

Compte rendu du projet ” Kmeans et KNN”

GANA Yassine MR1

le 07/03/2024



1 Composition du Projet

Le projet comprend trois fichiers :

- Un fichier contenant le dataset utilisé.
- Un fichier contenant le code d’implémentation des algorithmes K-NN et K-means.
- Ce document PDF qui explique la démarche du travail.

2 Choix du Dataset

Le dataset utilisé dans ce projet est celui des passagers du Titanic, comprenant 892 items.

1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25	S	
3	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925	S	
5	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05	S	
7	6	0	3	Moran, Mr. James	male		0	0	330877	8.4583	Q	
8	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Paisson, Master. Gosta Leonard	male	2	3	1	349909	21.075	S	
10	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333	S	
11	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708	C	

Figure 1: Capture d’écran du dataset

3 Démarche du Travail

La démarche du travail s'est déroulée comme suit :

1. Installation des bibliothèques nécessaires et l'importation des fonctionnalités qui vont être utilisées.
2. Phase de prétraitement des données, comprenant le nettoyage, l'encodage et la normalisation.
3. Exécution des algorithmes K-means et K-NN avec en changeant le paramètre k.

3.1 Code d'Importation

La première étape était l'installation des bibliothèques nécessaires l'importation des fonctionnalités qu'on va les utiliser.

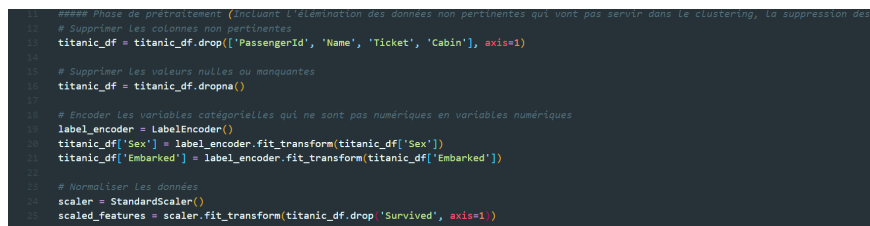


```
K-means_K-NN.py X
K-means_K-NN.py >
import pandas as pd # Pour manipuler les données
from sklearn.preprocessing import LabelEncoder, StandardScaler # Des outils nécessaires pour la phase de prétraitement
from sklearn.cluster import KMeans # Utiliser K-means pour faire le clustering
from sklearn.neighbors import KNeighborsClassifier # Utiliser K-NN pour faire la classification
from sklearn.metrics import silhouette_score # Utiliser cette fonction qui va attribuer un score à la qualité des clusters obtenus
from sklearn.model_selection import train_test_split # Pour faire la séparation en données de train et données de test
```

Figure 2: (Capture du code de l'importation des fonctionnalités nécessaires)

3.2 Phase de Prétraitement

Dans la phase de prétraitement on a fait les tâches suivantes pour qu'on puisse avoir des données manipulables. (nettoyage, encodage, normalisation...)



```
1. ##### Phase de prétraitement (incluant l'élimination des données non pertinentes qui vont pas servir dans le clustering, la suppression des
12 # Supprimer les colonnes non pertinentes
13 titanic_df = titanic_df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)
14
15 # Supprimer les valeurs nulles ou manquantes
16 titanic_df = titanic_df.dropna()
17
18 # Encoder les variables catégorielles qui ne sont pas numériques en variables numériques
19 label_encoder = LabelEncoder()
20 titanic_df['Sex'] = label_encoder.fit_transform(titanic_df['Sex'])
21 titanic_df['Embarked'] = label_encoder.fit_transform(titanic_df['Embarked'])
22
23 # Normaliser les données
24 scaler = StandardScaler()
25 scaled_features = scaler.fit_transform(titanic_df.drop('Survived', axis=1))
```

Figure 3: (Capture de la phase de prétraitement)

3.3 Tournage des Algorithmes et résultats obtenus

3.3.1 kNN

Code d'exécution de l'algorithme kNN. (Capture du code d'exécution)

```
86 ##### K-NN
87 # Choix de la valeur de K
88 NeighborsValue = 5
89
90 # KNN classification
91 knn = KNeighborsClassifier(n_neighbors=NeighborsValue)
92 knn.fit(X_train, y_train)
93
94 # Prédire Les étiquettes de classe pour Les données de test
95 knn_pred = knn.predict(X_test)
```

Figure 4: (Capture du code de l'algorithme KNN)

Résultat du KNN (cette figure ne montre pas la répartition entière, c'est juste une capture écran, vous pouvez trouver tout l'affichage en exécutant le code dans le console)

Résultats K-NN:	
Survivants	Non-Survivants
Passager 0	Passager 2
Passager 1	Passager 4
Passager 3	Passager 5
Passager 7	Passager 6
Passager 9	Passager 8
Passager 16	Passager 10
Passager 18	Passager 11
Passager 19	Passager 12
Passager 21	Passager 13
Passager 22	Passager 14
Passager 25	Passager 15
Passager 29	Passager 17
Passager 30	Passager 20
Passager 34	Passager 23
Passager 37	Passager 24
Passager 38	Passager 26
Passager 40	Passager 27
Passager 47	Passager 28
Passager 52	Passager 31
Passager 53	Passager 32
Passager 55	Passager 33
Passager 60	Passager 35
Passager 63	Passager 36
Passager 68	Passager 39
Passager 70	Passager 41
Passager 74	Passager 42
Passager 79	Passager 43
Passager 81	Passager 44
Passager 82	Passager 45
Passager 86	Passager 46

Figure 5: (Capture du résultat de l'algorithme KNN)

3.3.2 k-means avec $k = 2$

Code d'exécution de l'algorithme k-means avec $k = 2$. (Capture du code d'exécution)

```
63 # K-means avec K=2
64 KValue = 2
65 kmeans = KMeans(n_clusters=KValue, random_state=42)
66 kmeans.fit(X_train)
67
68 # Prédire Les clusters pour les données de test
69 kmeans_pred = kmeans.predict(X_test)
70
71 print("Résultats K-means (K=2):")
72 afficher_tableau_kmeans(kmeans_pred)
```

Figure 6: (Capture du code de l'algorithme Kmeans avec k=2)

Résultat du Kmeans avec k=2 (cette figure ne montre pas la répartition entière, c'est juste une capture écran, vous pouvez trouver tout l'affichage en exécutant le code dans le console)

Résultats K-means (K=2):	
Cluster 1	Cluster 2
Passager 0	Passager 4
Passager 1	Passager 6
Passager 2	Passager 9
Passager 3	Passager 10
Passager 5	Passager 11
Passager 7	Passager 12
Passager 8	Passager 14
Passager 13	Passager 15
Passager 18	Passager 16
Passager 21	Passager 17
Passager 22	Passager 19
Passager 29	Passager 20
Passager 34	Passager 23
Passager 37	Passager 24
Passager 38	Passager 25
Passager 40	Passager 26
Passager 55	Passager 27
Passager 56	Passager 28
Passager 59	Passager 30
Passager 61	Passager 31
Passager 63	Passager 32
Passager 71	Passager 33
Passager 72	Passager 35
Passager 74	Passager 36
Passager 75	Passager 39
Passager 76	Passager 41

Figure 7: (Capture du résultat de l'algorithme Kmeans avec k=2)

3.3.3 k-means avec $k = 3$

Code d'exécution de l'algorithme k-means avec $k = 3$. (Capture du code d'exécution)

```
74 # K-means avec K=3
75 KValue = 3
76 kmeans = KMeans(n_clusters=KValue, random_state=42)
77 kmeans.fit(X_train)
78
79 # Prédire Les clusters pour Les données de test
80 kmeans_pred = kmeans.predict(X_test)
81
82 print("Résultats K-means (K=3):")
83 afficher_tableau_kmeans(kmeans_pred)
84
85
```

Figure 8: (Capture du code de l'algorithme Kmeans avec k=3)

Résultat du Kmeans avec k=3 (cette figure ne montre pas la répartition entière, c'est juste une capture écran, vous pouvez trouver tout l'affichage en exécutant le code dans le console)

Résultats K-means (K=3):		
Cluster 1	Cluster 2	Cluster 3
Passager 0	Passager 4	Passager 6
Passager 1	Passager 9	Passager 14
Passager 2	Passager 10	Passager 19
Passager 3	Passager 11	Passager 25
Passager 5	Passager 12	Passager 26
Passager 7	Passager 15	Passager 29
Passager 8	Passager 16	Passager 36
Passager 13	Passager 17	Passager 37
Passager 18	Passager 20	Passager 41
Passager 21	Passager 23	Passager 47
Passager 22	Passager 24	Passager 57
Passager 34	Passager 27	Passager 60
Passager 38	Passager 28	Passager 67
Passager 40	Passager 30	Passager 68
Passager 55	Passager 31	Passager 70
Passager 56	Passager 32	Passager 79
Passager 59	Passager 33	Passager 86
Passager 61	Passager 35	Passager 93
Passager 63	Passager 39	Passager 104
Passager 71	Passager 42	Passager 115
Passager 72	Passager 43	Passager 121
Passager 74	Passager 44	Passager 124
Passager 75	Passager 45	Passager 125
Passager 76	Passager 46	Passager 128
Passager 78	Passager 48	Passager 130
Passager 81	Passager 49	Passager 132
Passager 82	Passager 50	Passager 137
Passager 85	Passager 51	
Passager 87	Passager 52	
Passager 94	Passager 53	

Figure 9: (Capture du résultat de l'algorithme Kmeans avec k=3)

4 Interprétation et Comparaison

Les résultats ont été analysés en fonction des métriques suivantes :

- Silhouette Score
- Accuracy

```
----- Résultats d'interprétation
Silhouette Score (K-means): 0.2489263817690846
Accuracy (KNN): 0.7692307692307693
PS C:\Users\yacin\OneDrive\Bureau\GANA_Yassine_Projet_K-means_K-NN>
```

Figure 10: Comparaison des résultats

4.1 Silhouette Score

Le Silhouette Score pour évaluer la cohésion de la séparation des clusters obtenus par K-means. On a obtenu la valeur de 0,2489. Généralement la valeur du silhouette varie entre 1 et -1. Une valeur proche de 1 indique que les points sont bien placés dans leurs clusters, une valeur proche de 0 indique qu'il peut y avoir un chevauchement de clusters, et une valeur proche -1 indique que les points peuvent être mal placés.

4.2 Accuracy

L'Accuracy est utilisée pour évaluer la performance du modèle. Cette métrique indique le pourcentage de prédiction correctes faites par le modèle par rapport au nombre total de l'échantillon.

4.3 Comparaison des deux algorithmes

En comparant les deux approches, on peut dire que K-means fournit une structure de clustering des données, tandis que KNN cherche à prédire les classes des échantillons individuels. K-means est un algorithme de clustering non supervisé, tandis que KNN est un algorithme de classification supervisé.