], label=f'{feature} - Tirs Incorrects', color=colors[i], linestyle='--') # Utilise
plt.xlabel('Valeurs des Caractéristiques')
plt.ylabel('Densité')
plt.title('Courbes de Densité des Caractéristiques pour les Tirs Corrects et Incorrects')
plt.legend()
plt.show()
```

Représentation en 3D

```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Lire les données à partir des fichiers texte
def read_data(file_path):
    data = []
    with open(file_path, 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            if len(parts) == 8:
                try:
                    parts = [int(part) for part in parts]
                    data.append(parts)
                except ValueError:
                    continue
    return data

# Lire les données des tirs corrects et incorrects
correct_data = read_data('tirs1.txt')
incorrect_data = read_data('tirs2.txt')

# Modifier la dernière colonne de incorrect_data en mettant toutes les valeurs à 0
for row in incorrect_data:
    row[-1] = 0

# Combiner les données
all_data = correct_data + incorrect_data
df = pd.DataFrame(all_data, columns=['timestamp', 'AcX', 'AcY', 'AcZ', 'GyX', 'GyY', 'GyZ', 'Label'])

# Afficher le graphique en 3D
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df[df['Label'] == 1]['AcX'], df[df['Label'] == 1]['AcY'], df[df['Label'] == 1]['AcZ'], label='Correct', color='blue')
ax.scatter(df[df['Label'] == 0]['AcX'], df[df['Label'] == 0]['AcY'], df[df['Label'] == 0]['AcZ'], label='Incorrect', color='red')
ax.set_xlabel('AcX')
ax.set_ylabel('AcY')
ax.set_zlabel('AcZ')
ax.set_title('Représentation 3D des tirs corrects et incorrects')
ax.legend()
plt.show()
```