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.

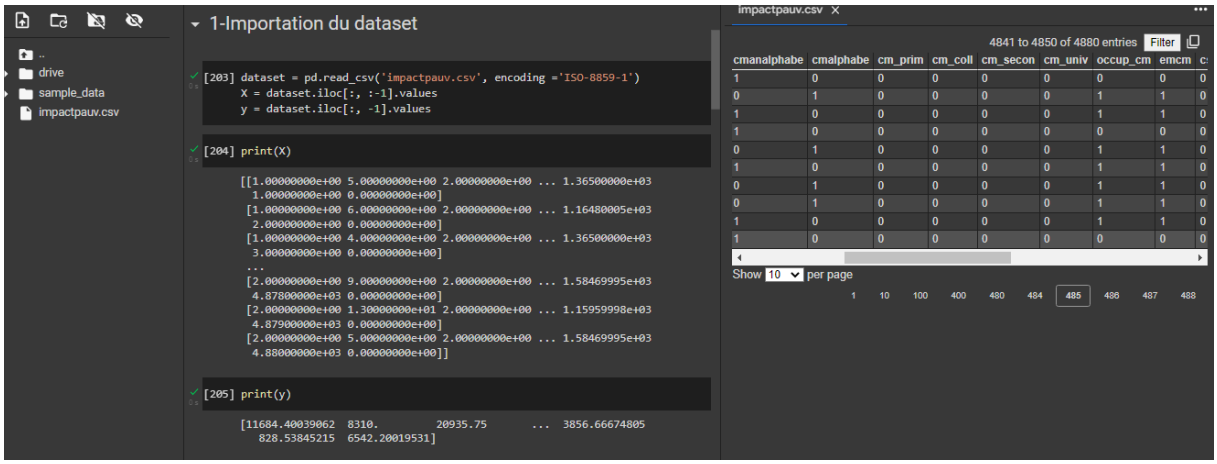
Table des matières

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

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

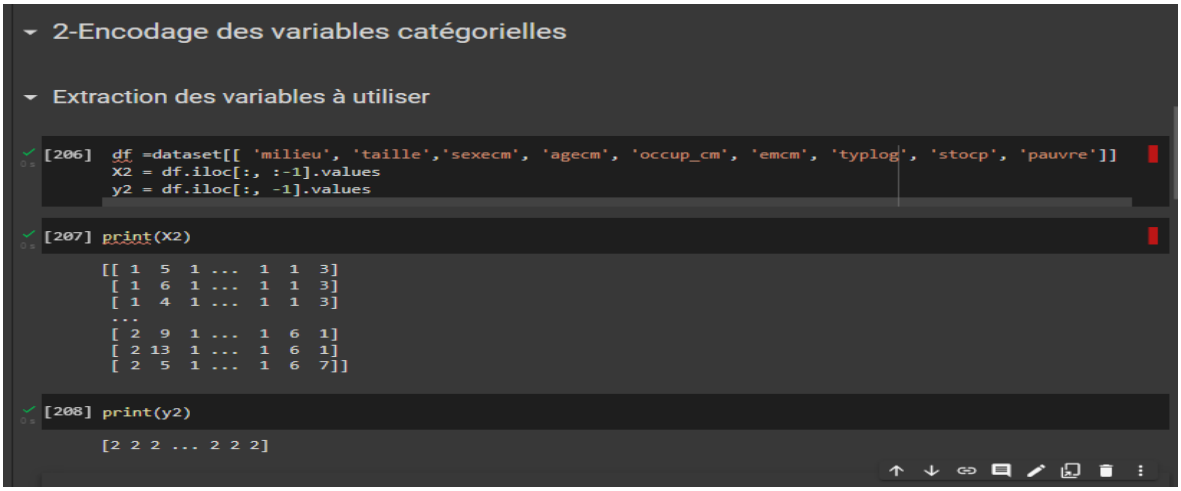
1. Importation du dataset

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



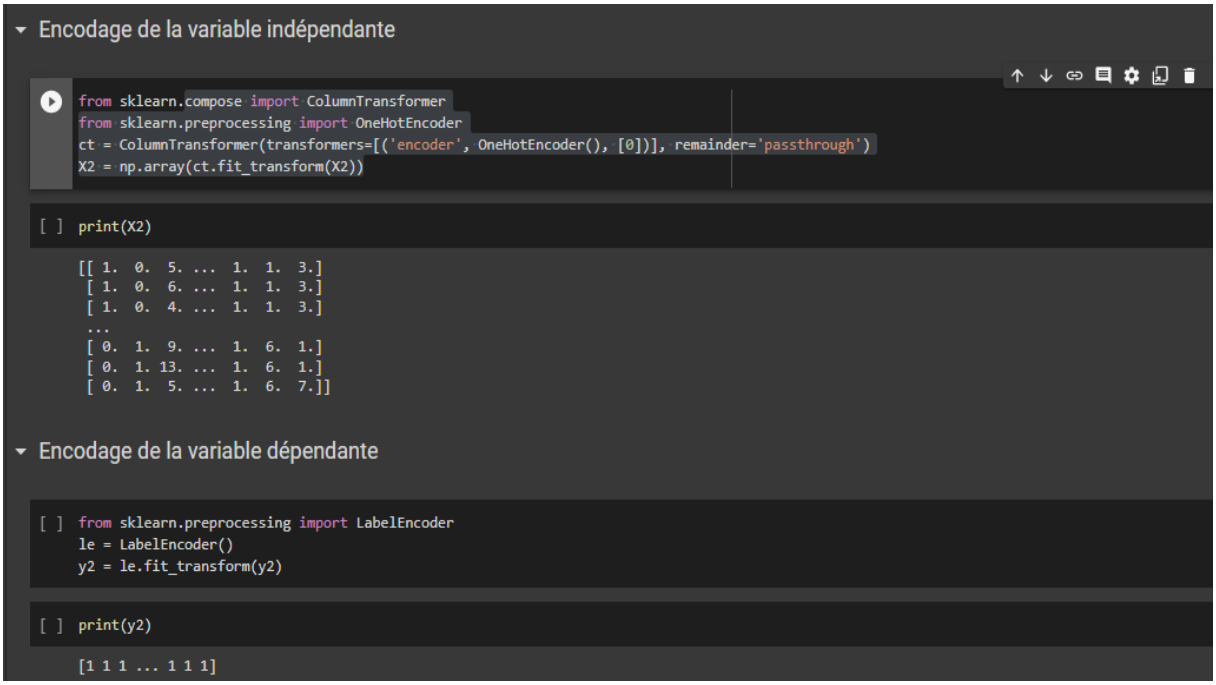
2. Extraction des variables à utiliser et encodage des variables catégorielles

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



y2 représente la colonne contenant la variable pauvre.

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

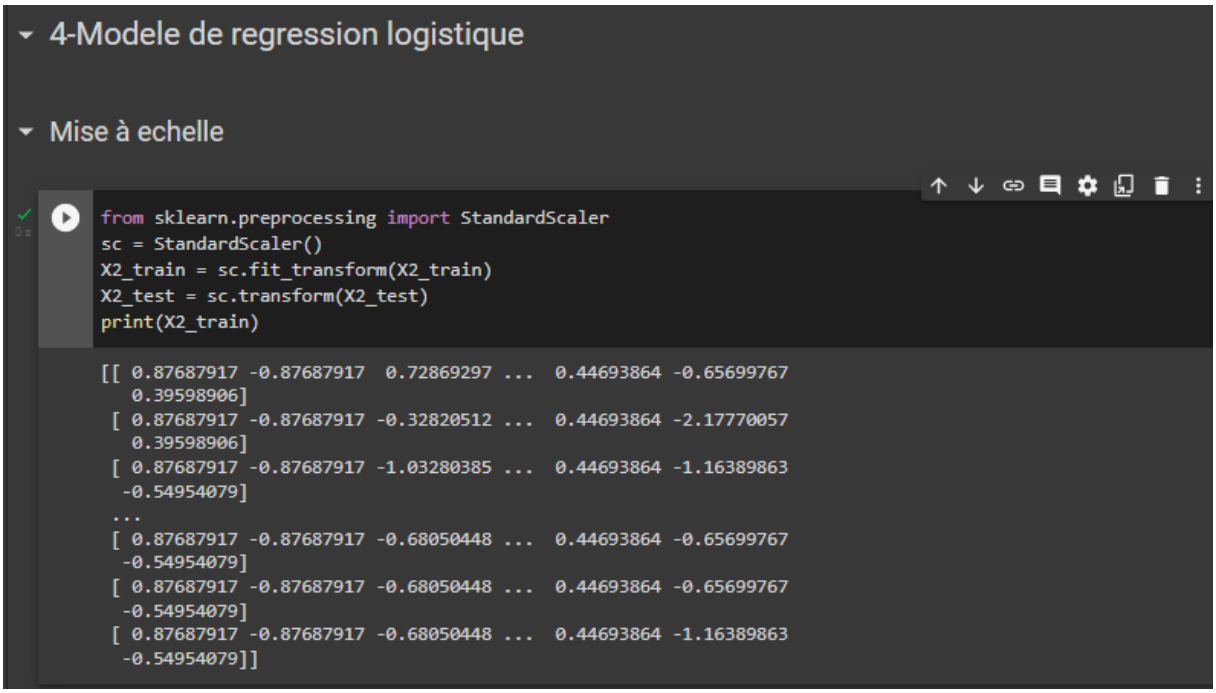


3. 3-Répartition de la base de données en sous base de données (Entrainement/test)

On a entrainé notre modelé puis afficher le X2_train

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.



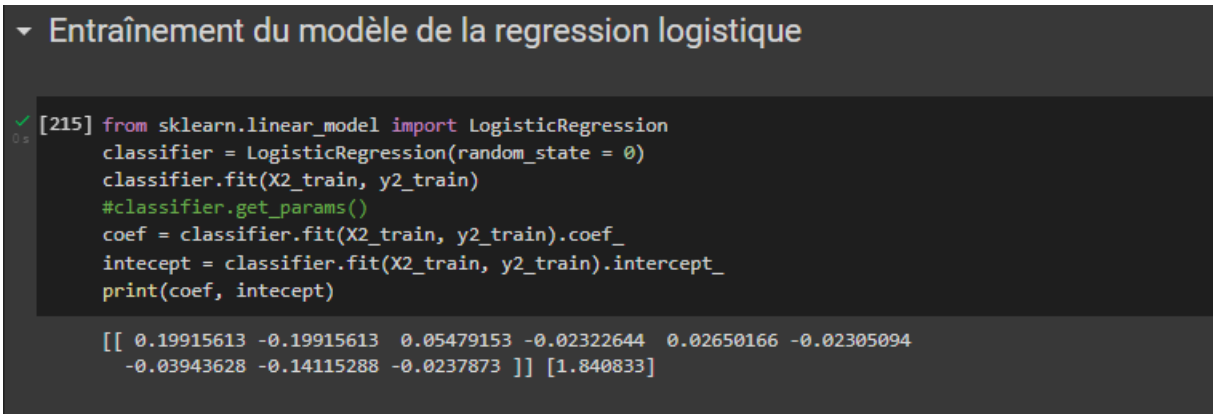
4-Modele de regression logistique

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

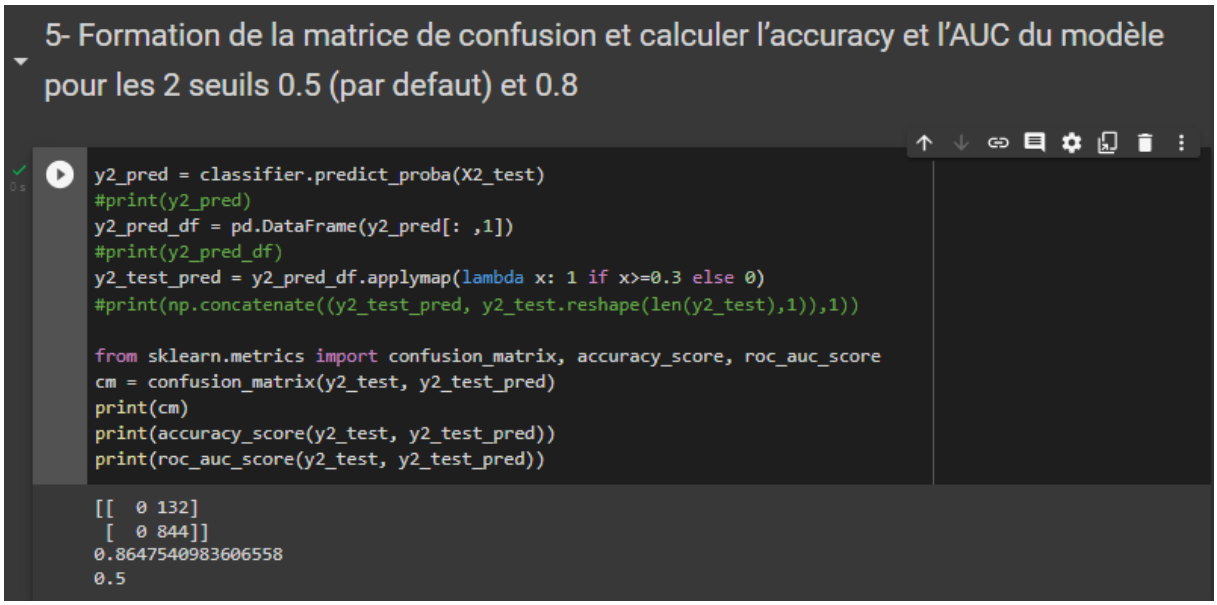


Entraînement du modèle de la regression logistique

```
[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

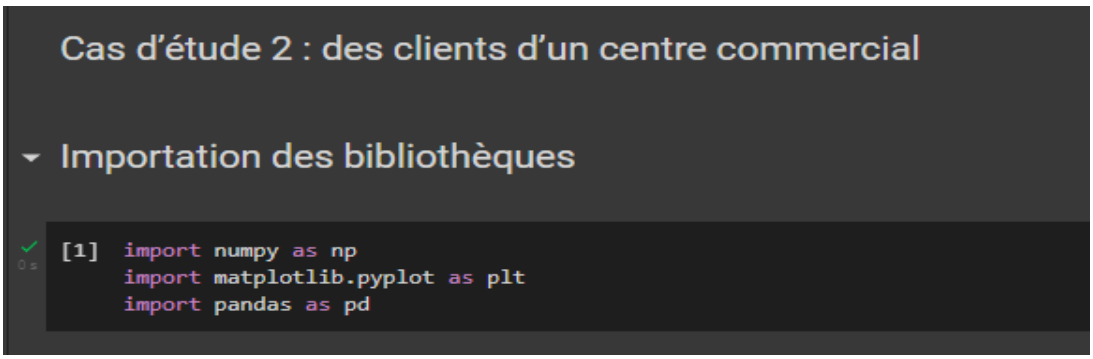


D'après le résultat donne par 'cm', on a 844 bons résultats et 132 mauvais résultats.

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

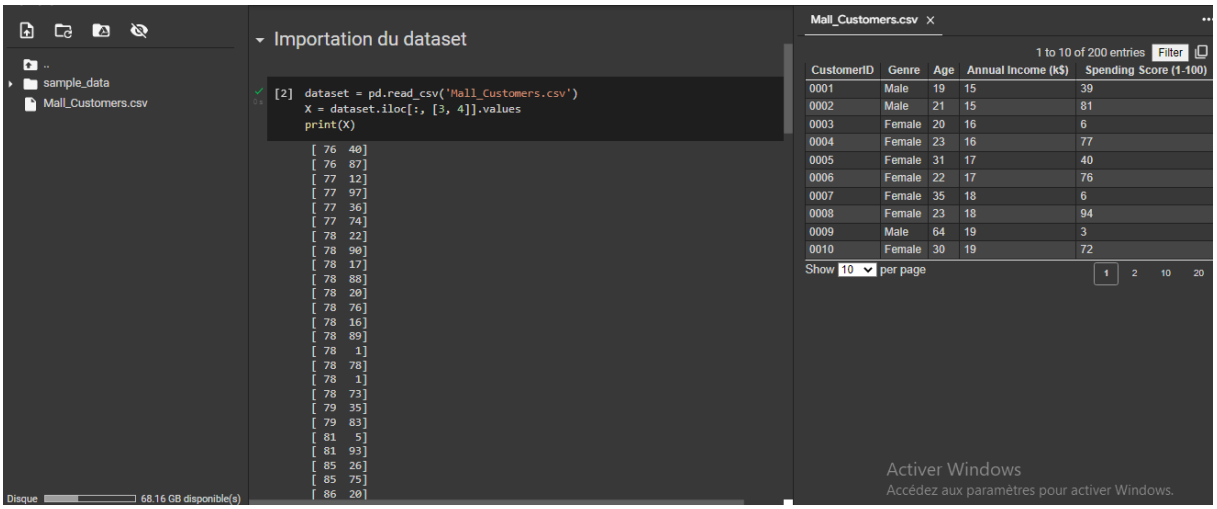
II. Cas d'étude 2 : des clients d'un centre commercial

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.



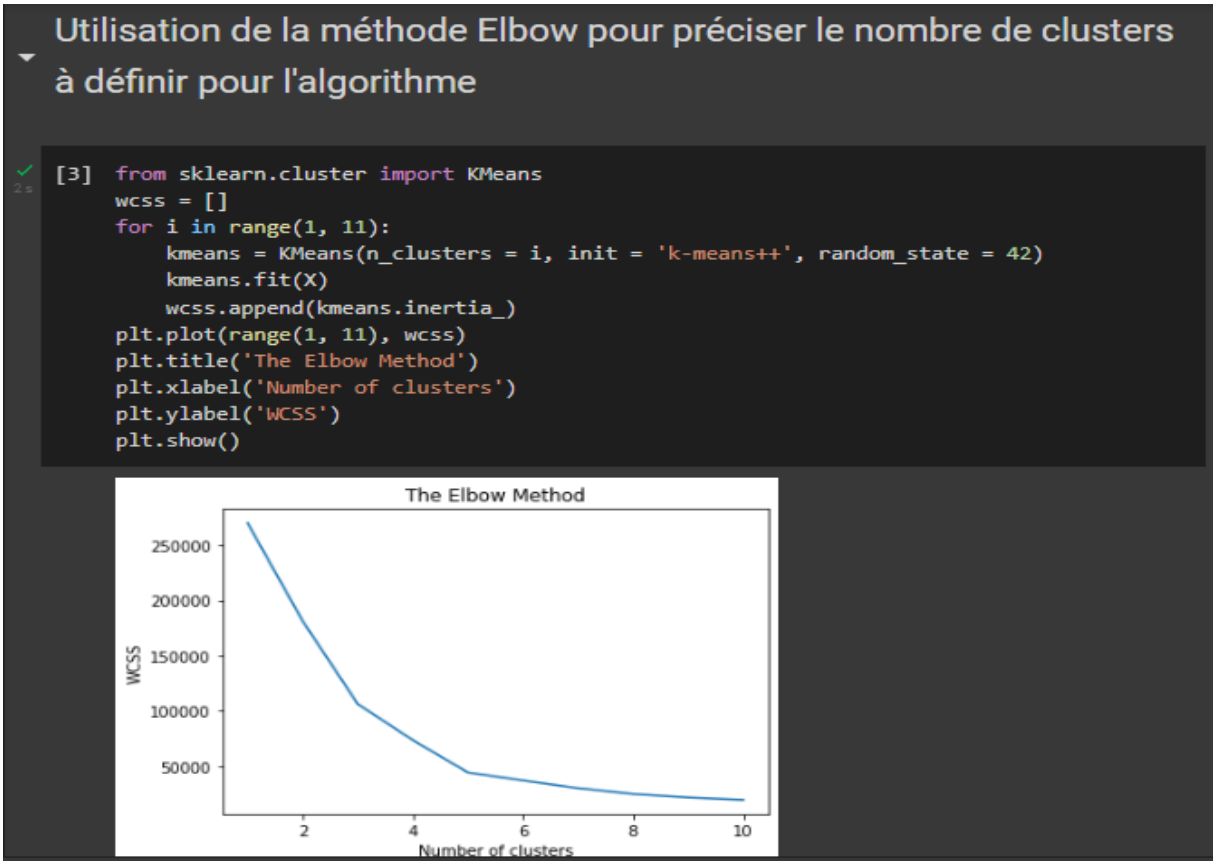
1. Importation du dataset

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.



2. Détermination du k-means

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



3. Entrainement du modèle de clustering sur les deux variables

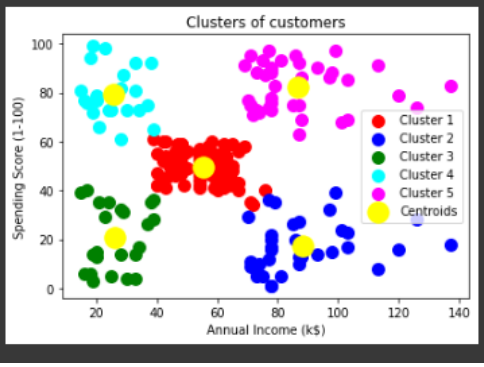
Entrainement du modèle de clustering sur les deux variables

```
[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()
```



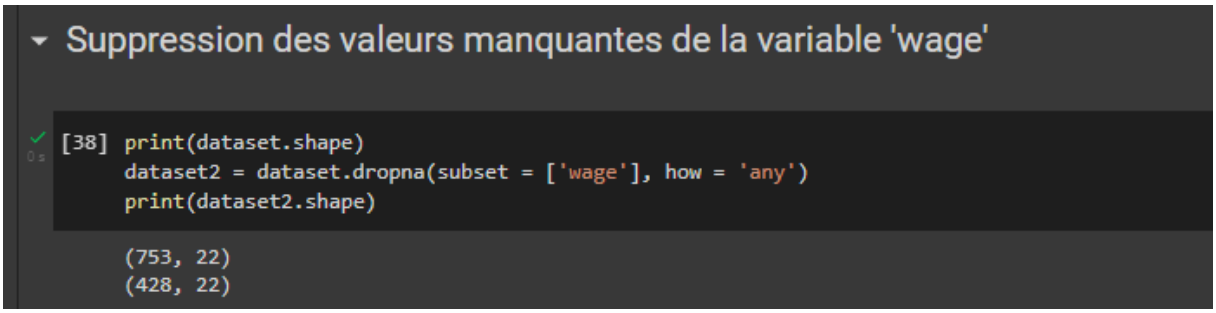
Chaque groupe de clients est représenté par une couleur. D'après ce graphe, les publicités du centre commerciales doivent viser majoritairement deux groupes : ceux dont les revenus sont compris inferieurs à 40.000$ et dont le score de dépense est compris entre 60 et 100 (en couleur bleue), ceux qui ont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

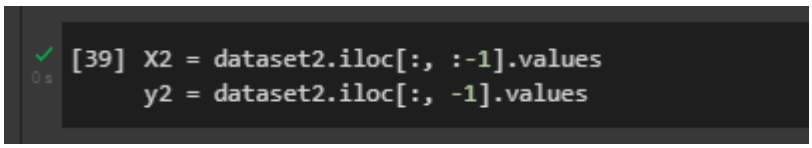
1. Importation du dataset

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 lignes



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



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	nwifeinc	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

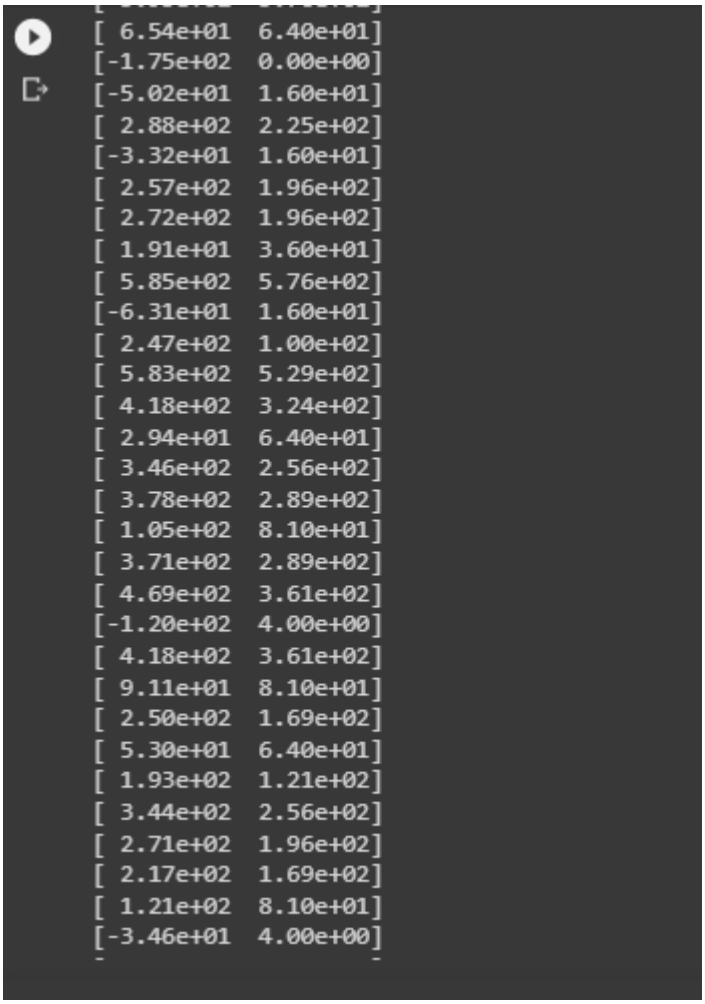
```
[42] from sklearn.linear_model import LinearRegression
     reg = LinearRegression().fit(X2_train, y2_train)
     coefs = reg.fit(X2_train, y2_train).coef_
     intecept = reg .fit(X2_train, y2_train).intercept_
     print(intecept , coefs)
```

```
-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]
```



7. Calcul du coefficient de détermination R carré

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 a réduire la somme des carrés des résidus.