Implémentation de la méthode SVM

Réalisé par :

Kawtar Oukil

Année 2021-2022

Table des matières

Chapitre 1 :

Introduction

Les machines à vecteurs de support (ou Support Vector Machine, SVM) sont une famille d'algorithmes d'apprentissage automatique de type supervisé et qui peuvent être utilisées pour des problèmes de discrimination (à quelle classe appartient un échantillon), de régression et de détection d'anomalies.

Les machines à vecteurs de support (SVM en abrégé) sont des algorithmes d'apprentissage automatique utilisés à des fins de classification et de régression. Les SVM sont l'un des puissants algorithmes d'apprentissage automatique à des fins de classification, de régression et de détection des valeurs aberrantes. Un classificateur SVM construit un modèle qui attribue de nouveaux points de données à l'une des catégories données. Ainsi, il peut être considéré comme un classificateur linéaire binaire non probabiliste.

nous nous intéresserons aux différents SVM de classification mise en place par la bibliothèque d'apprentissage automatique Scikit-Learn de Python.

1. L'intuition du Support Vecteur Machines

Hyperplan

Un hyperplan est une frontière de décision qui sépare un ensemble donné de points de données ayant différentes étiquettes de classe. Le classificateur SVM sépare les points de données à l'aide d'un hyperplan avec le maximum de marge. Cet hyperplan est appelé hyperplan de marge maximale et le classificateur linéaire qu'il définit est appelé classificateur de marge maximale.

Vecteurs de support

Les vecteurs de support sont les exemples de points de données, qui sont les plus proches de l'hyperplan. Ces points de données définiront mieux la ligne de séparation ou l'hyperplan en calculant les marges.

Marge

Une marge est un espace de séparation entre les deux lignes sur les points de données les plus proches. Elle est calculée comme la distance perpendiculaire entre la ligne et les vecteurs de support ou les points de données les plus proches. Dans les SVM, nous essayons de maximiser cet écart de séparation afin d'obtenir une marge maximale

Les SVM peuvent être utilisés à des fins de classification linéaire. En plus d'effectuer une classification linéaire, les SVM peuvent effectuer efficacement une classification non linéaire en utilisant l'astuce du noyau. Cela nous permet de mapper implicitement les entrées dans des espaces de caractéristiques de grande dimension.

en un espace de dimension supérieure, comme indiqué dans le diagramme ci-dessous. Il utilise une fonction de mappage pour transformer l'espace d'entrée 2D en espace d'entrée 3D. Maintenant, nous pouvons facilement séparer les points de données en utilisant une séparation linéaire.

2. Astuce du noyau

En pratique, l'algorithme SVM est implémenté à l'aide d'un noyau. Il utilise une technique appelée l'astuce du noyau. En termes simples, un noyau est juste une fonction qui mappe les données à une dimension supérieure où les données sont séparables. Un noyau transforme un espace de données

d'entrée de faible dimension en un espace de dimension supérieure. Ainsi, il convertit les problèmes séparables non linéaires en problèmes séparables linéaires en y ajoutant plus de dimensions. Ainsi, l'astuce du noyau nous aide à construire un classificateur plus précis. Par conséquent, il est utile dans les problèmes de séparation non linéaires.

Dans le contexte des SVM, il existe 4 noyaux populaires - noyau linéaire, noyau polynomial, noyau de fonction de base radiale (RBF) (également appelé noyau gaussien) et noyau sigmoïde. Ceux-ci sont décrits ci-dessous –

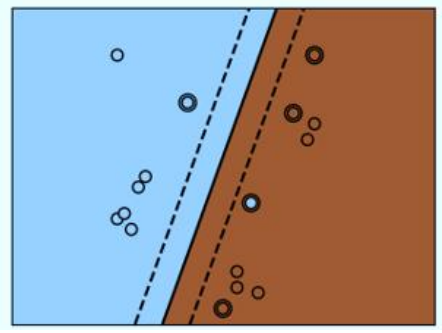
2.1 Noyau linéaire

Dans le noyau linéaire, la fonction du noyau prend la forme d'une fonction linéaire comme suit-

noyau linéaire : $K(xi , xj) = xi^T xj$

Le noyau linéaire est utilisé lorsque les données sont linéairement séparables. Cela signifie que les données peuvent être séparées à l'aide d'une seule ligne. C'est l'un des noyaux les plus couramment utilisés. Il est principalement utilisé lorsqu'il existe un grand nombre d'entités dans un jeu de données. Le noyau linéaire est souvent utilisé à des fins de classification de texte.

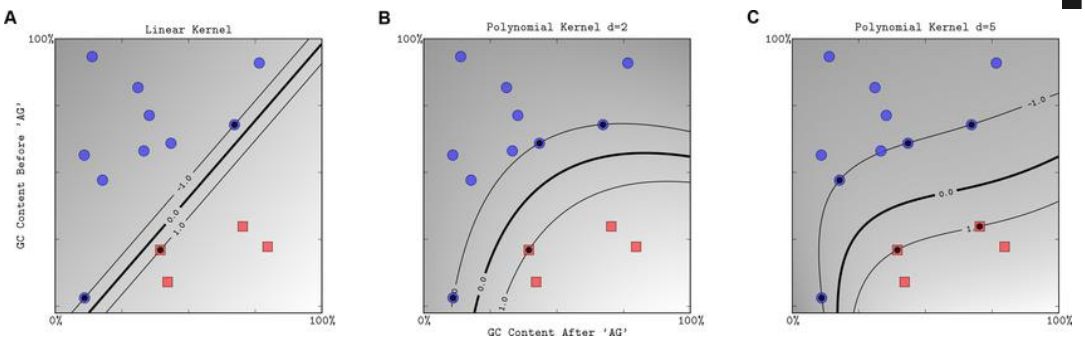
L'entraînement avec un noyau linéaire est généralement plus rapide, car nous n'avons besoin que d'optimiser le paramètre de régularisation C. Lors de l'entraînement avec d'autres noyaux, nous devons également optimiser le paramètre γ. Ainsi, effectuer une recherche par grille prendra généralement plus de temps.Le noyau linéaire peut être visualisé avec la figure suivante.



2.2 Noyau polynomial

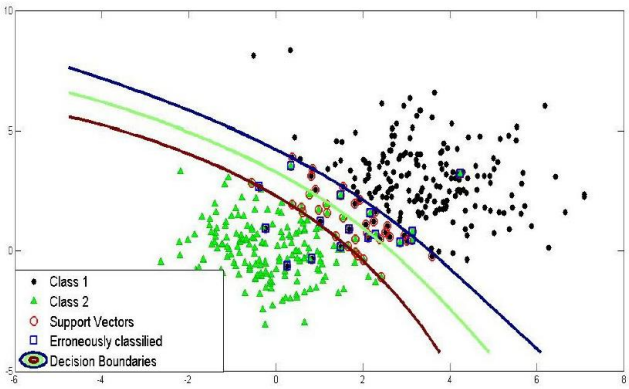
Le noyau polynomial représente la similarité des vecteurs (échantillons d'apprentissage) dans un espace de caractéristiques sur les polynômes des variables d'origine. Le noyau polynomial examine non seulement les caractéristiques données des échantillons d'entrée pour déterminer leur similarité, mais également les combinaisons des échantillons d'entrée.

Le noyau polynomial est très populaire dans le traitement du langage naturel. Le degré le plus courant est d = 2 (quadratique), car les degrés plus élevés ont tendance à surajuster les problèmes de PNL. Il peut être visualisé avec le schéma suivant.



2.3 Noyau de la fonction de base radiale

Le noyau de fonction de base radiale est un noyau à usage général. Il est utilisé lorsque nous n'avons aucune connaissance préalable des données. Le noyau RBF sur deux échantillons x et y est défini par l'équation suivante -



2.4 Noyau sigmoïde

Le noyau sigmoïde trouve son origine dans les réseaux de neurones. Nous pouvons l'utiliser comme proxy pour les réseaux de neurones.

Le noyau sigmoïde peut être visualisé avec le diagramme suivant -

Chapitre 2 :

Implémentation

Bibliothèques SVM Scikit-Learn

Scikit-Learn fournit des bibliothèques utiles pour implémenter l'algorithme Support Vector Machine sur un ensemble de données. Il existe de nombreuses bibliothèques qui peuvent nous aider à implémenter SVM en douceur. Nous avons juste besoin d'appeler la bibliothèque avec des paramètres adaptés à nos besoins. Dans ce projet, je m'occupe d'une tâche de classification. Je mentionnerai donc les bibliothèques Scikit-Learn à des fins de classification SVM

```
from sklearn.svm import SVC
classifier = SVC()
classifier.fit(X_Train, Y_Train)
```

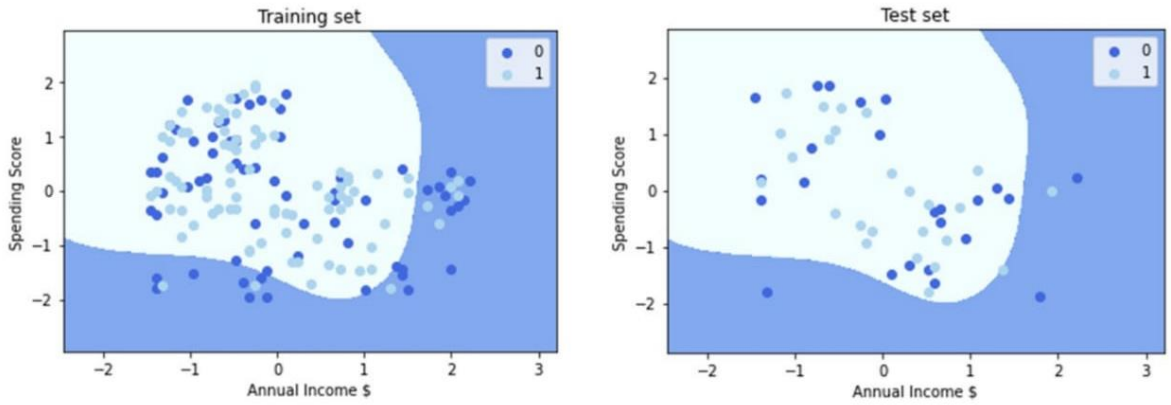
Jeu de données 1 : Mall_customers

On a choisi d'utiliser l'ensemble de données Mall_customers.csv qui contient des informations concernant les clients d'une entreprise : leur ID, âge, genre et leur revenu annuel et leur note de dépenses. Et on a choisit d'expliquer le genre par rapport à le revenu annuel et la note de dépenses. Les étiquettes de classe utilisées sont 0 (négatif) et 1 (positif).

Exécuter SVM avec l'hyperparamètre par défaut :

L'hyperparamètre par défaut signifie

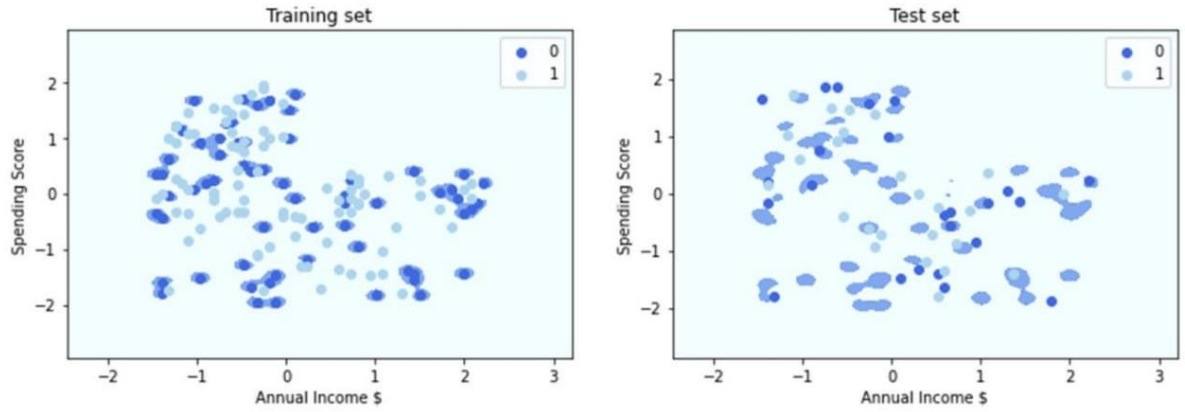
- **C=1** et **kernel='rbf'** et **gamma=auto**



Les supports Vecteurs :

```
1  =>  [ 1.50790978 -1.83258185]
2  =>  [0.02726271 0.98047619]
3  =>  [ 0.66182574 -0.04965774]
4  =>  [-0.39577931  0.38616816]
5  =>  [-0.46628632  0.90123512]
6  =>  [-0.95983534 -1.51561757]
7  =>  [ 1.9309518  -0.08927827]
8  =>  [-0.1137513  -1.47599703]
9  =>  [-0.25476531  0.42578869]
10  =>  [-0.46628632  1.69364584]
11  =>  [-0.74831433  0.70313244]
12  =>  [-1.38287736 -1.59485864]
13  =>  [ 2.07196581 -0.28738095]
14  =>  [ 1.01436076 -0.16851935]
15  =>  [-0.46628632  0.94085566]
16  =>  [ 2.0014588  -1.43637649]
```

C=1 et gamma=100

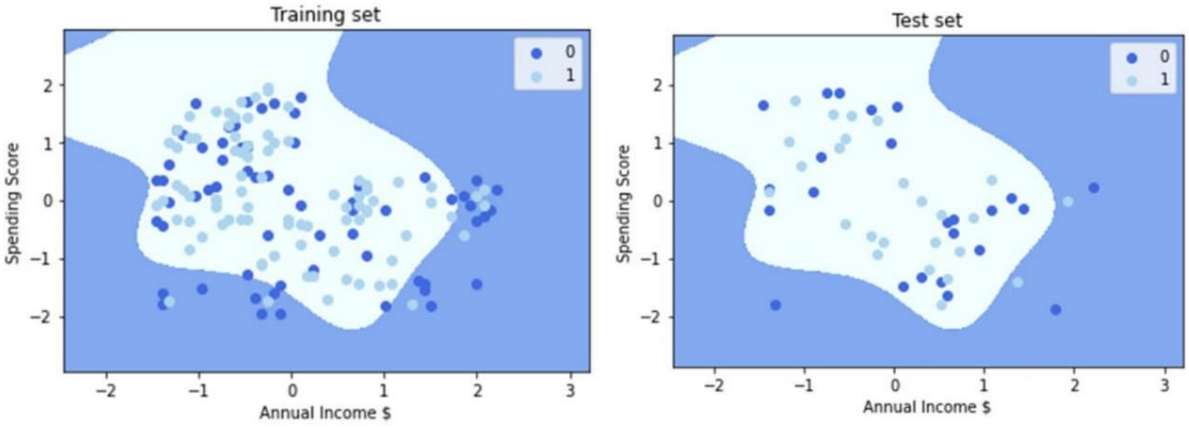


Les supports Vecteurs et Accuracy score :

```
135  =>  [ 0.16827671 -1.31751489]
136  =>  [ 0.94385375 -1.47599703]
137  =>  [-0.32527231 -1.07979167]
138  =>  [-1.31237036  0.98047619]
139  =>  [-0.81882133  0.02958333]
140  =>  [ 1.29638877 -1.79296132]
141  =>  [-0.32527231  0.38616816]
142  =>  [-1.17135635  0.18806548]
143  =>  [ 0.23878372 -1.31751489]
144  =>  [-0.60730032  0.86161459]
145  =>  [ 2.07196581 -0.08927827]
146  =>  [0.73233274 0.34654762]
147  =>  [-0.81882133 -0.12889881]
Model accuracy score :0.5400
```

C = 10

```
# Fitting the classifier into the Training set
classifier = SVC(C=10)
classifier.fit(X_Train, Y_Train)
```



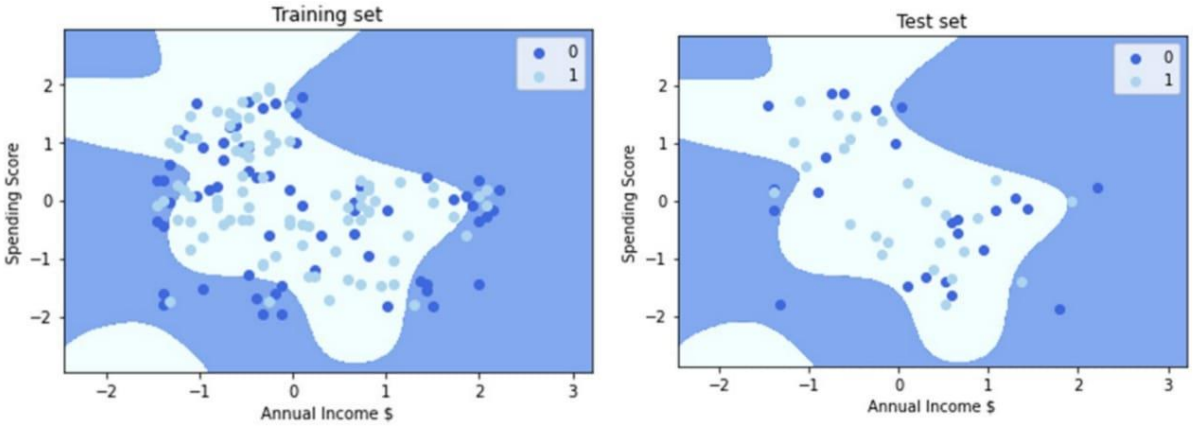
Les supports Vecteurs :

```
0  =>  [-0.32527231  1.57478423]
1  =>  [0.02726271 0.98047619]
2  =>  [ 0.66182574 -0.04965774]
3  =>  [-0.39577931  0.38616816]
4  =>  [-0.46628632  0.90123512]
5  =>  [-0.95983534 -1.51561757]
6  =>  [ 1.9309518  -0.08927827]
7  =>  [-0.1137513  -1.47599703]
8  =>  [-0.25476531  0.42578869]
9  =>  [-0.46628632  1.69364584]
10  =>  [-0.74831433  0.70313244]
11  =>  [-1.38287736 -1.59485864]
12  =>  [ 2.07196581 -0.28738095]
13  =>  [ 1.01436076 -0.16851935]
14  =>  [-0.46628632  0.94085566]
```

Accuracy Score :

```
Model accuracy score :0.5400
```

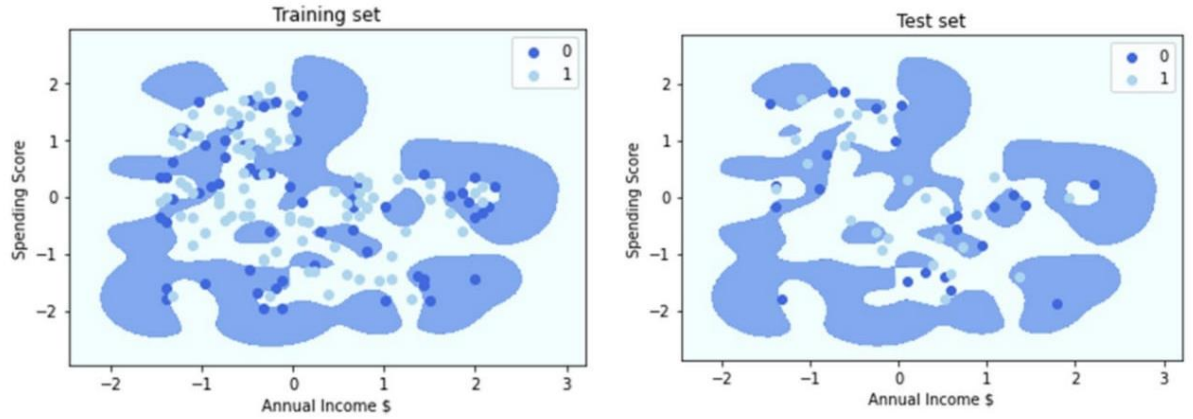
C=100 et gamma=0.1



Les Support Vecteurs et Accuracy Score :

```
103  =>  [ 1.50790978 -0.04965774]
104  =>  [-0.1842583   0.98047619]
105  =>  [ 0.16827671 -1.31751489]
106  =>  [-0.32527231 -1.07979167]
107  =>  [-1.31237036  0.98047619]
108  =>  [-0.81882133  0.02958333]
109  =>  [ 1.29638877 -1.79296132]
110  =>  [-0.32527231  0.38616816]
111  =>  [-1.17135635  0.18806548]
112  =>  [ 0.23878372 -1.31751489]
113  =>  [-0.60730032  0.86161459]
114  =>  [ 2.07196581 -0.08927827]
115  =>  [0.73233274 0.34654762]
Model accuracy score :0.5800
```

C=100 et gamma=10



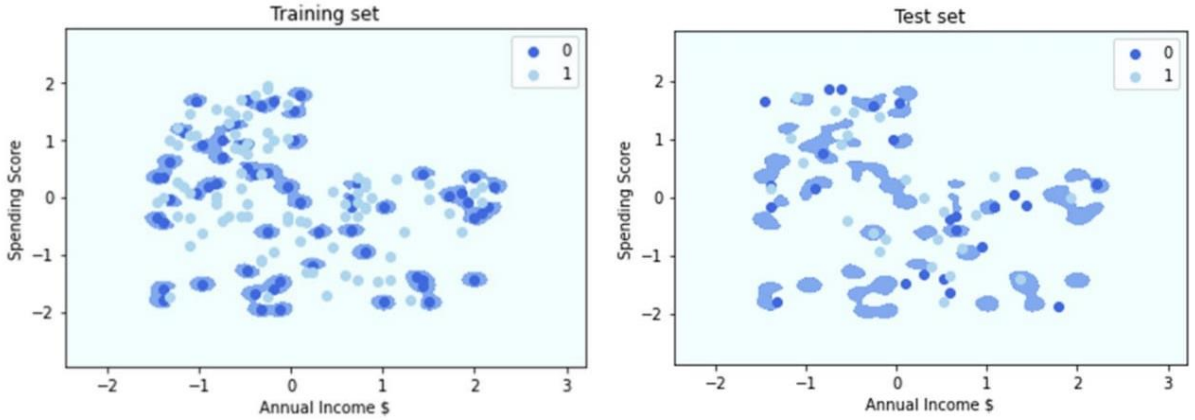
Les Support Vecteurs et Accuracy Score

:

```
90  =>  [-0.32527231 -1.11941221]
91  =>  [-0.0432443   1.61440477]
92  =>  [ 1.50790978 -0.04965774]
93  =>  [ 0.16827671 -1.31751489]
94  =>  [ 0.94385375 -1.47599703]
95  =>  [-1.31237036  0.98047619]
96  =>  [-0.81882133  0.02958333]
97  =>  [ 1.29638877 -1.79296132]
98  =>  [-0.32527231  0.38616816]
99  =>  [ 0.23878372 -1.31751489]
100  =>  [-0.60730032  0.86161459]
101  =>  [ 2.07196581 -0.08927827]
102  =>  [0.73233274 0.34654762]
Model accuracy score :0.5800
```

Nous avons vu qu'il y a des valeurs aberrantes dans notre jeu de données. Nous devrions donc augmenter la valeur de C car un C plus élevé signifie moins de valeurs aberrantes. Donc, je vais exécuter SVM avec kernel=rbf et C=100.0. et gamma=100

C=1000 et gamma=100



Les Support Vecteurs et Accuracy Score :

```
135  =>  [ 0.16827671 -1.31751489]
136  =>  [ 0.94385375 -1.47599703]
137  =>  [-0.32527231 -1.07979167]
138  =>  [-1.31237036  0.98047619]
139  =>  [-0.81882133  0.02958333]
140  =>  [ 1.29638877 -1.79296132]
141  =>  [-0.32527231  0.38616816]
142  =>  [-1.17135635  0.18806548]
143  =>  [ 0.23878372 -1.31751489]
144  =>  [-0.60730032  0.86161459]
145  =>  [ 2.07196581 -0.08927827]
146  =>  [0.73233274 0.34654762]
147  =>  [-0.81882133 -0.12889881]
Model accuracy score :0.5200
```

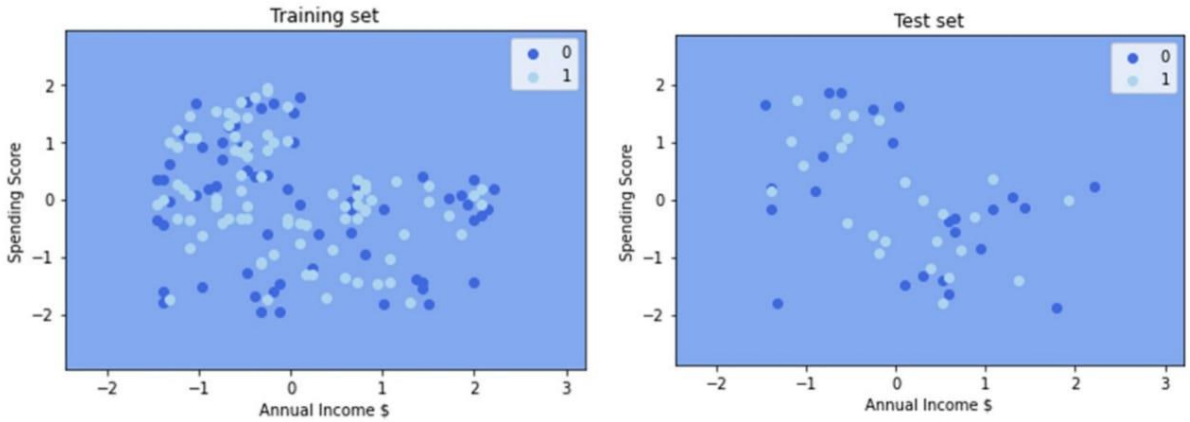
On remarque dans le cas d'un noyau en base radiale (rbf) le coefficient gamma joue un rôle important sur l'efficacité de la classification.

⇨ plus gamma est grande plus le modèle est efficace.

Exécuter SVM avec noyau linéaire :

C=1

```
# Fitting the classifier into the Training set
classifier = SVC(kernel='linear',C=1)
classifier.fit(X_Train, Y_Train)
```



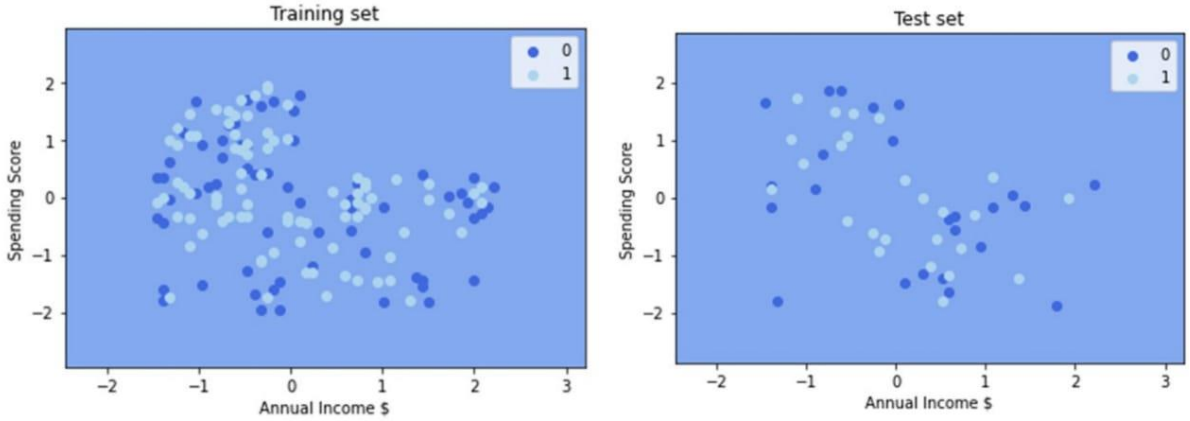
Les Support Vecteurs et Accuracy Score :

```
116  =>  [-0.0432443   1.61440477]
117  =>  [ 1.50790978 -0.04965774]
118  =>  [-0.1842583   0.98047619]
119  =>  [ 0.16827671 -1.31751489]
120  =>  [ 0.94385375 -1.47599703]
121  =>  [-0.32527231 -1.07979167]
122  =>  [-0.81882133  0.02958333]
123  =>  [ 1.29638877 -1.79296132]
124  =>  [-0.32527231  0.38616816]
125  =>  [-1.17135635  0.18806548]
126  =>  [ 0.23878372 -1.31751489]
127  =>  [-0.60730032  0.86161459]
128  =>  [ 2.07196581 -0.08927827]
Model accuracy score :0.5200
```

Fonction de l'hyperplan :

```
function of hyperplan :  y = 1.000 + -0.000 * x1 + 0.000 * x2
```

C=10000



Les Support Vecteurs et Accuracy Score :

```
121  =>  [ 1.50790978 -0.04965774]
122  =>  [-0.1842583   0.98047619]
123  =>  [ 0.16827671 -1.31751489]
124  =>  [ 0.94385375 -1.47599703]
125  =>  [-0.32527231 -1.07979167]
126  =>  [-1.31237036  0.98047619]
127  =>  [ 1.29638877 -1.79296132]
128  =>  [-0.32527231  0.38616816]
129  =>  [-1.17135635  0.18806548]
130  =>  [ 0.23878372 -1.31751489]
131  =>  [-0.60730032  0.86161459]
132  =>  [ 2.07196581 -0.08927827]
133  =>  [-0.81882133 -0.12889881]
Model accuracy score :0.5200
```

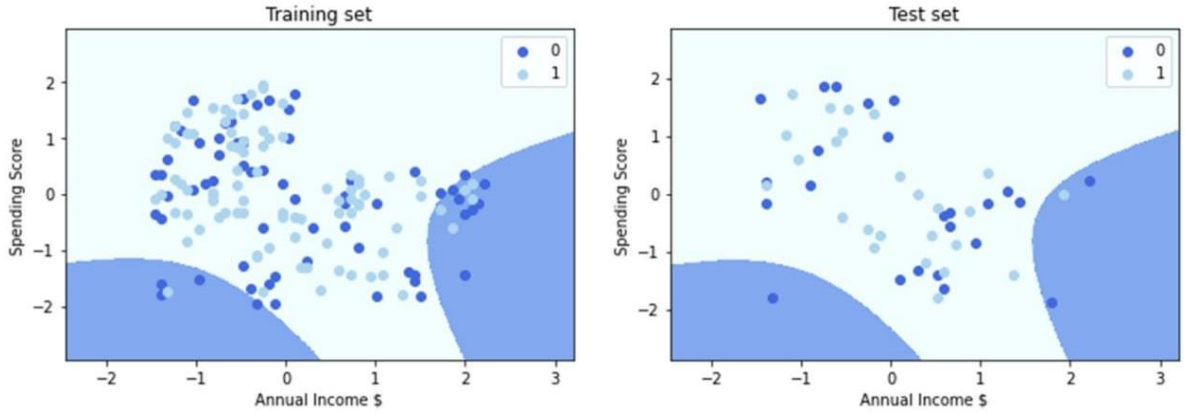
Fonction de l'hyperplan :

```
function of hyperplan :  y = 0.997 + -0.001 * x1 + 0.001 * x2
```

On remarque dans le cas d'un noyau linéaire, le modèle n'est pas efficace dans tous les cas même si on augmente C , seul le nombre des supports vecteurs qui se diffère , mais le résultat reste le même.

Exécuter SVM avec un noyau polynomial :

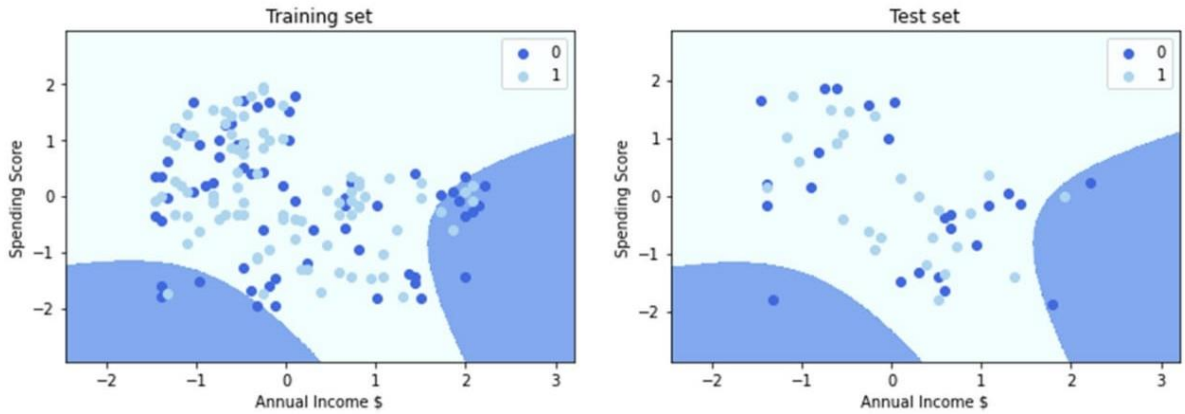
C=1



Les Support Vecteurs et Accuracy Score :

```
113  =>  [ 0.73233274 -0.32700149]
114  =>  [-0.32527231 -1.11941221]
115  =>  [ 1.50790978 -0.04965774]
116  =>  [-0.1842583   0.98047619]
117  =>  [ 0.16827671 -1.31751489]
118  =>  [ 0.94385375 -1.47599703]
119  =>  [-0.32527231 -1.07979167]
120  =>  [ 1.29638877 -1.79296132]
121  =>  [-0.32527231  0.38616816]
122  =>  [ 0.23878372 -1.31751489]
123  =>  [-0.60730032  0.86161459]
124  =>  [ 2.07196581 -0.08927827]
125  =>  [0.73233274 0.34654762]
Model accuracy score :0.5600
```

C=1000

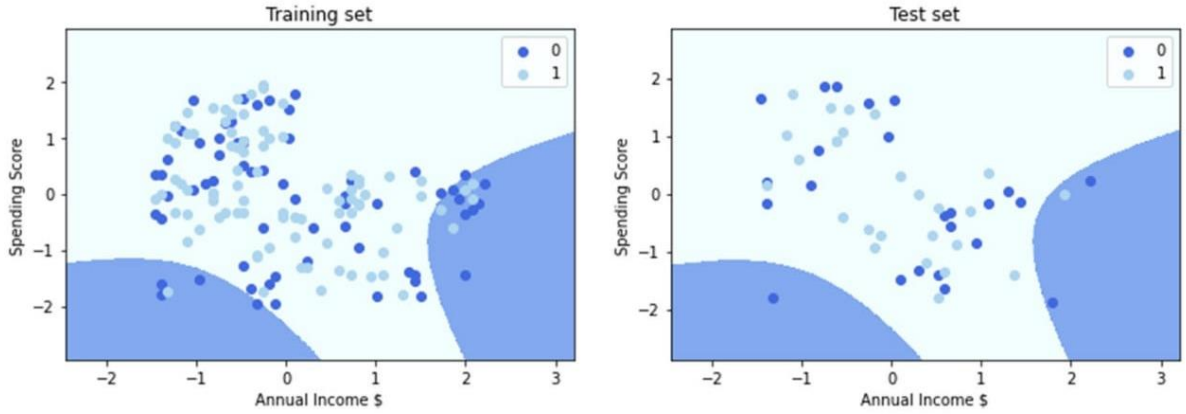


Le noyau polynomial donne de mauvaises performances. Il peut s'agir d'un surajustement de l'ensemble d'entraînement.

Exécuter SVM avec un noyau sigmoïde :

C=1

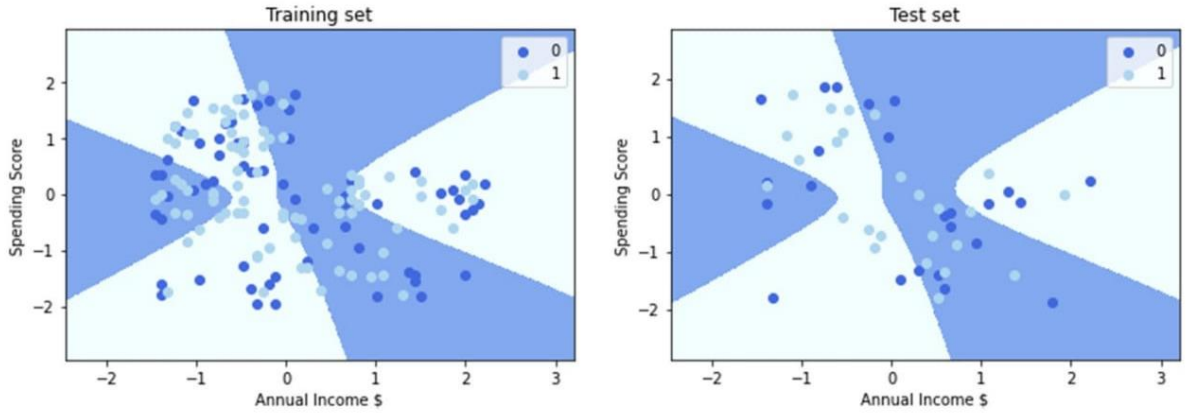
```
classifier = SVC(kernel='sigmoid',C=1)
```



Les Support Vecteurs et Accuracy Score :

```
69  =>  [ 0.73233274 -1.43637649]
70  =>  [ 0.09776971 -0.76282738]
71  =>  [-1.24186335  0.90123512]
72  =>  [ 0.73233274 -0.32700149]
73  =>  [-0.0432443   1.61440477]
74  =>  [-0.1842583   0.98047619]
75  =>  [ 0.94385375 -1.47599703]
76  =>  [-1.31237036  0.98047619]
77  =>  [-0.81882133  0.02958333]
78  =>  [ 1.29638877 -1.79296132]
79  =>  [-1.17135635  0.18806548]
80  =>  [ 0.23878372 -1.31751489]
81  =>  [-0.60730032  0.86161459]
Model accuracy score :0.6400
```

C=100



Nous pouvons voir que le noyau sigmoïde fonctionne également mal, tout comme avec le noyau polynomial.

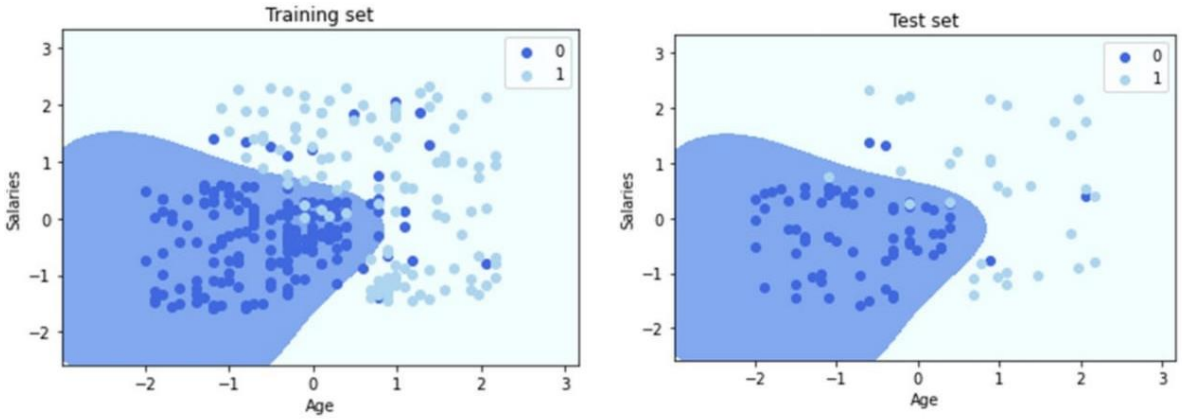
Jeu de données 2 : Employées

On a choisi d'utiliser l'ensemble de données Employees.csv qui contient des informations concernant les employées d'une entreprise : leur ID, âge, salaire et leur situation familiale . Et on a choisit d'expliquer la situation familiale par rapport a l'âge et le salaire des employées.

Exécuter SVM avec l'hyperparamètre par défaut :

L'hyperparamètre par défaut signifie

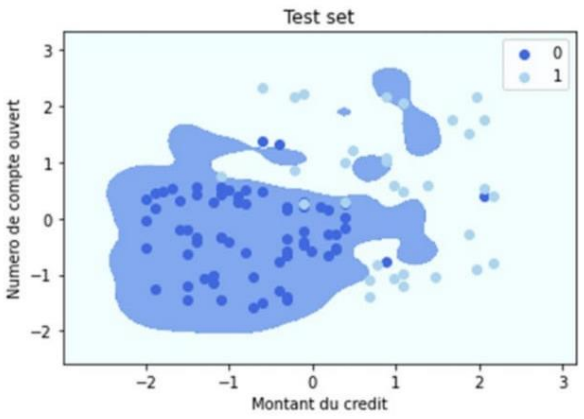
- **C=1** et **kernel='rbf'** et **gamma=auto**



Les Support Vecteurs et Accuracy Score :

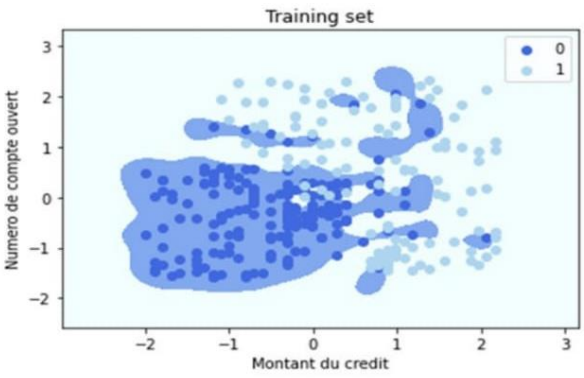
```
75  =>  [0.97777845 0.12805305]
76  =>  [1.47293972 0.07006676]
77  =>  [-0.60673761  1.37475825]
78  =>  [ 0.68068169 -1.37959044]
79  =>  [1.07681071 0.56295021]
80  =>  [0.08648817 1.05583366]
81  =>  [-1.10189888  1.95462113]
82  =>  [ 0.8787462 -0.5677824]
83  =>  [-0.11157634  0.67892279]
84  =>  [ 2.1661655  -0.68375498]
85  =>  [1.17584296 0.53395707]
86  =>  [1.07681071 0.53395707]
87  =>  [1.37390747 2.331532  ]
Model accuracy score :0.9300
```

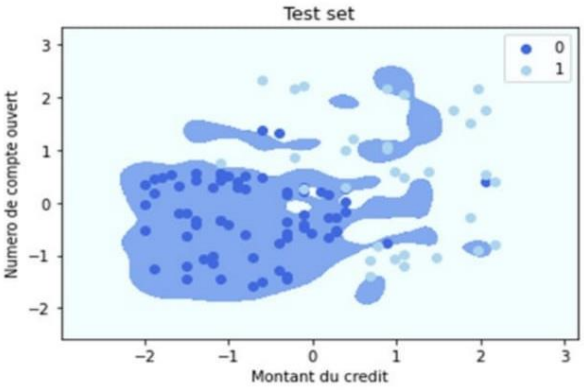
C=10 et gamma=10



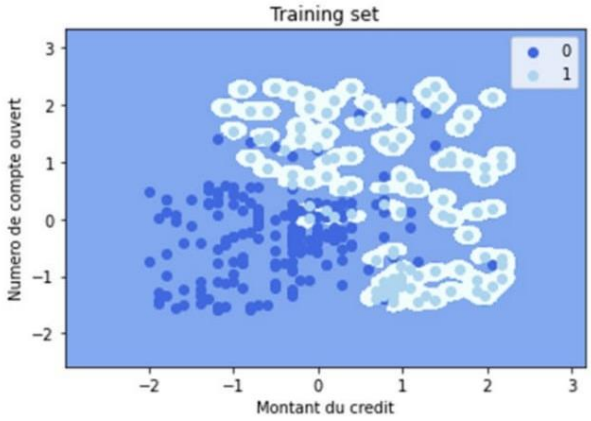

```
132  =>  [1.67100423 1.6067034 ]
133  =>  [ 1.96810099 -0.65476184]
134  =>  [1.47293972 0.35999821]
135  =>  [ 2.1661655  -1.03167271]
136  =>  [0.08648817 1.05583366]
137  =>  [1.57197197 1.11381995]
138  =>  [0.08648817 1.8676417 ]
139  =>  [0.08648817 1.8676417 ]
140  =>  [-1.10189888  1.95462113]
141  =>  [0.38358493 1.11381995]
142  =>  [ 0.8787462  -1.43757673]
143  =>  [ 1.86906873 -1.06066585]
144  =>  [ 1.07681071 -0.88670699]
Model accuracy score :0.8900
```

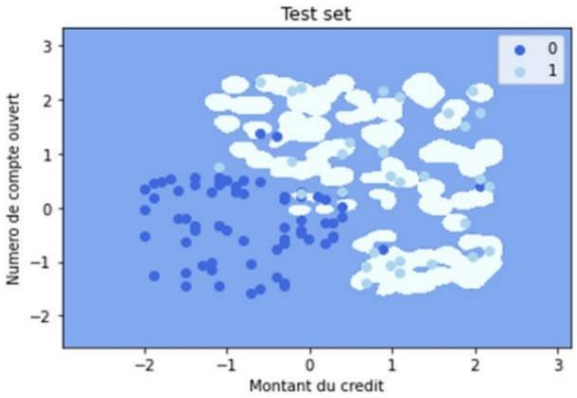
C=100 et gamma=10





C=1000 et gamma=10

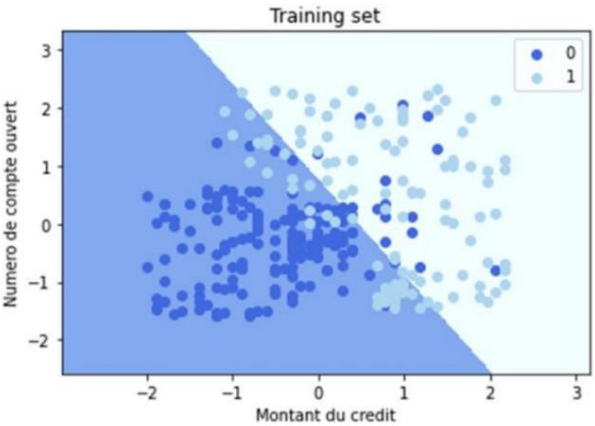


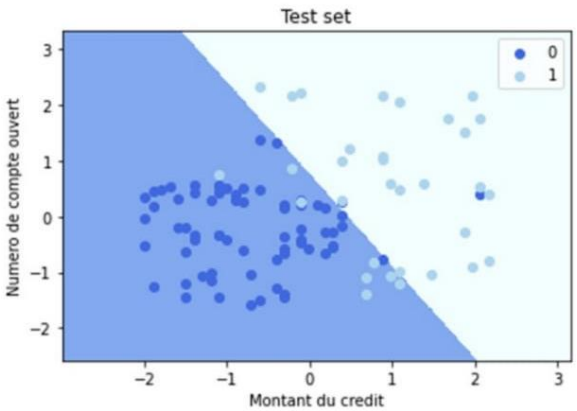



```
247  =>  [0.08648817 1.8676417 ]
248  =>  [-1.10189888  1.95462113]
249  =>  [-0.11157634  0.67892279]
250  =>  [ 2.1661655  -0.68375498]
251  =>  [1.77003648 0.99784738]
252  =>  [0.38358493 1.11381995]
253  =>  [ 0.8787462  -1.43757673]
254  =>  [ 1.86906873 -1.06066585]
255  =>  [ 1.07681071 -0.88670699]
256  =>  [ 1.67100423 -0.88670699]
257  =>  [1.17584296 0.53395707]
258  =>  [1.07681071 0.53395707]
259  =>  [1.37390747 2.331532  ]
Model accuracy score :0.8200
```

Exécuter SVM avec noyau linéaire :

C=1

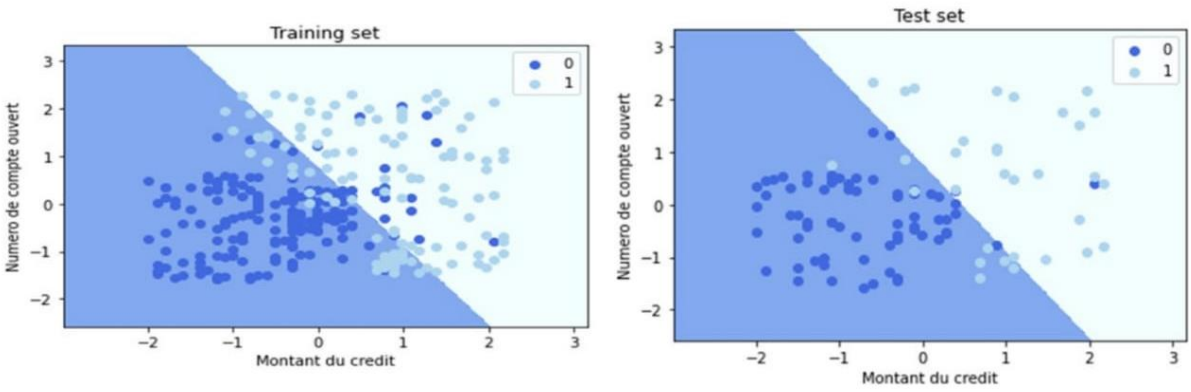




```
117  =>  [ 1.57197197 -1.26361786]
118  =>  [-0.80480212  1.89663484]
119  =>  [ 1.17584296 -0.97368642]
120  =>  [0.08648817 1.05583366]
121  =>  [-1.10189888  1.95462113]
122  =>  [ 0.8787462 -0.5677824]
123  =>  [-0.11157634  0.67892279]
124  =>  [0.38358493 1.11381995]
125  =>  [ 0.8787462  -1.43757673]
126  =>  [ 1.07681071 -0.88670699]
127  =>  [ 0.97777845 -1.14764529]
Model accuracy score :0.9000
```

```
function of hyperplan :  y = -0.828 + 1.675 * x1 + 1.031 * x2
```

C=100

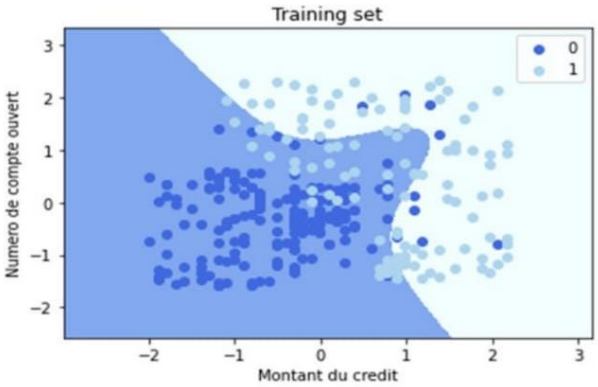


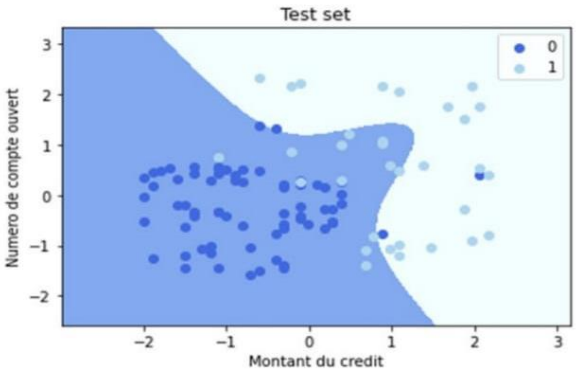

```
112  =>  [ 0.68068169 -1.37959044]
113  =>  [0.08648817 1.51972397]
114  =>  [ 1.57197197 -1.26361786]
115  =>  [-0.80480212  1.89663484]
116  =>  [ 1.17584296 -0.97368642]
117  =>  [0.08648817 1.05583366]
118  =>  [-1.10189888  1.95462113]
119  =>  [ 0.8787462 -0.5677824]
120  =>  [-0.11157634  0.67892279]
121  =>  [0.38358493 1.11381995]
122  =>  [ 0.8787462  -1.43757673]
123  =>  [ 1.07681071 -0.88670699]
124  =>  [ 0.97777845 -1.14764529]
Model accuracy score :0.8900

 function of hyperplan :  y = -0.828 + 1.675 * x1 + 1.031 * x2
```

Exécuter SVM avec un noyau polynomial :

C=1



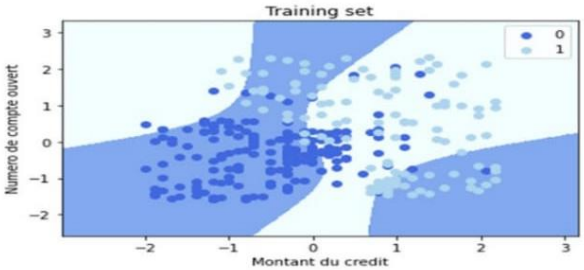


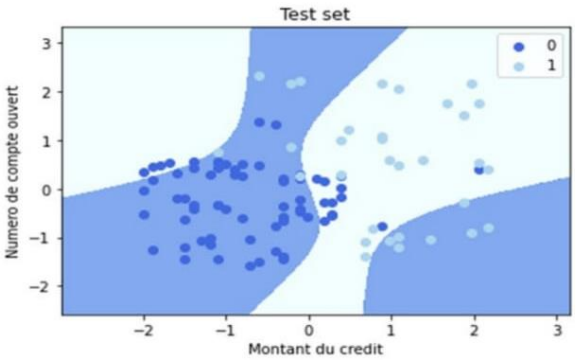

```
103  =>  [-0.80480212  1.89663484]
104  =>  [0.97777845 1.78066227]
105  =>  [1.07681071 0.56295021]
106  =>  [0.08648817 1.05583366]
107  =>  [1.57197197 1.11381995]
108  =>  [-1.10189888  1.95462113]
109  =>  [ 0.8787462 -0.5677824]
110  =>  [-0.11157634  0.67892279]
111  =>  [0.38358493 1.11381995]
112  =>  [ 0.8787462  -1.43757673]
113  =>  [1.17584296 0.53395707]
114  =>  [1.07681071 0.53395707]
115  =>  [ 0.97777845 -1.14764529]
Model accuracy score :0.8500
```

Le noyau polynomial donne de mauvaises performances. Il nous donne la même classification quelque soit la valeur de C

Exécuter SVM avec un noyau sigmoïde :

C=1





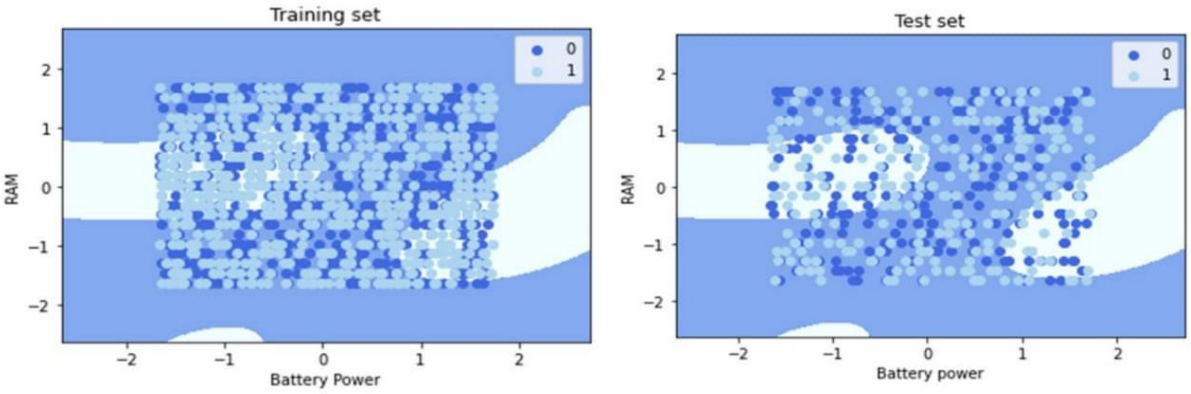
```
94  =>  [-0.80480212  1.89663484]
95  =>  [ 1.17584296 -0.97368642]
96  =>  [ 1.96810099 -0.65476184]
97  =>  [ 2.1661655  -1.03167271]
98  =>  [0.08648817 1.8676417 ]
99  =>  [0.08648817 1.8676417 ]
100  =>  [-0.11157634  0.67892279]
101  =>  [ 2.1661655  -0.68375498]
102  =>  [ 0.8787462  -1.43757673]
103  =>  [ 1.86906873 -1.06066585]
104  =>  [ 1.07681071 -0.88670699]
105  =>  [ 1.67100423 -0.88670699]
106  =>  [ 0.97777845 -1.14764529]
Model accuracy score :0.7000
```

Jeu de données 3 : Smartphones

On a choisi d'utiliser l'ensemble de données Smartphones.csv qui contient des informations concernant les smartphones : la capacite du batterie , la taille de la RAM , le prix , la capacite d'affichage , l'existence de l'option NFC (near-field-communication) …

Exécuter SVM avec l'hyperparamètre par défaut :

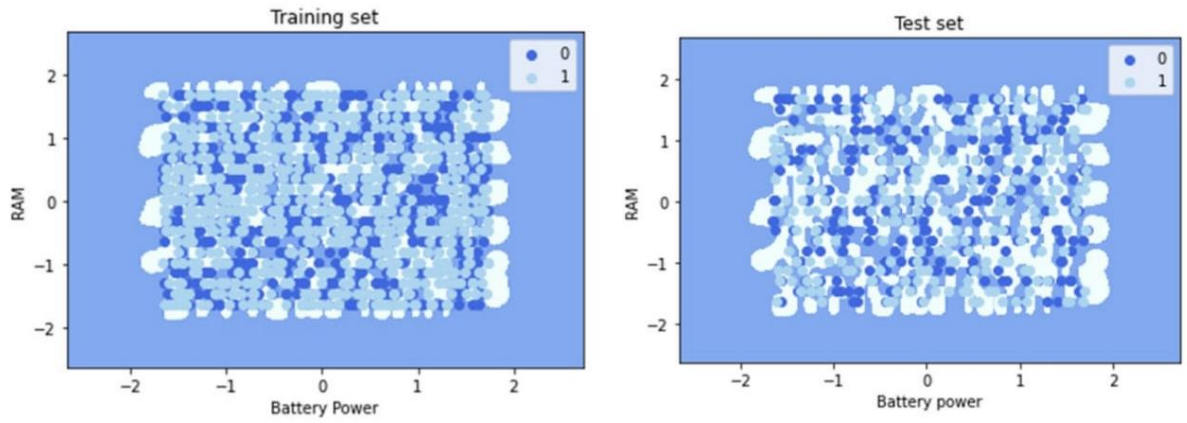
- **C=1** et **kernel='rbf'** et **gamma=auto**



Les Support Vecteurs et Accuracy Score :

```
1432  =>  [0.11974884 1.35641417]
1433  =>  [-1.53521212  0.03458779]
1434  =>  [-0.33983676  0.19981609]
1435  =>  [ 1.50982549 -0.6263254 ]
1436  =>  [-0.91941269  1.19118587]
1437  =>  [1.4373785  1.19118587]
1438  =>  [-1.23410431 -1.28723859]
1439  =>  [1.36266754 0.53027268]
1440  =>  [0.7762997  1.19118587]
1441  =>  [ 0.58612635 -0.4610971 ]
1442  =>  [0.108429   1.52164247]
1443  =>  [ 0.90308193 -1.45246689]
1444  =>  [ 1.1498545  -1.61769519]
1445  =>  [-0.02061721  1.35641417]
Model accuracy score :0.4880
```

C=100 et gamma=100



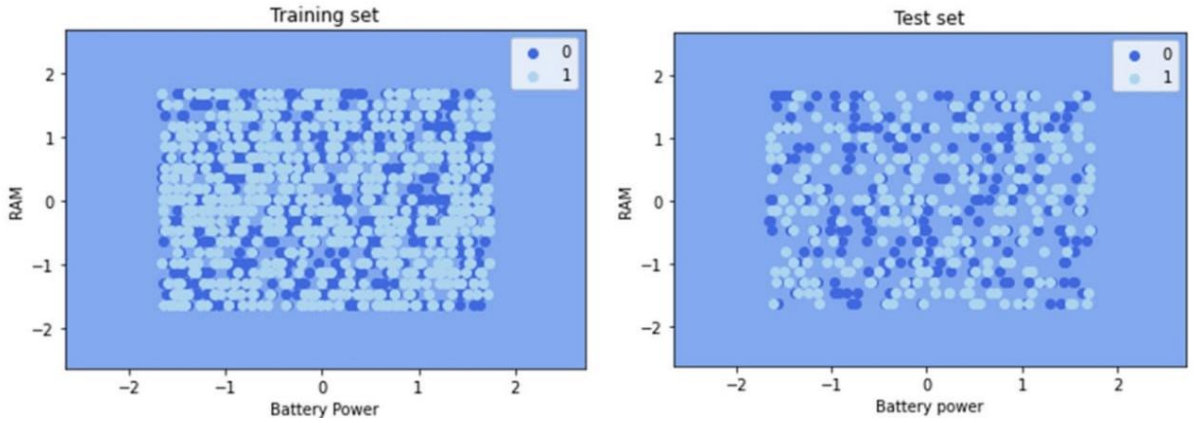
Les Support Vecteurs et Accuracy Score :

```
971  =>  [ 1.50982549 -0.13064051]
972  =>  [-1.12316986  1.02595758]
973  =>  [ 0.16502821 -0.13064051]
974  =>  [0.11974884 1.35641417]
975  =>  [ 1.50982549 -0.6263254 ]
976  =>  [-0.91941269  1.19118587]
977  =>  [1.4373785  1.19118587]
978  =>  [1.36266754 0.53027268]
979  =>  [0.7762997  1.19118587]
980  =>  [ 0.58612635 -0.4610971 ]
981  =>  [0.108429   1.52164247]
982  =>  [ 0.90308193 -1.45246689]
983  =>  [ 1.1498545  -1.61769519]
984  =>  [-0.02061721  1.35641417]
Model accuracy score :0.4680
```

On remarque dans le cas d'un noyau rbf, même si on a augmenté C et gamma l'Accuracy Score=0.4680 ce qui nous mène à dire que plus le jeu de données est grande , la classification devient plus en plus non évidente.

Exécuter SVM avec noyau linéaire :

- **C=100**



Les Support Vecteurs et Accuracy Score :

```
1457  =>  [-1.53521212  0.03458779]
1458  =>  [-0.33983676  0.19981609]
1459  =>  [ 1.50982549 -0.6263254 ]
1460  =>  [-0.91941269  1.19118587]
1461  =>  [1.4373785  1.19118587]
1462  =>  [-1.23410431 -1.28723859]
1463  =>  [1.36266754 0.53027268]
1464  =>  [0.7762997  1.19118587]
1465  =>  [ 0.58612635 -0.4610971 ]
1466  =>  [0.108429   1.52164247]
1467  =>  [ 0.90308193 -1.45246689]
1468  =>  [ 1.1498545  -1.61769519]
1469  =>  [-0.02061721  1.35641417]
Model accuracy score :0.4840
```

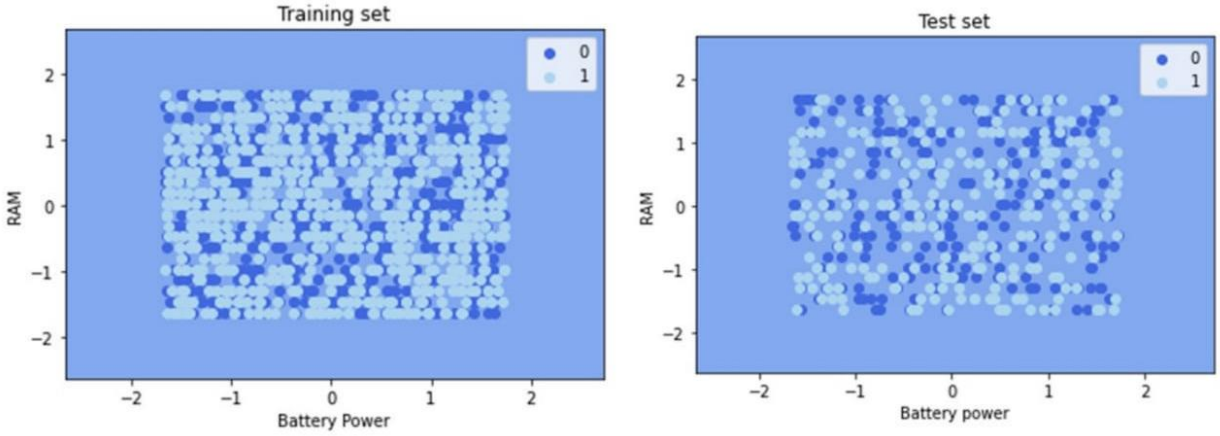
La fonction de l'hyperplan :

```
function of hyperplan :  y = -1.000 + -0.000 * x1 + -0.000 * x2
```

Le noyau linéaire donne de mauvaises performances.

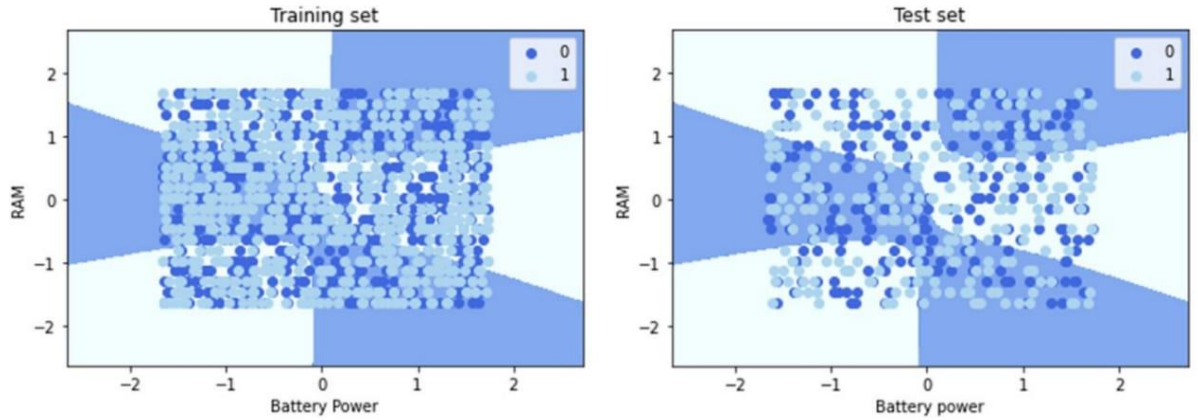
Exécuter SVM avec noyau polynomial :

C=10



De Même pour Le noyau polynomiale, il donne de mauvaises performances dans ce cas.

Exécuter SVM avec noyau sigmoïde :



Les Support Vecteurs et Accuracy Score :

```
795  =>  [0.81025923 1.02595758]
796  =>  [-0.10212007 -1.61769519]
797  =>  [-1.39937401 -0.2958688 ]
798  =>  [-0.60472107  0.03458779]
799  =>  [ 0.56122269 -1.61769519]
800  =>  [1.64792756 1.68687077]
801  =>  [0.11974884 1.35641417]
802  =>  [-1.53521212  0.03458779]
803  =>  [-0.33983676  0.19981609]
804  =>  [1.4373785  1.19118587]
805  =>  [0.7762997  1.19118587]
806  =>  [0.108429   1.52164247]
807  =>  [ 0.90308193 -1.45246689]
808  =>  [ 1.1498545  -1.61769519]
Model accuracy score :0.4860
```

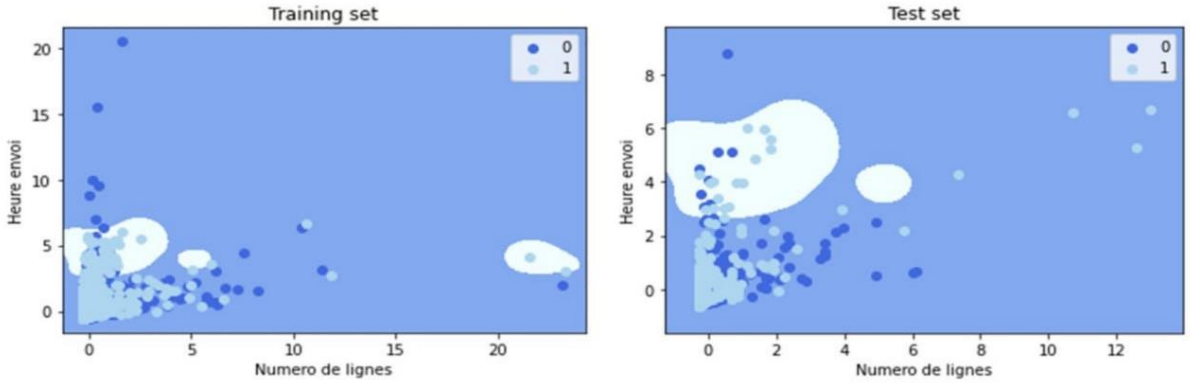
La classification est toujours faible pour un jeu de données aussi grand avec le noyau sigmoïde.

Jeu de données 4 : Email_Spam

On a choisi d'utiliser l'ensemble de données Email_Spam.csv qui contient des informations concernant des emails : ID d'email , Numero de lignes , heure d'envoi , information de spam ,

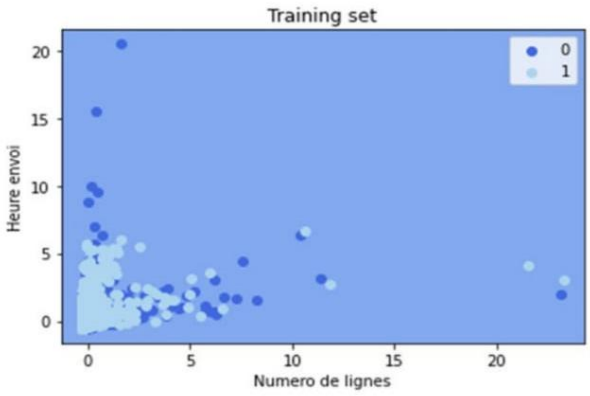
Exécuter SVM avec l'hyperparamètre par défaut :

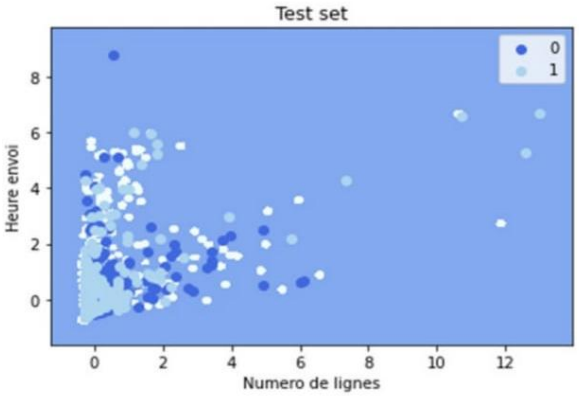
- **C=1** et **kernel='rbf'** et **gamma=auto**




```
2934  =>  [-0.22079941 -0.44959809]
2935  =>  [-0.28953232 -0.38262927]
2936  =>  [-0.28953232 -0.23753014]
2937  =>  [-0.28953232 -0.57237428]
2938  =>  [-0.28953232 -0.13707691]
2939  =>  [-0.15206649 -0.08126955]
2940  =>  [-0.28953232 -0.32682191]
2941  =>  [-0.28953232 -0.60585869]
2942  =>  [-0.28953232 -0.25985309]
2943  =>  [-0.28953232 -0.60585869]
2944  =>  [-0.01460065  0.14195987]
2945  =>  [-0.28953232 -0.52772839]
2946  =>  [-0.22079941  0.16428281]
Model accuracy score :0.6319
```

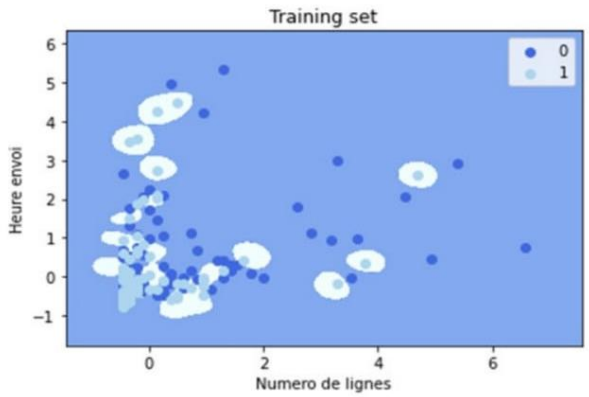
- **C=100 et gamma=100**

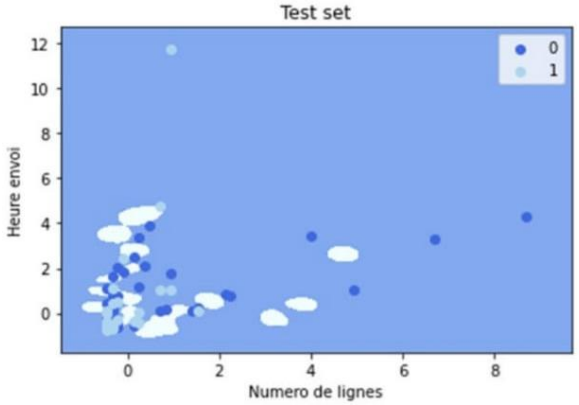





```
2889  =>  [-0.22079941 -0.44959809]
2890  =>  [-0.28953232 -0.38262927]
2891  =>  [-0.28953232 -0.23753014]
2892  =>  [-0.28953232 -0.57237428]
2893  =>  [-0.28953232 -0.13707691]
2894  =>  [-0.15206649 -0.08126955]
2895  =>  [-0.28953232 -0.32682191]
2896  =>  [-0.28953232 -0.60585869]
2897  =>  [-0.28953232 -0.25985309]
2898  =>  [-0.28953232 -0.60585869]
2899  =>  [-0.01460065  0.14195987]
2900  =>  [-0.28953232 -0.52772839]
2901  =>  [-0.22079941  0.16428281]
Model accuracy score :0.5994
```

- **C=10 et gamma=10**

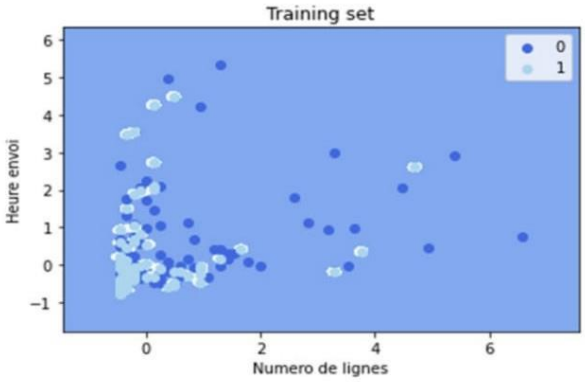


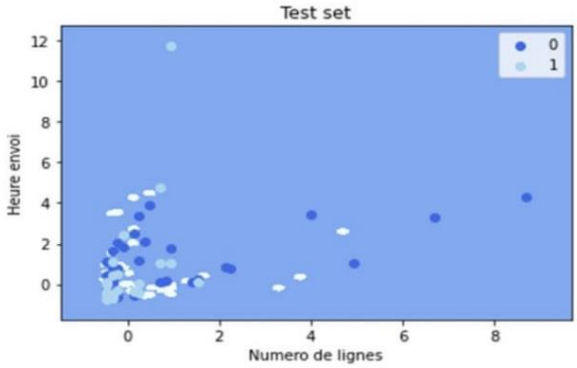


```
222  =>  [-0.45073574 -0.59233183]
223  =>  [-0.33376279 -0.38492927]
224  =>  [-0.33376279 -0.45900161]
225  =>  [ 0.95293957 -0.0441965 ]
226  =>  [-0.45073574 -0.74047651]
227  =>  [-0.33376279  3.48164699]
228  =>  [-0.09981691  0.82985714]
229  =>  [-0.45073574 -0.75529098]
230  =>  [ 3.2923984  -0.17752671]
231  =>  [ 0.95293957 -0.45900161]
232  =>  [-0.33376279 -0.63677524]
233  =>  [-0.45073574  0.60764011]
234  =>  [-0.45073574  0.05950478]
Model accuracy score :0.6100
```

On remarque que en augmentant la valeur de C et gamma , le modèle de classification devient de plus en plus efficace

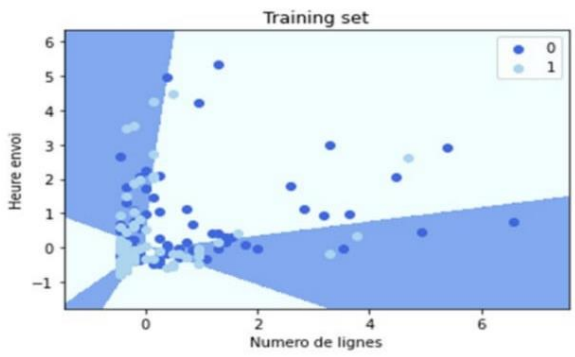
- **C=100 et gamma=100**

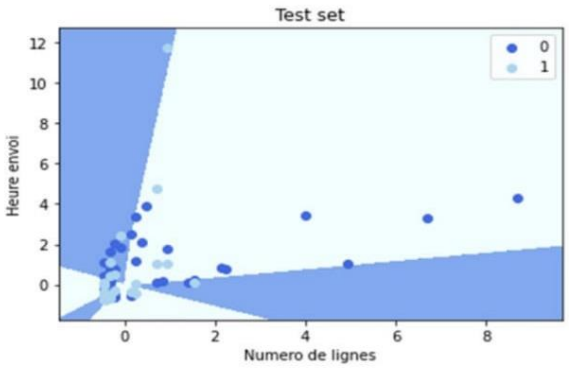




Exécuter SVM avec noyau sigmoïde :

- **C=1 000 et gamma=100**





```
154  =>  [-0.33376279 -0.54788842]
155  =>  [-0.21678985 -0.3701148 ]
156  =>  [-0.45073574 -0.6515897 ]
157  =>  [-0.45073574 -0.59233183]
158  =>  [-0.33376279 -0.38492927]
159  =>  [-0.33376279 -0.45900161]
160  =>  [ 0.95293957 -0.0441965 ]
161  =>  [-0.45073574 -0.74047651]
162  =>  [-0.33376279  3.48164699]
163  =>  [-0.09981691  0.82985714]
164  =>  [-0.45073574 -0.75529098]
165  =>  [ 3.2923984  -0.17752671]
166  =>  [ 0.95293957 -0.45900161]
167  =>  [-0.33376279 -0.63677524]
168  =>  [-0.45073574  0.60764011]
Model accuracy score :0.4500
```

Jeu de données 5 : Crédit

On a choisi d'utiliser l'ensemble de données Credit.csv qui contient des informations concernant les clients d'une banque qui ont fait une demande de crédits : leur ID ,

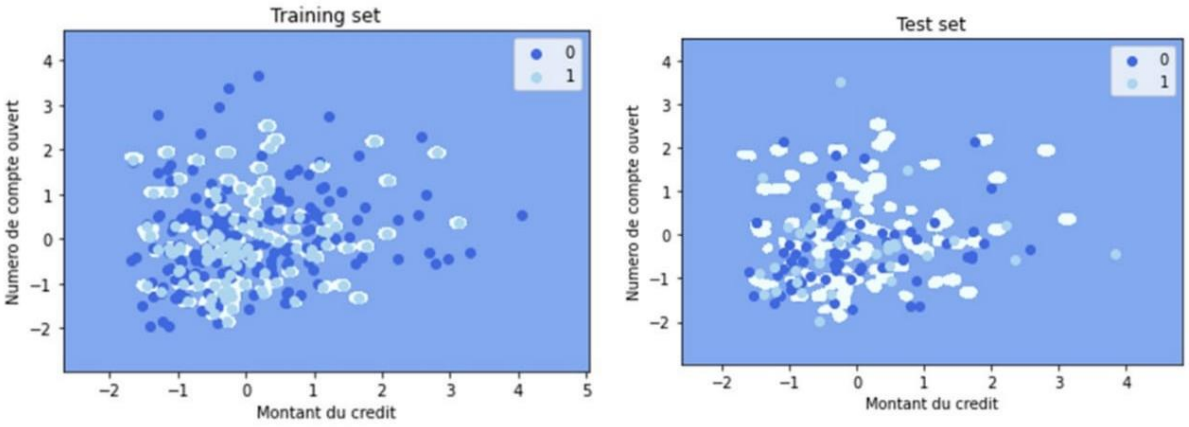
Exécuter SVM avec l'hyperparamètre par défaut :

- **C=1** et **kernel='rbf'** et **gamma=auto**

Les Support Vecteurs et Accuracy Score :

```
317  =>  [-0.14694493  0.15669686]
318  =>  [-0.3589002   0.16817282]
319  =>  [-0.45042978  0.06488911]
320  =>  [-0.50156532 -0.21053411]
321  =>  [-0.22077375  0.05341315]
322  =>  [-0.6076186  -1.33517895]
323  =>  [-0.57418382 -0.17610621]
324  =>  [1.36594066 0.44359605]
325  =>  [-0.31079044 -0.7499046 ]
326  =>  [-0.462079   -0.48595734]
327  =>  [-0.13060577 -0.53186121]
328  =>  [-0.3549667   0.29440847]
329  =>  [-0.370852   -0.76138056]
330  =>  [-0.20791422  0.40916815]
Model accuracy score :0.7208
```

C=100 et gamma=100

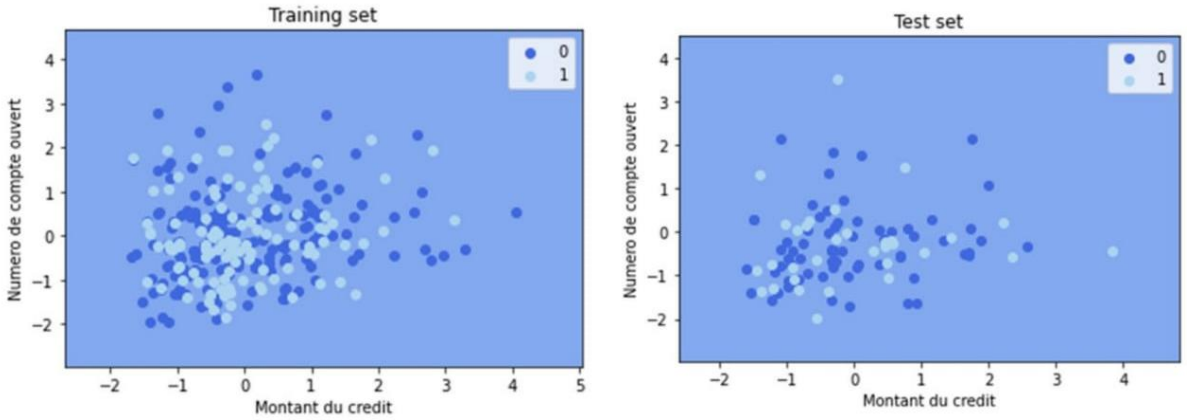


Les Support Vecteurs et Accuracy Score :

```
259  =>  [-0.7250901   1.75334221]
260  =>  [-1.46412566 -1.03744124]
261  =>  [ 0.08492175 -0.36225169]
262  =>  [-0.54075803 -1.4875676 ]
263  =>  [-1.65507341  1.76834642]
264  =>  [-1.14442111 -0.21220957]
265  =>  [-0.43711237  0.91310633]
266  =>  [-0.50366043 -0.36225169]
267  =>  [-0.9908733   1.34822848]
268  =>  [ 1.65587915 -1.32252127]
269  =>  [-0.2866412 -1.8626729]
270  =>  [ 0.70334562 -1.38253811]
271  =>  [ 0.94387541 -0.37725591]
Model accuracy score :0.5700
```

Exécuter SVM avec noyau linéaire :

- **C=10 / C=1000 et gamma=100**



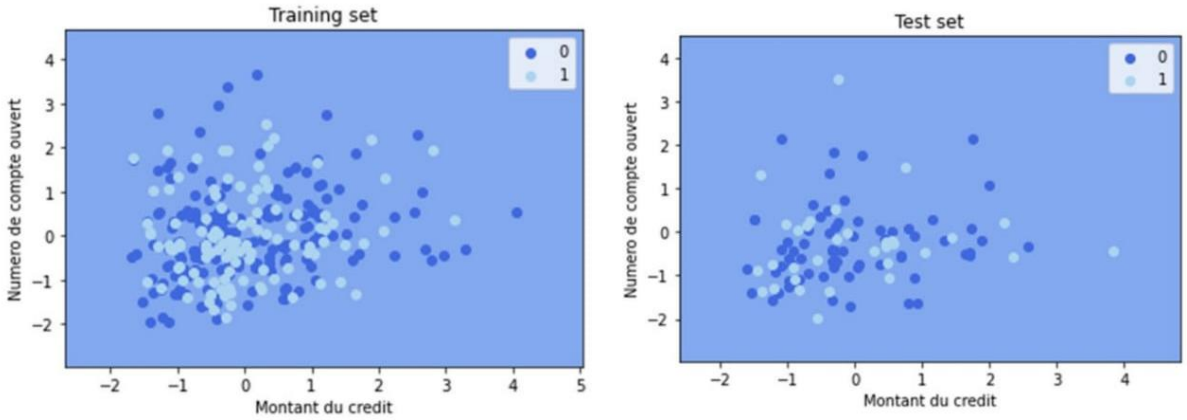

```
215  =>  [-1.24406536 -1.18748336]
216  =>  [-0.7250901   1.75334221]
217  =>  [-1.46412566 -1.03744124]
218  =>  [ 0.08492175 -0.36225169]
219  =>  [-0.54075803 -1.4875676 ]
220  =>  [-1.65507341  1.76834642]
221  =>  [-1.14442111 -0.21220957]
222  =>  [-0.43711237  0.91310633]
223  =>  [-0.50366043 -0.36225169]
224  =>  [-0.9908733   1.34822848]
225  =>  [ 1.65587915 -1.32252127]
226  =>  [-0.2866412 -1.8626729]
227  =>  [ 0.70334562 -1.38253811]
228  =>  [ 0.94387541 -0.37725591]
Model accuracy score :0.6800
```

Fonction de l'hyperplan :

```
function of hyperplan :  y = -1.000 + -0.000 * x1 + 0.000 * x2
```

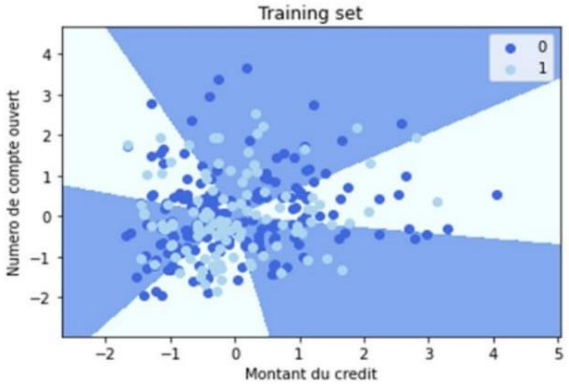
Exécuter SVM avec noyau polynomial :

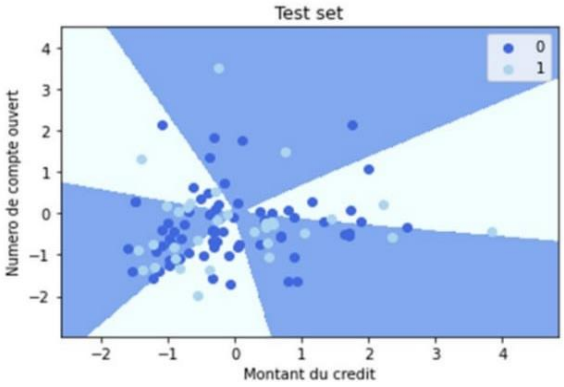
C=10




```
219  =>  [-0.7250901   1.75334221]
220  =>  [-1.46412566 -1.03744124]
221  =>  [ 0.08492175 -0.36225169]
222  =>  [-0.54075803 -1.4875676 ]
223  =>  [-1.65507341  1.76834642]
224  =>  [-1.14442111 -0.21220957]
225  =>  [-0.43711237  0.91310633]
226  =>  [-0.50366043 -0.36225169]
227  =>  [-0.9908733   1.34822848]
228  =>  [ 1.65587915 -1.32252127]
229  =>  [-0.2866412 -1.8626729]
230  =>  [ 0.70334562 -1.38253811]
231  =>  [ 0.94387541 -0.37725591]
Model accuracy score :0.6800
```

Exécuter SVM avec noyau sigmoïde :


Training set
Numero de compte ouvert
Montant du credit
0
1


Test set
Numero de compte ouvert
Montant du credit
0
1

Conclusion

Dans ce travail nous avons présenté les Support Vector Machines qui sont des modèles très intéressants dont les principaux points forts sont les suivants :

- L'efficacité dans les espaces à dimension élevés
- L'efficacité dans les cas où le nombre de dimensions est supérieur au nombre d'échantillons.
- L'utilisation d'un sous-ensemble de points d'apprentissage dans la fonction de décision (appelés vecteurs de support), ce qui la rend également efficace en termes de mémoire.
- La polyvalence/ flexibilité : différentes fonctions de noyau peuvent être spécifiées pour la fonction de décision.

Cependant, il faut faire attention au cas particulier où le nombre de « features » est beaucoup plus grand que le nombre d'échantillons, car un mauvais le choix des fonctions de noyau peut entrainer l'over-fitting.