

Université Ibn Tofail
Ecole Nationale des Sciences Appliquées de Kénitra



Rapport du Mini-projet en Data Mining et Machine Learning

Réalisé par
TRAORE Idrissa

Professeur
M. Omar OUSTOUS

Année Universitaire 2021/2022

Remerciements

Avant de commencer la présentation de ce travail, il nous paraît naturel d'exprimer notre grande reconnaissance à nos parents, aux personnes qui nous ont confié ce travail et ceux qui nous ont permis de le réaliser.

Nous remercions donc les professeurs de l'Ecole Nationale des Sciences Appliquées de Kénitra en l'occurrence Mr Omar OUSTOUS qui nous a enseigné le module de Machine Learning ainsi que Mr Ayoub AIT LAHCEN qui est le responsable de ce master.

Table des matières

I.	Cas d'étude 1 : déterminants de l'état de pauvreté	4
1.	Importation du dataset	4
2.	Extraction des variables à utiliser et encodage des variables catégorielles.....	4
3.	3-Répartition de la base de données en sous base de données (Entrainement/test).....	5
4.	Modèle de régression logistique	6
5.	Formation de la matrice de confusion et calculer l'accuracy et l'AUC du modèle pour les 2 seuils 0.5 (par défaut) et 0.8.....	7
II.	Cas d'étude 2 : des clients d'un centre commercial.....	7
1.	Importation du dataset	7
2.	Détermination du k-means	8
3.	Entrainement du modèle de clustering sur les deux variables	9
4.	Visualisation des clusters sur un graphique	9
III.	Cas d'étude 3 : scolarité, expérience et revenus « Mincer 1974 »	9
1.	Importation du dataset	9
2.	Suppression des valeurs manquantes de la variables « revenu ».....	10
3.	Création des deux nouvelles variables age2 (carre de la variable age) et exper2 (carre de la variable exper).....	11
4.	Répartition de la base de données en sous base de données (Entrainement/test).....	11
5.	Construction du modèle de régression linéaire multiple.....	11
6.	Prédiction des résultats Test	12
7.	Calcul du coefficient de détermination R carré.....	13

I. Cas d'étude 1 : déterminants de l'état de pauvreté

Nous avons fait appel aux bibliothèques ; numpy, qui est destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux et pandas, qui sert à la manipulation et à l'analyse des données

```
▼ Importation des libraries

import numpy as np
import pandas as pd
```

1. Importation du dataset

Le fichier impactpauv contient 4880 lignes, 24 colonnes avec 24 variables (de milieu à deppc: dépense par personne). Nous avons utilisé l'encodage 'ISO-8859-1' puis afficher X et y

```
1-Importation du dataset

[203] dataset = pd.read_csv('impactpauv.csv', encoding='ISO-8859-1')
      x = dataset.iloc[:, :-1].values
      y = dataset.iloc[:, -1].values

[204] print(X)

[[1.00000000e+00 5.00000000e+00 2.00000000e+00 ... 1.36500000e+03
 1.00000000e+00 0.00000000e+00]
 [1.00000000e+00 6.00000000e+00 2.00000000e+00 ... 1.16480005e+03
 2.00000000e+00 0.00000000e+00]
 [1.00000000e+00 4.00000000e+00 2.00000000e+00 ... 1.36500000e+03
 3.00000000e+00 0.00000000e+00]
 ...
 [2.00000000e+00 9.00000000e+00 2.00000000e+00 ... 1.58469995e+03
 4.87800000e+03 0.00000000e+00]
 [2.00000000e+00 1.30000000e+01 2.00000000e+00 ... 1.15959998e+03
 4.87900000e+03 0.00000000e+00]
 [2.00000000e+00 5.00000000e+00 2.00000000e+00 ... 1.58469995e+03
 4.88000000e+03 0.00000000e+00]]

[205] print(y)

[11684.40039062  8310.        20935.75      ...  3856.66674805
 828.53845215  6542.20019531]
```

2. Extraction des variables à utiliser et encodage des variables catégorielles

Nous avons créé une liste qui contient les variables à utiliser puis encoder les variables catégorielles

```
▼ 2-Encodage des variables catégorielles

▼ Extraction des variables à utiliser

[206] df = dataset[['milieu', 'taille', 'sexecm', 'agecm', 'occup_cm', 'emcm', 'typlog', 'stocp', 'pauvre']]
      X2 = df.iloc[:, :-1].values
      y2 = df.iloc[:, -1].values

[207] print(X2)

[[ 1  5  1 ...  1  1  3]
 [ 1  6  1 ...  1  1  3]
 [ 1  4  1 ...  1  1  3]
 ...
 [ 2  9  1 ...  1  6  1]
 [ 2 13  1 ...  1  6  1]
 [ 2  5  1 ...  1  6  7]]

[208] print(y2)

[2 2 2 ... 2 2 2]
```

y2 représente la colonne contenant la variable pauvre.

Nous avons encodé les variables : milieu, taille, sexecm, agecm, occup_cm, emcm, typlog et stocp qui sont contenu dans x2 puis la variable pauvre.

```
▼ Encodage de la variable indépendante

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
X2 = np.array(ct.fit_transform(X2))

[ ] print(X2)

[[ 1.  0.  5. ...  1.  1.  3.]
 [ 1.  0.  6. ...  1.  1.  3.]
 [ 1.  0.  4. ...  1.  1.  3.]
 ...
 [ 0.  1.  9. ...  1.  6.  1.]
 [ 0.  1. 13. ...  1.  6.  1.]
 [ 0.  1.  5. ...  1.  6.  7.]]

▼ Encodage de la variable dépendante

[ ] from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y2 = le.fit_transform(y2)

[ ] print(y2)

[1 1 1 ... 1 1 1]
```

3. 3-Répartition de la base de données en sous base de données (Entrainement/test)

On a entraîné notre modèle puis afficher le X2_train

```
▼ 3-Répartition de la base de données en sous base de données (Entrainement/test)

from sklearn.model_selection import train_test_split
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size = 0.2, random_state = 0)
print(X2_train)

[[1.  0.  8. ...  1.  3.  3.]
 [1.  0.  5. ...  1.  0.  3.]
 [1.  0.  3. ...  1.  2.  1.]
 ...
 [1.  0.  4. ...  1.  3.  1.]
 [1.  0.  4. ...  1.  3.  1.]
 [1.  0.  4. ...  1.  2.  1.]]
```

4. Modèle de régression logistique

Pour éviter que notre modèle soit mal fait et étant donné que les valeurs sont différentes et n'ont pas les mêmes unités non plus, nous avons donc procédé une mise à échelle. D'où les valeurs proches de 0 au résultat du `x2_train`.

```
▼ 4-Modele de regression logistique

▼ Mise à echelle

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X2_train = sc.fit_transform(X2_train)
X2_test = sc.transform(X2_test)
print(X2_train)

[[ 0.87687917 -0.87687917  0.72869297 ...  0.44693864 -0.65699767
  0.39598906]
 [ 0.87687917 -0.87687917 -0.32820512 ...  0.44693864 -2.17770057
  0.39598906]
 [ 0.87687917 -0.87687917 -1.03280385 ...  0.44693864 -1.16389863
 -0.54954079]
 ...
 [ 0.87687917 -0.87687917 -0.68050448 ...  0.44693864 -0.65699767
 -0.54954079]
 [ 0.87687917 -0.87687917 -0.68050448 ...  0.44693864 -0.65699767
 -0.54954079]
 [ 0.87687917 -0.87687917 -0.68050448 ...  0.44693864 -1.16389863
 -0.54954079]]
```

Nous voyons que l'intecept= 1.84 alors la probabilité d'avoir le résultat sera supérieur a 0.5

```
▼ Entraînement du modèle de la regression logistique

[215] from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X2_train, y2_train)
#classifier.get_params()
coef = classifier.fit(X2_train, y2_train).coef_
intecept = classifier.fit(X2_train, y2_train).intercept_
print(coef, intecept)

[[ 0.19915613 -0.19915613  0.05479153 -0.02322644  0.02650166 -0.02305094
 -0.03943628 -0.14115288 -0.0237873 ] [1.840833]]
```

5. Formation de la matrice de confusion et calculer l'accuracy et l'AUC du modèle pour les 2 seuils 0.5 (par défaut) et 0.8

```
5- Formation de la matrice de confusion et calculer l'accuracy et l'AUC du modèle
pour les 2 seuils 0.5 (par défaut) et 0.8

y2_pred = classifieur.predict_proba(X2_test)
#print(y2_pred)
y2_pred_df = pd.DataFrame(y2_pred[:,1])
#print(y2_pred_df)
y2_test_pred = y2_pred_df.applymap(lambda x: 1 if x>=0.3 else 0)
#print(np.concatenate((y2_test_pred, y2_test.reshape(len(y2_test),1)),1))

from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score
cm = confusion_matrix(y2_test, y2_test_pred)
print(cm)
print(accuracy_score(y2_test, y2_test_pred))
print(roc_auc_score(y2_test, y2_test_pred))

[[ 0 132]
 [ 0 844]]
0.8647540983606558
0.5
```

D'après le résultat donné par 'cm', on a 844 bons résultats et 132 mauvais résultats.

En termes de précision, l'accuracy_score nous donne un résultat de 86.47 % et AUC montre une qualité de prédiction du modèle de 50 %.

II. Cas d'étude 2 : des clients d'un centre commercial

Nous avons importé les mêmes bibliothèques que dans le cas 1. La bibliothèque **matplotlib.pyplot** permet de tracer et visualiser des données sous formes de graphiques.

```
Cas d'étude 2 : des clients d'un centre commercial

▼ Importation des bibliothèques

[1] import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

1. Importation du dataset

On a importé le dataset et affiché les deux variables qu'on veut utiliser dans un tableau nommé, à savoir les variables : revenu annuel et score de dépense.

The screenshot shows a Jupyter Notebook interface. On the left, a file explorer shows a folder named 'sample_data' containing a file 'Mall_Customers.csv'. The main area is titled 'Importation du dataset' and contains the following code:

```
[2] dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
print(X)
```

The output of the code is a list of 10 rows, each containing two values in brackets, representing the Annual Income (k\$) and Spending Score (1-100) for each customer. The values are: [76, 40], [76, 87], [77, 12], [77, 97], [77, 36], [77, 74], [78, 22], [78, 90], [78, 17], [78, 88], [78, 20], [78, 76], [78, 16], [78, 89], [78, 1], [78, 78], [78, 1], [78, 73], [79, 35], [79, 83], [81, 5], [81, 93], [85, 26], [85, 75], [86, 20].

On the right, a preview of the 'Mall_Customers.csv' file is shown, displaying columns: CustomerID, Genre, Age, Annual Income (k\$), and Spending Score (1-100). The first 10 rows of data are visible.

2. Détermination du k-means

D'après le graphe , le K-means est égal à 5, on aura donc 5 groupes de clients

The screenshot shows a Jupyter Notebook interface. The main area is titled 'Utilisation de la méthode Elbow pour préciser le nombre de clusters à définir pour l'algorithme'. It contains the following code:

```
[3] from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

The output of the code is a line graph titled 'The Elbow Method'. The x-axis is labeled 'Number of clusters' and ranges from 1 to 10. The y-axis is labeled 'WCSS' and ranges from 0 to 250,000. The graph shows a sharp decrease in WCSS as the number of clusters increases from 1 to 5, after which the decrease levels off, forming an 'elbow' shape. This suggests that 5 is the optimal number of clusters.

3. Entrainement du modèle de clustering sur les deux variables

▼ Entrainement du modèle de clustering sur les deux variables

```
[4] kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
     y_kmeans = kmeans.fit_predict(X)
```

4. Visualisation des clusters sur un graphique

▼ Visualisation des clusters sur un graphique

```
[5] plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
     plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
     plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
     plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
     plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
     plt.scatter(kmeans.cluster_centers[:, 0], kmeans.cluster_centers[:, 1], s = 300, c = 'yellow', label = 'Centroids')
     plt.title('Clusters of customers')
     plt.xlabel('Annual Income (k$)')
     plt.ylabel('Spending Score (1-100)')
     plt.legend()
     plt.show()
```



Chaque groupe de clients est représenté par une couleur. D'après ce graphe, les publicités du centre commerciales doivent viser majoritairement deux groupes : ceux dont les revenus sont compris inférieurs à 40.000\$ et dont le score de dépense est compris entre 60 et 100 (en couleur bleue), ceux qui ont des revenus compris entre 65000\$ et 140.000\$ avec un score de dépense de 60 à 100 (en couleur magenta).

III. Cas d'étude 3 : scolarité, expérience et revenus « Mincer 1974 »

1. Importation du dataset

Fichiers + Code + Texte

Cas d'étude 3 : scolarité, expérience et revenus « Mincer 1974 » :

▼ Importation des bibliothèques

```
[36] import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

▼ Importation du dataset

```
[37] dataset = pd.read_csv('Mincer.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Mincer.csv X

1 to 10 of 753 entries Filter

	inlf	hours	kidslt6	kidsge6	age	educ	wage	rep
1	1610	1	0	32	12	3.35400009155273	2.650000	
1	1656	0	2	30	12	1.3889000415802	2.650000	
1	1980	1	3	35	12	4.54549980163574	4.039999	
1	456	0	3	34	12	1.09650003910065	3.25	
1	1568	1	2	31	14	4.59180021286011	3.599999	
1	2032	0	0	54	12	4.74209976196289	4.699999	
1	1440	0	2	37	16	8.33329963684082	5.949999	
1	1020	0	0	54	12	7.84310007095337	9.979999	
1	1458	0	2	48	12	2.12619996070862	0	
1	1600	0	2	39	12	4.6875	4.150000	

Show 10 per page 1 2 10 70 76

2. Suppression des valeurs manquantes de la variables « revenu »

Initialement, il y avait 753 lignes et 22 colonnes et après la suppression des NaN, il ne reste que 428 lignes

▼ Suppression des valeurs manquantes de la variable 'wage'

```
[38] print(dataset.shape)
dataset2 = dataset.dropna(subset = ['wage'], how = 'any')
print(dataset2.shape)
```

```
(753, 22)
(428, 22)
```

Pour la suite, nous allons travailler Avec dataset2 qui ne contient pas de valeurs manquantes

```
[39] X2 = dataset2.iloc[:, :-1].values
y2 = dataset2.iloc[:, -1].values
```

3. Création des deux nouvelles variables `age2` (carre de la variable `age`) et `exper2` (carre de la variable `exper`)

Création des variables `age2` et `exper2`

```
[40] dataset2 = dataset2.assign(age2 = dataset2['age'] ** 2)
dataset2['exper2'] = dataset2['exper'] ** 2
dataset2.head()
```

	inlf	hours	kidslt6	kidsge6	age	educ	wage	repwage	hushrs	husage	...	motheduc	fatheduc	unem	city	exper	mwifeinc	lwage	expersq	age2	exper2
0	1	1610	1	0	32	12	3.3540	2.65	2708	34	...	12	7	5.0	0	14	10.910060	1.210154	196	1024	196
1	1	1656	0	2	30	12	1.3889	2.65	2310	30	...	7	7	11.0	1	5	19.499981	0.328512	25	900	25
2	1	1980	1	3	35	12	4.5455	4.04	3072	40	...	12	7	5.0	0	15	12.039910	1.514138	225	1225	225
3	1	456	0	3	34	12	1.0965	3.25	1920	53	...	7	7	5.0	0	6	6.799996	0.092123	36	1156	36
4	1	1568	1	2	31	14	4.5918	3.60	2000	32	...	12	14	9.5	1	7	20.100060	1.524272	49	961	49

5 rows × 24 columns

4. Répartition de la base de données en sous base de données (Entrainement/test)

Répartition de la base de données en sous base de données (Entrainement/test)

```
[41] from sklearn.model_selection import train_test_split
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size = 0.2, random_state = 0)
print(X2_train)
```

```
[[1.00e+00 6.10e+02 0.00e+00 ... 1.40e+01 1.35e+01 1.41e+00]
 [1.00e+00 1.98e+03 0.00e+00 ... 1.40e+01 1.31e+01 1.22e+00]
 [1.00e+00 3.36e+02 0.00e+00 ... 4.00e+00 9.00e+00 1.09e+00]
 ...
 [1.00e+00 2.00e+03 0.00e+00 ... 9.00e+00 1.78e+01 1.79e+00]
 [1.00e+00 1.98e+03 0.00e+00 ... 2.20e+01 7.14e+00 1.52e+00]
 [1.00e+00 2.32e+03 0.00e+00 ... 1.00e+01 7.80e+00 1.03e+00]]
```

5. Construction du modèle de régression linéaire multiple

Construction du modèle de la regression lineaire multiple

```
[42] from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X2_train, y2_train)
coefs = reg.coef_
intecept = reg.intercept_
print(intecept , coefs)
```

```
-143.80828252738155 [ 0.00e+00 -2.05e-02 1.66e+01 7.29e+00 3.61e+00 -9.41e-02 5.01e+00
 4.90e-01 5.63e-03 -7.57e-01 5.33e-01 -1.11e+00 3.97e-03 -1.40e+02
 1.11e+00 -1.96e+00 -1.81e+00 -5.35e+00 3.19e+01 -4.79e+00 -4.76e+01]
```

6. Prédiction des résultats Test

```
[51] y2_pred = reg.predict(X2_test)
      np.set_printoptions(precision=2)
      print(np.concatenate((y2_pred.reshape(len(y2_pred),1), y2_test.reshape(len(y2_test),1)),1))
```

[[3.81e+01	4.90e+01]
[[1.41e+02	1.21e+02]
[[9.38e+02	1.16e+03]
[[8.01e+01	1.00e+02]
[[2.87e+02	2.25e+02]
[[3.72e+02	2.89e+02]
[[2.33e+02	1.69e+02]
[[9.02e+01	8.10e+01]
[[2.28e+02	1.69e+02]
[[4.52e+02	3.61e+02]
[[1.30e+02	1.21e+02]
[[3.82e+02	3.24e+02]
[[-7.55e+01	9.00e+00]
[[1.13e+02	8.10e+01]
[[1.02e+02	8.10e+01]
[[4.16e+02	3.61e+02]
[[1.13e+02	1.00e+02]
[[-5.15e+01	1.00e+00]
[[2.00e+02	1.44e+02]
[[-1.52e+02	4.00e+00]
[[5.87e+02	5.76e+02]
[[1.84e+02	1.21e+02]
[[5.38e+02	5.29e+02]
[[-4.91e+01	2.50e+01]
[[7.26e+02	7.84e+02]
[[-1.18e+02	4.00e+00]
[[1.67e+02	1.44e+02]
[[5.95e+02	5.76e+02]

```

[ 6.54e+01  6.40e+01]
[-1.75e+02  0.00e+00]
[-5.02e+01  1.60e+01]
[ 2.88e+02  2.25e+02]
[-3.32e+01  1.60e+01]
[ 2.57e+02  1.96e+02]
[ 2.72e+02  1.96e+02]
[ 1.91e+01  3.60e+01]
[ 5.85e+02  5.76e+02]
[-6.31e+01  1.60e+01]
[ 2.47e+02  1.00e+02]
[ 5.83e+02  5.29e+02]
[ 4.18e+02  3.24e+02]
[ 2.94e+01  6.40e+01]
[ 3.46e+02  2.56e+02]
[ 3.78e+02  2.89e+02]
[ 1.05e+02  8.10e+01]
[ 3.71e+02  2.89e+02]
[ 4.69e+02  3.61e+02]
[-1.20e+02  4.00e+00]
[ 4.18e+02  3.61e+02]
[ 9.11e+01  8.10e+01]
[ 2.50e+02  1.69e+02]
[ 5.30e+01  6.40e+01]
[ 1.93e+02  1.21e+02]
[ 3.44e+02  2.56e+02]
[ 2.71e+02  1.96e+02]
[ 2.17e+02  1.69e+02]
[ 1.21e+02  8.10e+01]
[-3.46e+01  4.00e+00]
_

```

7. Calcul du coefficient de détermination R carré

▼ Calcul du R carré du modèle

```

[62] from sklearn.metrics import r2_score
      R_carre = r2_score(y2_test, y2_pred)
      R_carre
0.900091804446975

```

Le R carré est proche de 1, donc ce modèle est adéquat.

Pour l'améliorer, étant donné que la somme totale des carrés est invariable, on doit chercher à réduire la somme des carrés des résidus.