TP7 : L'algorithme K-Means clustering

Exercice 1:

On souhaite créer un modèle K-Means pour regrouper (clustering) les clients qui ont des scores de dépenses presque similaires.

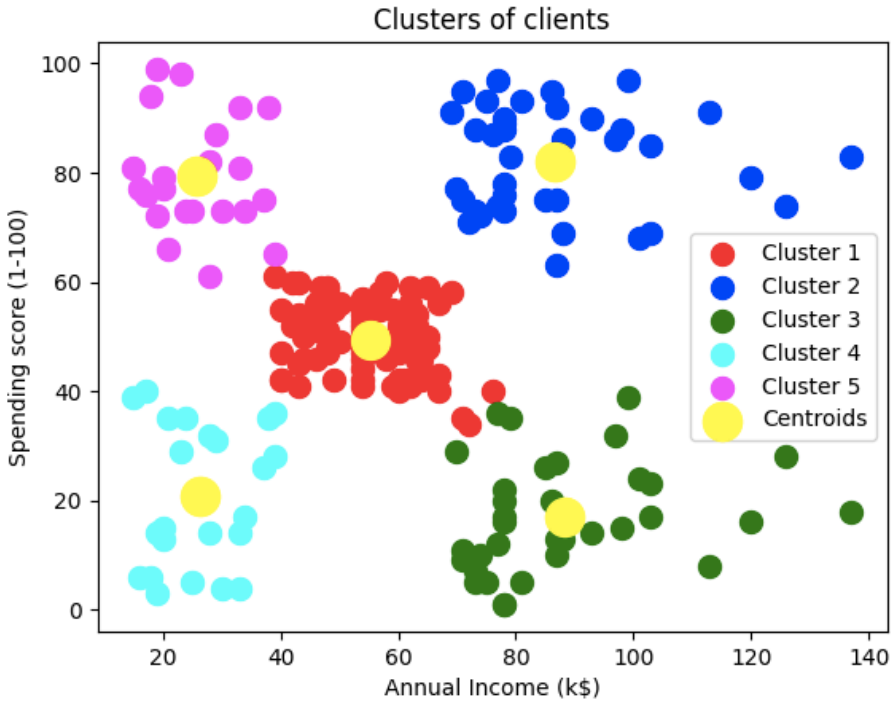
Nous allons utiliser le dataset « Mall_Customers.csv » qui contient 200 enregistrements avec les attributs 'CustomerID', 'Genre', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)'.

1. Importer les bibliothèques nécessaires : **numpy**, **pandas, matplotlib** et **sklearn**.
   La bibliothèque **sklearn** comprend la fonction suivante :
   - from sklearn.cluster import **KMeans**
2. Importer le dataset « Mall_Customers.csv ».
3. Visualiser les 5 premières et 5 dernières lignes du dataset.
4. Afficher les dimensions du dataset.
5. Préciser **les données d'entrée X**, correspondantes aux colonnes 'Annual Income (k$)' et 'Spending Score (1-100)'.
6. Afficher les données d'entrée **X**.

Nous allons utiliser la méthode Elbow pour trouver la valeur optimale de K.

7. Ecrire une fonction **elbow()** qui calcule les valeurs de WCSS pour des valeurs de K allant de 1 à 10. Utiliser les fonctions **KMeans()** et **fit()**. Préciser les paramètres suivants de la fonction **KMeans()**: n_clusters = K, init = 'k-means++', max_iter =300, n_init = 10, random_state = 0. Avec K, une valeur qui varie entre 1 et 10. Utiliser le paramètre **inertia_** pour retrouver la valeur de WCSS. Faire l'appel de la fonction **elbow()** pour afficher les valeurs de WCSS pour chaque valeur de K.
8. Visualiser le graphe des valeurs de WCSS en fonction de la valeur de K. Utiliser les fonctions **plot(), title(), xtable(), ytable(), et show()**.
9. Choisir la valeur optimale de K en analysant la **courbe d'Elbow**.
10. Créer un modèle K-Means avec la valeur optimale de K. Faire l'apprentissage du modèle avec la fonction **fit()** et la prédiction sur les données d'entrée **X** avec la fonction **predict().** Afficher les résultats de prédiction **y_pred1**.

11. Essayer cette fois-ci d'utiliser la fonction **fit_predict()** pour faire l'apprentissage du modèle K-Means et la prédiction avec les données **X**. Afficher les résultats de prédiction **y_pred2**. Qu'est-ce que vous remarquez ?
12. Visualiser les point de données sous forme de cluster en utilisant la fonction **scatter()**. Utiliser les paramètres suivants : X[y_pred2 == cluster$_i$, 0], X[y_pred2 == cluster$_i$,1],s = 100, c='red', label = 'cluster$_i$'. La variable cluster$_i$ prend les valeurs 0, 1, 2, 3 et 4.
13. Visualiser les centroids aussi. Utiliser la fonction **scatter()** avec les paramètres suivants: kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s = 300, c = 'yellow', label = 'Centroids'.
14. Utiliser les fonctions **legend(), title(), xlabel(), ylabel() et show()** pour visualiser la figure. La figure devrait ressembler à ça :



Exercice 2:

On souhaite créer modèle K-Means pour regrouper les personnes qui ont des revenus (incomes) presque similaires.

Nous allons utiliser le dataset « income.csv » qui contient 22 enregistrements avec les attributs 'Name', 'Age', et 'Income($)'.

1. Importer les bibliothèques nécessaires : **pandas, matplotlib** et **sklearn**.

La bibliothèque **sklearn** comprend la fonction suivante :

from sklearn.cluster import **KMeans**

Ajouter la fonction **MinMaxScaler()** pour faire la normalisation des données :

from sklearn.preprocessing import **MinMaxScaler**

2. Importer le dataset « income.csv ».
3. Visualiser les 22 lignes du dataset.
4. Afficher les dimensions du dataset.
5. Visualiser les données du dataset sur une figure. L'axe des abscisses correspond à 'Age' et l'axe des ordonnées à 'Income($)'. Utiliser les fonctions plot et scatter.

Nous allons utiliser la méthode Elbow pour trouver la valeur optimale de K.

6. Utiliser la fonction elbow() précédente (Exercice 1) pour calculer les valeurs de WCSS pour des valeurs de K allant de 1 à 10. Afficher les valeurs de WCSS pour chaque valeur de K.
7. Visualiser sur un graphe des valeurs de WCSS en fonction de la valeur de K. Utiliser les fonctions **plot(), title(), xtable(), ytable(), et show()**.
8. Choisir la valeur optimale de K en analysant la **courbe d'Elbow**.
9. Créer un modèle K-Means avec la valeur optimale de K.
10. Préciser les données d'entrée **X** correspondantes à "Age", et "Income($)".
11. Réaliser l'apprentissage et la prédiction du modèle avec la fonction **fit_predict()** et les données **X**. Afficher les résultats de prédiction **y_pred**.
12. Ajouter une colonne "Cluster" au dataset. Cette nouvelle colonne contient les valeurs prédites **y_pred**. Afficher le dataset après modification.
13. Afficher les données du dataset sur une figure où les données de chaque classe sont représentées par une couleur différente. Utiliser les instructions suivantes :

    df_i = df[df["Cluster"] == $cluster_i$]

    ax = df_i.plot.scatter("Age", "Income($)", color="blue")

    Avec : $cluster_i$ prend les valeurs 0, 1 et 2

    Le i dans df peut prendre les valeurs 0, 1 et 2
14. Transformez les caractéristiques "Income($)" et "Age" en adaptant leurs valeurs à la plage [0, 1]. Utiliser les fonctions **MinMaxScaler(), fit() et transform()**. Afficher les données du dataset après transformation des données.

15. Créer un deuxième modèle K-Means avec le paramètre n_clusters=K. Avec K la valeur optimale.
16. Préciser les données d'entrée **X** correspondant à "Age", et "Income($)".
17. Réaliser l'apprentissage et la prédiction du modèle avec la fonction **fit_predict()** et les données **X**. Afficher les résultats de prédiction.
18. Ajouter une colonne "Cluster" au dataset. Cette nouvelle colonne contient les valeurs prédites. Afficher le dataset après modification.
19. Afficher les coordonnées des centroïdes. Utiliser la variable **cluster_centers_**.
20. Afficher les données du dataset sur une figure où les données de chaque classe sont représentées par une couleur différente. Utiliser les instructions suivantes :

    df_i = df[df["Cluster"] == $cluster_i$]

    ax = df_i.plot.scatter("Age", "Income($)", color="green")

    $cluster_i$ prend les valeurs 0, 1 et 2

    Le i dans df peut prendre les valeurs 0, 1 et 2
21. Afficher les centroïdes aussi. Utiliser l'instruction suivante :

plt.scatter(model.cluster_centers_[:,0], model.cluster_centers_[:,1], color="red", marker="*")

Exercice 3:

On souhaite créer un modèle K-Means pour regrouper les fleurs iris qui ont des caractéristiques qui se ressemblent.

Nous allons utiliser le dataset iris sur sklearn qui contient 150 enregistrements avec les attributs 'Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width', et 'Target'.

1. Importer les bibliothèques nécessaires : **pandas, numpy, matplotlib** et **sklearn**.

   La bibliothèque **sklearn** comprend les fonctions suivantes :

   from sklearn import **datasets**

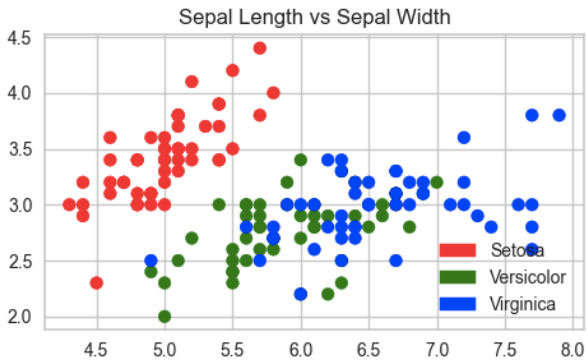
   from sklearn.cluster import **KMeans**

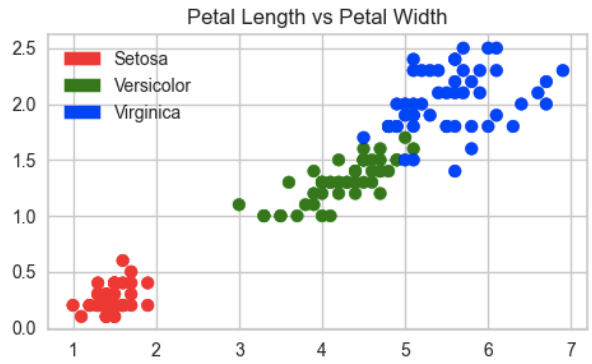
   import sklearn.metrics as **sm**
2. Importer aussi la bibliothèque **patches** de matplotlib:

   import matplotlib.patches as **mpatches**
3. Importer le dataset iris.
4. Faire la découverte du dataset avec les paramètres : data, target_names, et target.

5. Préciser l'entrée **X** correspondante aux données des colonnes 'Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width' et **y** à la colonne 'Target'.
6. Visualiser les 5 premières lignes de X.
7. Afficher les dimensions des données X et y.
8. Sur deux figures différentes, visualiser sur une première figure les données de 'Sepal Length' en fonction de 'Sepal Width'. Et sur la deuxième figure, visualiser les données de 'Petal Length' en fonction de 'Petal Width'. Utiliser les fonctions suivantes : **Patch** (de **matplotlib.patches** avec les paramètres **color** et **label**), **plot(), scatter(), title(), et legend()** avec le paramètre **handles**.

   Voilà à quoi devraient ressembler les figures résultats :

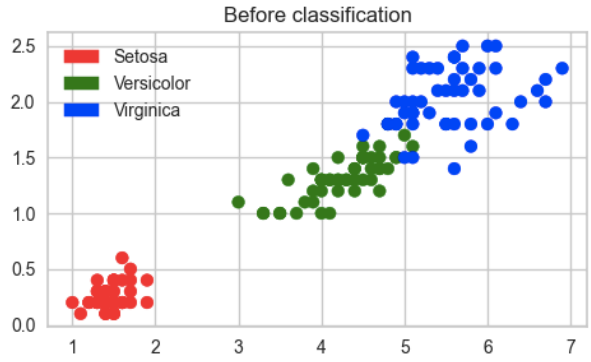


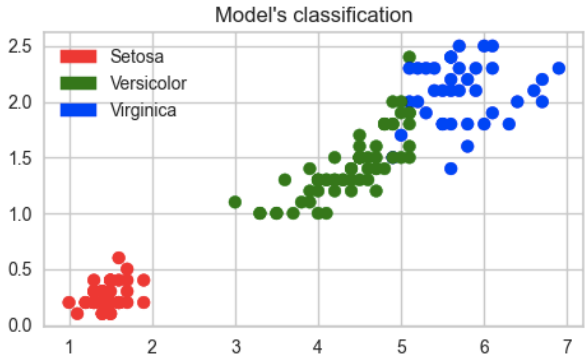


Partie 1 : Sélection de la valeur de K visuellement

9. Créer un modèle K-Means avec une valeur de K= 3.
10. Faire l'apprentissage de du modèle avec la fonction **fit()** et les données **X**.
11. Afficher les résultats de regroupement de votre modèle K-Means avec les variables : **labels_** et **cluster_centers_**. Analyser et commenter les résultats obtenus.
12. Cette fois-ci encore, sur deux figures différentes, visualiser sur une première figure les données de 'Petal Length' en fonction de 'Petal Width' en utilisant les données de classification réelles (Target attributs). Et sur la deuxième figure, visualiser les données de 'Petal Length' en fonction de 'Petal Width' en utilisant les données de classification du modèle K-Means.

    Les figures devraient ressembler à ça :





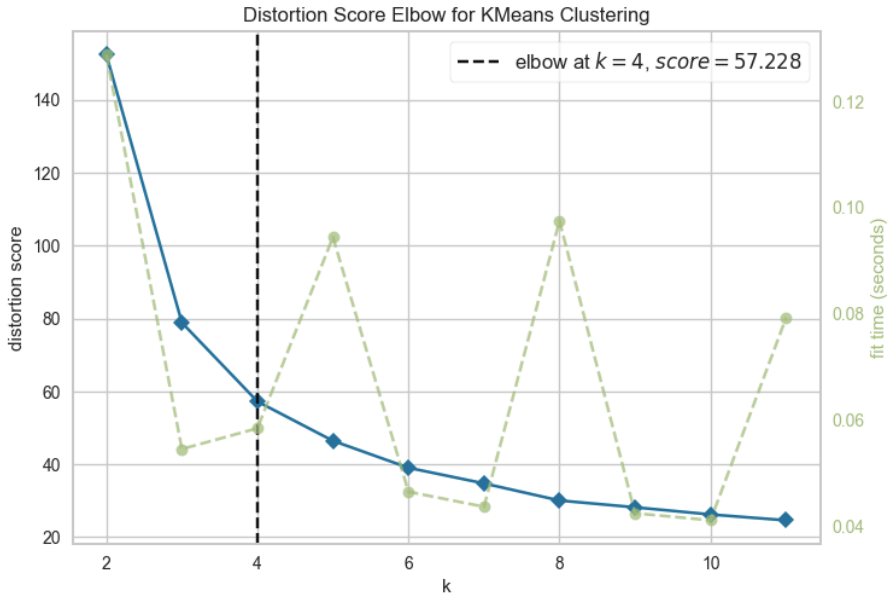
13. Qu'est-ce que vous remarquez ?
14. Calculer la précision (accuracy) et la matrice de confusion de votre modèle. Qu'est-ce que vous remarquez ?
15. A part le paramètre 'n_clusters' de la fonction KMeans, changer les valeurs des autres paramètres de la fonction càd **init**, **n_init**, **max_iter**, et **algorithm** et refaire l'apprentissage des différentes variations du modèle . Qu'est-ce que vous remarquez ?

Partie 2 : Sélection de la valeur de K avec la méthode elbow

16. Importer la fonction **KElbowVisualizer()** comme suit :

from yellowbrick.cluster import **KElbowVisualizer**

17. Chercher la documentation de la fonction **KElbowVisualizer()**.
18. Utiliser la fonction **KElbowVisualizer()** pour reconnaitre la valeur optimale de K. Utiliser le modèle K-Means précédent (de la partie 1) et les valeurs de K entre 2 et 12.
19. Faire l'apprentissage avec **fit()** pour calculer les valeurs de WCSS pour chaque valeur de K.
20. Afficher la courbe elbow sur une figure. La figure devrait ressembler à ça :



21. Quelle est la valeur optimale de K ?

22. Refaire les questions de la partie 1. Qu'est-ce que vous remarquez ?