AFD sous Python avec scientisttools

Duvérier DJIFACK ZEBAZE

Ce tutoriel a pour objectif de présenter rapidement les principales fonctionnalités offertes par le package « scientisttools » pour réaliser une Analyse Factorielle Discriminante ou Analyse Discriminante Descriptive.

Présentation des données

Nous allons illustrer ce tutoriel à travers l'exemple des Vins de Bordeaux (Michel Tenenhaus, 2007). On cherche à relier la qualité des vins de Bordeaux à des caractéristiques météorologiques. La variable à expliquer y est la qualité du vin et prend 3 modalités : 1 = bon, 2 = moyen et 3 = médiocre. Les variables explicatives de la qualité du vin sont les suivantes : X_1 (Somme des températures moyennes journalières (°C)), X_2 (Durée d'insolation (h)), X_3 (Nombre de jours de grande chaleur) et X_4 (Hauteur des pluies (mm)).

```
# Chargement des données
import pandas as pd
data = pd.read_excel("./donnee/wine_quality.xls",index_col=1)
```

Pour la suite, nous allons supprimer la colonne « Obs » de notre jeu de données.

```
# Suppression Obs.
donnee = data.drop(['Obs.'],axis=1)
```

Objectifs

L'analyse factorielle discriminante est une méthode descriptive. Elle vise à produire un système de représentation de dimension réduite qui permet de discerner les classes lorsqu'on y projette les individus. Il s'agit d'une méthode d'analyse factorielle. On peut la voir comme une variante de l'analyse en composantes principales où les centres de classes sont les individus, pondérés par leurs effectifs, et avec une métrique particulière (SAPORTA, 2006). Les variables latentes (ou discriminantes) sont exprimées par des combinaisons linéaires des variables originelles. Elles sont deux à deux orthogonales. Elles cherchent à assurer un écartement maximal entre les centres de classes. In fine, l'objectif est de mettre en évidence les caractéristiques qui permettent de distinguer au mieux les groupes.

Problématique

L'analyse factorielle discriminante ou analyse discriminante descriptive permet de caractériser de manière multidimensionnelle l'appartenance des individus à des groupes prédéfinis, ceci à l'aide de plusieurs variables explicatives prises de façon simultanée. En effet, il s'agit de construire un nouveau système de représentation qui permet de mettre en évidence ces groupes. Les objectifs de l'analyse factorielle discriminante sont double :

Table 1 – Qualité des vins de Bordeaux

	Obs.	Temperature	Soleil	Chaleur	Pluie	Qualite
1924	1	3064	1201	10	361	Moyen
1925	2	3000	1053	11	338	Mediocre
1926	3	3155	1133	19	393	Moyen
1927	4	3085	970	4	467	Mediocre
1928	5	3245	1258	36	294	Bon
1929	6	3267	1386	35	225	Bon
1930	7	3080	966	13	417	Mediocre
1931	8	2974	1189	12	488	Mediocre
1932	9	3038	1103	14	677	Mediocre
1933	10	3318	1310	29	427	Moyen
1934	11	3317	1362	25	326	Bon
1935	12	3182	1171	28	326	Mediocre
1936	13	2998	1102	9	349	Mediocre
1937	14	3221	1424	21	382	Bon
1938	15	3019	1230	16	275	Moyen
1939	16	3022	1285	9	303	Moyen
1940	17	3094	1329	11	339	Moyen
1941	18	3009	1210	15	536	Mediocre
1942	19	3227	1331	21	414	Moyen
1943	20	3308	1366	24	282	Bon
1944	21	3212	1289	17	302	Moyen
1945	22	3361	1444	25	253	Bon
1946	23	3061	1175	12	261	Moyen
1947	24	3478	1317	42	259	Bon
1948	25	3126	1248	11	315	Moyen
1949	26	3458	1508	43	286	Bon
1950	27	3252	1361	26	346	Moyen
1951	28	3052	1186	14	443	Mediocre
1952	29	3270	1399	24	306	Bon
1953	30	3198	1259	20	367	Bon
1954	31	2904	1164	6	311	Mediocre
1955	32	3247	1277	19	375	Bon
1956	33	3083	1195	5	441	Mediocre
1957	34	3043	1208	14	371	Mediocre

1. **Descriptif** : Mettre en évidence les caractéristiques qui permettent de distinguer au mieux les groupes ;
2. **Prédictif** : Classer automatiquement un nouvel individu (l'affecter à un groupe) à partir de ses caractéristiques

Rapport de corrélation

Nous mesurons le pouvons discriminant de chaque variables X_j en utilisant l'analyse de la variance à un facteur. Pour cela, nous utilisons le rapport de corrélation définit par :

$$\eta^2(X_j, y) = \frac{\text{Somme des carrés inter - classes}}{\text{Somme des carrés totale}} \tag{1}$$

Cet indicateur, compris entre 0 et 1, est basé sur la dispersion des moyennes conditionnelles. Il s'agit d'un indicateur de séparabilité des groupes :

— $\eta^2(X_j, y) = 0$, la discrimination est impossible, les moyennes conditionnelles sont confondues. La somme des carrés inter - classes est nulle.
— $\eta^2(X_j, y) = 1$, la discriminantion est parfaite, les points associés aux groupes sont agglutinés autour de leur moyenne respectives : la somme des carrés intra - classes est nulle, ce qui est équivalent à la somme des carrés inter - classes est egale à la somme des carrés totale.

```
# Pouvoir discriminant
from scientisttools.utils import eta2

R2 = dict()
for name in donnee.columns[:-1]:
    R2[name] = eta2(donnee["Qualite"],donnee[name])
R2 = pd.DataFrame(R2).T
```

Table 2 – Rapport de corrélation

	Sum. Intra	Sum. Inter	correlation ratio	F-stats	pvalue
Temperature	237722.121	420067.41	0.639	27.389	0.000
Soleil	202192.371	326909.07	0.618	25.061	0.000
Chaleur	1664.371	1646.57	0.497	15.334	0.000
Pluie	178499.212	97191.17	0.352	8.440	0.001

Toutes les p-values sont inférieures au seuil de 5%, par conséquent, il existe une différence significative dans la qualité du vin.

AFD

Chargement de scientisttools

Sage précaution avec les packages pour Python, nous affichons le numéro de la version de « scientisttools » utilisée dans ce tutoriel.

```
# version
import scientisttools
print(scientisttools.__version__)
```

```
## 0.0.9
```

Nous fonctionnons avec la version « 0.0.9 ».

```
from scientisttools.discriminant_analysis import CANDISC
```

On crée une instance de la classe CANDISC, en lui passant ici des étiquettes pour les lignes et les variables.

Le constructeur de la classe CANDISC possède un paramètre `n_components` qui indique le nombre d'axes discriminants à garder. Par défaut, la valeur du paramètre `n_components` est fixée à `None`.

Réalisez l'AFD sur toutes observations en tapant la ligne de code suivante :

```
# Instanciation
my_cda = CANDISC(n_components=2,
                 target=["Qualite"],
                 row_labels=donnee.index,
                 features_labels=["Temperature","Soleil","Chaleur","Pluie"],
                 parallelize=False)
```

<

— `n_components` : le nombre d'axes discriminants à garder dans les résultats
— `target` : le label de la variable cible.
— `row_labels` : les noms des lignes
— `features_labels` : les noms des variables explicatives
— `parallelize` : paralleliser l'algorithme.

On estime le modèle en appliquant la méthode `.fit` de la classe CANDISC sur le jeu de données à traiter.

```
# Apprentissage
my_cda.fit(donnee)
```

```
## CANDISC(features_labels=['Temperature', 'Soleil', 'Chaleur', 'Pluie'],
##         n_components=2,
##         row_labels=Int64Index([1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933
##             1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945,
##             1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956,
##             1957],
##            dtype='int64', name='Annee'),
##         target=['Qualite'])
```

Les valeurs propres

L'exécution de la méthode `my_cda.fit(donnee)` provoque le calcul de plusieurs attributs parmi lesquels `my_cda.eig_`.

```
print(my_cda.eig_)
```

```
## [[  3.27886049   0.13857402]
##  [  3.14028647          nan]
##  [ 95.94508632   4.05491368]
##  [ 95.94508632 100.        ]]
```

L'attribut `my_cda.eig_` contient :

— en 1ère ligne : les valeurs propres en valeur absolue
— en 2ème ligne : les différences des valeurs propres
— en 3ème ligne : les valeurs propres en pourcentage de la variance totale (proportions)
— en 4ème ligne : les valeurs propres en pourcentage cumulé de la variance totale.

La fonction `get_eig` retourne les valeurs propres sous forme de tableau de données.

```
# Valeurs propres
from scientisttools.extractfactor import get_eig
print(get_eig(my_cda))
```

```
##      eigenvalue  difference  proportion  cumulative
## LD1    3.278860    3.140286   95.945086   95.945086
## LD2    0.138574         NaN    4.054914  100.000000
```

Le premier axe discriminant contient 96% de l'information totale disponible.

On peut obtenir un résumé des principaux résultats en utilisant la fonction `summaryCANDISC`.

```
from scientisttools.extractfactor import summaryCANDISC
summaryCANDISC(my_cda)
```

```
##                      Canonical Discriminant Analysis - Results
##
##
## Summary Information
##        Total Sample Size  Variables  ...  DF Within Classes  DF Between Classes
## value                 34          4  ...                 31                   2
##
## [1 rows x 6 columns]
##
## Class Level information
##           n(k)      p(k)
## Qualite
## Mediocre    12  0.352941
## Bon         11  0.323529
## Moyen       11  0.323529
##
## Importance of components
##                              LD1      LD2
## Variance                   3.279    0.139
## Difference                 3.140      NaN
## % of var.                 95.945    4.055
## Cumulative of % of var.   95.945  100.000
##
## Group means:
##           Temperature    Soleil  Chaleur    Pluie
## Qualite
## Bon          3306.364  1363.636   28.545  305.000
## Mediocre     3037.333  1126.417   12.083  430.333
## Moyen        3140.909  1262.909   16.455  339.636
##
## Coefficients of canonical discriminants:
##                 LD1    LD2
## Temperature  -0.009  0.000
## Soleil       -0.007  0.005
## Chaleur       0.027 -0.128
## Pluie         0.006 -0.006
## Intercept    32.876 -2.165
##
## Classification functions coefficients:
##                  Bon  Mediocre  Moyen
## Temperature    0.018    -0.018  0.001
## Soleil         0.013    -0.015  0.004
## Chaleur       -0.023     0.084 -0.069
## Pluie         -0.011     0.014 -0.004
## Intercept    -72.590    65.609 -7.192
```

```
##
## Individuals (the 10 first)
##
##          LD1    LD2
## Annee
## 1924   0.883  0.872
## 1925   2.325  0.094
## 1926   0.995 -0.833
## 1927   2.727 -0.247
## 1928  -0.744 -1.721
## 1929  -2.231 -0.484
## 1930   2.747 -1.109
## 1931   2.534 -0.236
## 1932   3.731 -2.114
## 1933  -1.130 -1.368
##
## Continues variables
##
##              total.1  between.1  within.1  total.2  between.2  within.2
## Temperature   -0.901     -0.987    -0.724   -0.375     -0.211    -0.584
## Soleil        -0.897     -0.999    -0.701    0.116      0.003     0.176
## Chaleur       -0.771     -0.957    -0.525   -0.590     -0.336    -0.780
## Pluie          0.663      0.977     0.398   -0.361     -0.167    -0.421
```

Le champ `.coef_` nous intéresse particulièrement. Il correspond aux coeffcients des fonctions discriminantes :

```
# Affichage brut des ceofficients
print(my_cda.coef_)
```

```
## [[-8.56604551e-03  4.62505890e-05]
##  [-6.77386899e-03  5.32929330e-03]
##  [ 2.70544919e-02 -1.27636164e-01]
##  [ 5.86566500e-03 -6.17455623e-03]]
```

La matrice est de dimension (4,2) puisque nous avons un problème à ($K = 3$) classes (d'où $K-1$ axes discriminants) et 4 descripteurs.

```
#dimensions
print(my_cda.coef_.shape)
```

```
## (4, 2)
```

Il ne faut pas oublier les constantes (*intercept*) des fonctions discriminantes.

```
# et les constantes pour chaque classe
print(my_cda.intercept_)
```

```
## [32.87628192 -2.16527944]
```

Nous pouvons dès lors adopter une présentation plus sympathique des fonctions discriminantes. Pour ce faire, nous utilisons la fonction `get_candisc_coef` en fixant le paramètre « `choice = "absolute"` ».

```
# Affichage des coefficients
from scientisttools.extractfactor import get_candisc_coef
coef = get_candisc_coef(my_cda,choice="absolute")
coef
```

```
##                     LD1       LD2
## Temperature  -0.008566  0.000046
## Soleil       -0.006774  0.005329
## Chaleur       0.027054 -0.127636
## Pluie         0.005866 -0.006175
## Intercept    32.876282 -2.165279
```

Représentations factorielles

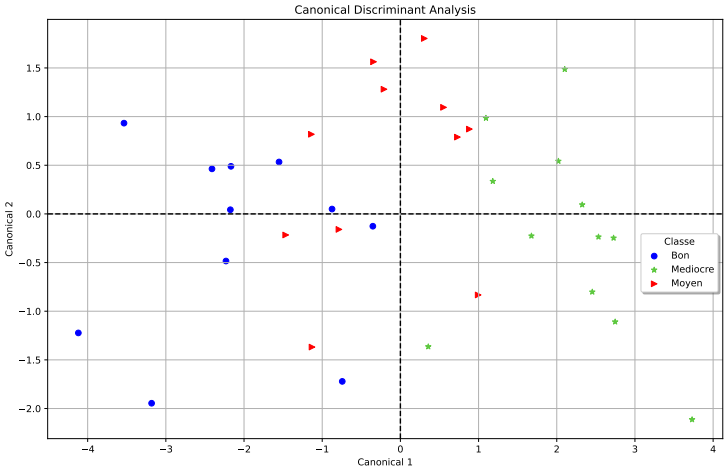
Coordonnées des individus

```
# Coordonnées des individus
from scientisttools.extractfactor import get_candisc_row
row_coord = get_candisc_row(my_cda)["coord"]
print(row_coord.head(6))
```

```
##             LD1       LD2
## Annee
## 1924   0.882552  0.871537
## 1925   2.325456  0.094220
## 1926   0.994856 -0.832957
## 1927   2.726862 -0.247244
## 1928  -0.743596 -1.721167
## 1929  -2.230889 -0.484319
```

```
# Carte des individus
from scientisttools.pyplot import plotCANDISC
import matplotlib.pyplot as plt

fig, axe = plt.subplots(figsize=(16,8))
plotCANDISC(my_cda,color=["blue",'#5DC83F','red'],marker=['o',"*",'>'],ax=axe)
plt.show()
```



Coordonnées des centres de classes

L'introduction des barycentre spermet de mieux situer la qualité relative des facteurs dans la discrimination des classes.

```
# Coordonnées des centres de classes
zk = my_cda.gcenter_
print(zk)
```

```
##                LD1       LD2
## Qualite
## Bon      -2.121963 -0.271812
## Mediocre  2.079247 -0.221184
## Moyen    -0.146307  0.513104
```

Evaluation globale du modèle

Evaluation statistique des facteurs

Distance entre centres de classes

Dans le plan factoriel, les distances sont camptabilisées à l'aide d'une simple distance euclidienne.

```
# Distances entre centres de classes
print(my_cda.gdisto_)
```

```
##                 Bon   Mediocre     Moyen
## Bon        0.000000  17.652729  4.519311
## Mediocre  17.652729   0.000000  5.492267
## Moyen      4.519311   5.492267  0.000000
```

Pouvoir discriminant des facteurs

Le pouvoir discriminant des facteurs est traduit par les valeurs propres qui leurs sont associées.

```
print(get_eig(my_cda))
```

```
##      eigenvalue  difference  proportion  cumulative
## LD1    3.278860    3.140286   95.945086   95.945086
## LD2    0.138574         NaN    4.054914  100.000000
```

Test MANOVA

Scientisttools forunit un test de sgnificativité globale du modèle.

```
# Significativité globale du modèle
print(my_cda.manova_)
```

```
##                   Multivariate linear model
## ============================================================
##
## ------------------------------------------------------------
##         Qualite          Value  Num DF  Den DF F Value Pr > F
## ------------------------------------------------------------
##           Wilks' lambda 0.2053 8.0000 56.0000  8.4505 0.0000
##          Pillai's trace 0.8880 8.0000 58.0000  5.7896 0.0000
##  Hotelling-Lawley trace 3.4174 8.0000 37.7500 11.7280 0.0000
##     Roy's greatest root 3.2789 4.0000 29.0000 23.7717 0.0000
## ============================================================
```

Nous nous intéressons en particulier à la ligne relative à « Wilks' Lambda ».

Performance globale

Nosu affichons les valeurs des statistiques suivantes : Lambda de Wilks, Transformation de Bartlett et de RAO.

```
# Performance globale
print(my_cda.global_performance_)
```

```
##                 Stat      Value       p-value
## 0     Wilks' Lambda   0.205263           NaN
## 1  Bartlett -- C(8)  46.712169  1.739815e-07
## 2    Rao -- F(8,56)   8.450507  1.890358e-07
```

L'écartement entre les barycentres conditionnels est significatif à 5%. L'analyse discriminante est viable dans ce contexte.

Test sur un ensemble de facteurs

Combien de facteurs faut - il retenir ?.

```
# Test sur un ensemble de facteur
print(my_cda.likelihood_test_)
```

```
##    statistic  DDL num.  DDL den.          Pr>F
## 0   8.450507       8.0      56.0  1.890358e-07
## 1   1.339549       3.0      29.0  2.807850e-01
```

Matrices de covariance

Elles sont directement fournies par l'objet « scientisttools »

Matrice de covariance intra - classe

```
# Covariance intra - classe
print(my_cda.wcov_)
```

```
##              Temperature       Soleil     Chaleur        Pluie
## Temperature  6991.827094  1714.255793  420.615865   392.273619
## Soleil       1714.255793  5946.834447  154.271168  -144.059715
## Chaleur       420.615865   154.271168   48.952094   -31.750446
## Pluie         392.273619  -144.059715  -31.750446  5249.976827
```

Matrice de covariance totale

```
# Covariance totale
print(my_cda.tcov_)
```

```
##               Temperature        Soleil      Chaleur        Pluie
## Temperature  19346.750865  12360.302768  1187.420415 -5130.448097
## Soleil       12360.302768  15561.807093   795.792388 -5317.760381
## Chaleur       1187.420415    795.792388    97.380623  -356.451557
## Pluie        -5130.448097  -5317.760381  -356.451557  8108.540657
```

Matrice de covariance inter - classe

```
# Matrice de covaiance inter - classe
print(my_cda.bcov_)
```

```
##               Temperature        Soleil     Chaleur        Pluie
## Temperature  12354.923771  10646.046975  766.804551 -5522.721715
## Soleil       10646.046975   9614.972646  641.521220 -5173.700666
## Chaleur        766.804551    641.521220   48.428528  -324.701111
## Pluie        -5522.721715  -5173.700666 -324.701111  2858.563830
```

Interprétation des facteurs

Elle permet la compréhension de la nature des facteurs.

Corrélation totale

```
# Correlation totale
print(my_cda.tcorr_)
```

```
##                    LD1       LD2
## Temperature -0.900589 -0.374779
## Soleil      -0.896744  0.116190
## Chaleur     -0.770513 -0.590030
## Pluie        0.662815 -0.361294
```

Correlation intra - classe

```
# Correlation intra - classe
print(my_cda.wcorr_)
```

```
##                    LD1       LD2
## Temperature -0.724221 -0.584256
## Soleil      -0.701280  0.176148
## Chaleur     -0.525372 -0.779910
## Pluie        0.398218 -0.420797
```

Correlation inter - classe

```
# Corrélation inter - classe
print(my_cda.bcorr_)
```

```
##                    LD1       LD2
## Temperature -0.986651 -0.211244
## Soleil      -0.998654  0.002625
## Chaleur     -0.957391 -0.335599
## Pluie        0.976576 -0.166812
```

Prediction des classes

Considérons l'année 19858. Les données (hypothétiques) de cette année sont :

```
## Inidvidu supplémentaire
XTest = pd.DataFrame({"Temperature" : 3000,
                      "Soleil" : 1100,
                      "Chaleur" : 20,
                      "Pluie" : 300},index=[1958])
XTest
```

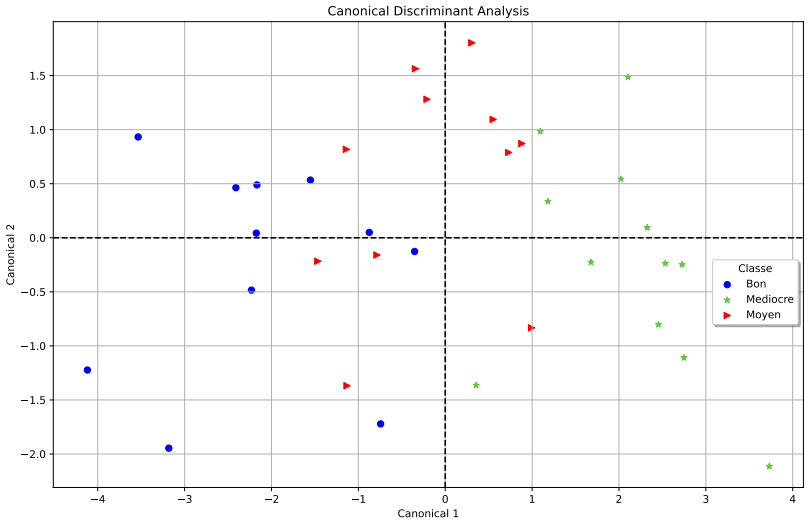
```
##       Temperature  Soleil  Chaleur  Pluie
## 1958         3000    1100       20    300
```

Coordonnées factorielles

```
# Coordonées factorielles
row_sup_coord = my_cda.transform(XTest)
print(row_sup_coord)
```

```
##            LD1       LD2
## 1958  2.027679 -0.569395
```

```
fig, axe = plt.subplots(figsize=(16,8))
plotCANDISC(my_cda,color=["blue",'#5DC83F','red'],marker=['o',"*",'>'],ax=axe)
axe.plot(row_sup_coord["LD1"],row_sup_coord["LD2"])
plt.show()
```



La fonction `predict()` permet de produire les prédictions à partir de la matrice des explicatives en test.

```
# Prédiction simple
pred = my_cda.predict(XTest)
print(pred)
```

```
##        predict
## 1958  Mediocre
```

Fonctions de classement explicites

La classe CANDISC de scientisttools retourne les fonctions de décision issues de l'analyse factorielle discriminante. Pour celà, il faut spécifier l'argument « `choice == "score"`.

```
# Fonctions de décision - AFD
score_coef = get_candisc_coef(my_cda,choice = "score")
print(score_coef)
```

```
##                     Bon   Mediocre     Moyen
## Temperature    0.018164  -0.017821  0.001277
## Soleil         0.012925  -0.015263  0.003726
## Chaleur       -0.022716   0.084484 -0.069449
## Pluie         -0.010768   0.013562 -0.004026
## Intercept    -72.590473  65.609287 -7.191833
```

Prédiction des classes sur l'échantillon d'apprentissage

```
import numpy as np
# Prédiction sur XTrain
X = donnee[donnee.columns[:-1]]
y_pred = my_cda.predict(X)

# Distribution des classes prédictes
print(np.unique(y_pred,return_counts=True))
```

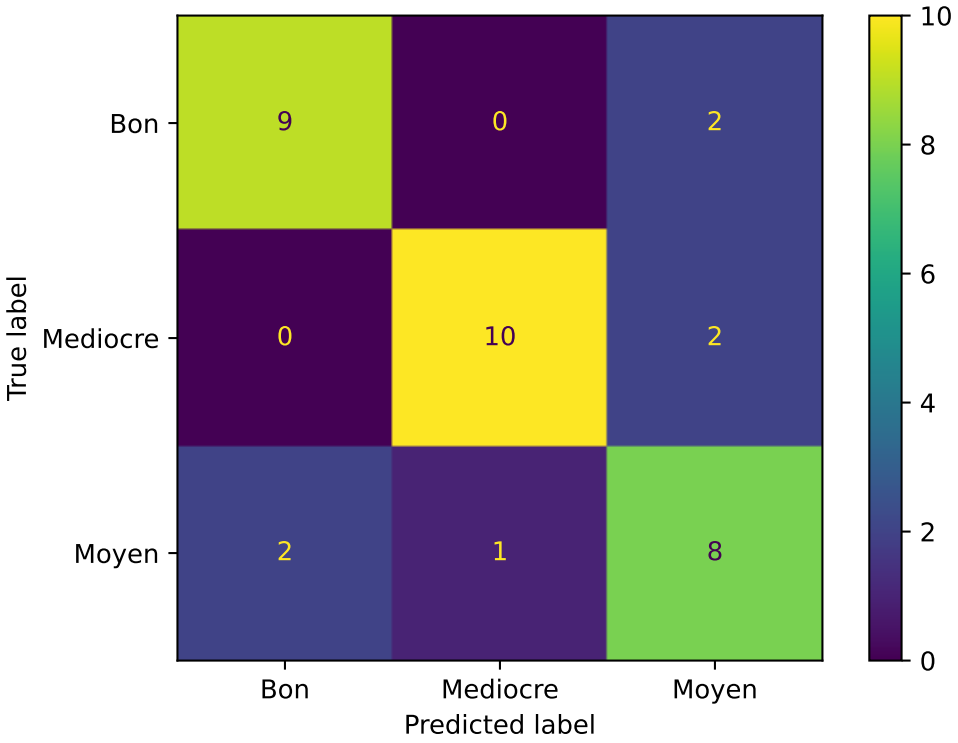
```
## (array(['Bon', 'Mediocre', 'Moyen'], dtype=object), array([11, 11, 12], dtype=int64))
```

11 observations ont été prédite « Bon », 11 « Medicocre » et 12 « Moyen ».

Matrice de confusion et taux de bon classement

La matrice de confusion est issue de la confrontation entre ces prédictions et les classes observées. Nous faisons appel au module « metrics » de la librairie « scikit-learn ».

```
# Matrice de confusion
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(donnee.Qualite,y_pred,labels=my_cda.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=my_cda.classes_)
disp.plot();
plt.show()
```



La fonction `score()` nous donne le taux de reconnaissance (ou taux de succès).

```
# Taux de succès
print(my_cda.score(X,donnee.Qualite))
```

```
## 0.7941176470588235
```

Notre taux de succès est de 79%.

La fonction `classification_report()` génère un rapport sur les performances globales, mais aussi sur les reconnaissances par classe (rappel, précision et F-Measure[F1-Score])

```
# rapport
from sklearn.metrics import classification_report
print(classification_report(donnee.Qualite,y_pred))
```

```
##               precision    recall  f1-score   support
##
##          Bon       0.82      0.82      0.82        11
##     Mediocre       0.91      0.83      0.87        12
##        Moyen       0.67      0.73      0.70        11
##
##     accuracy                           0.79        34
##    macro avg       0.80      0.79      0.79        34
## weighted avg       0.80      0.79      0.80        34
```

Nous retrouvons, entre autres le taux de succès de 79%.

Probabilité d'appartenance

« scientisttools » peut aussi calculer les probabilités d'affectation aux classes avec `predict_proba()`. Elle permettent une analyse plus fine de la qualité du modèle, via la construction de la courbe ROC par exemple, dont le principe reste valable pour les problèmes multi - classes.

```
# Probabilité d'appartenance
print(my_cda.predict_proba(X).head(6))
```

```
##              Bon  Mediocre     Moyen
## Annee
## 1924    0.006695  0.344613  0.648692
## 1925    0.000045  0.958846  0.041109
## 1926    0.009222  0.698039  0.292739
## 1927    0.000009  0.986519  0.013472
## 1928    0.641715  0.031256  0.327029
## 1929    0.933408  0.000094  0.066499
```

Sélection de variables

Limiter le modèle aux variables explicatives pertinentes est primordial pour l'interprétation et le deploiement des modèles.

Actuellement sous « scientisttools », seule la sélection backward est disponible.

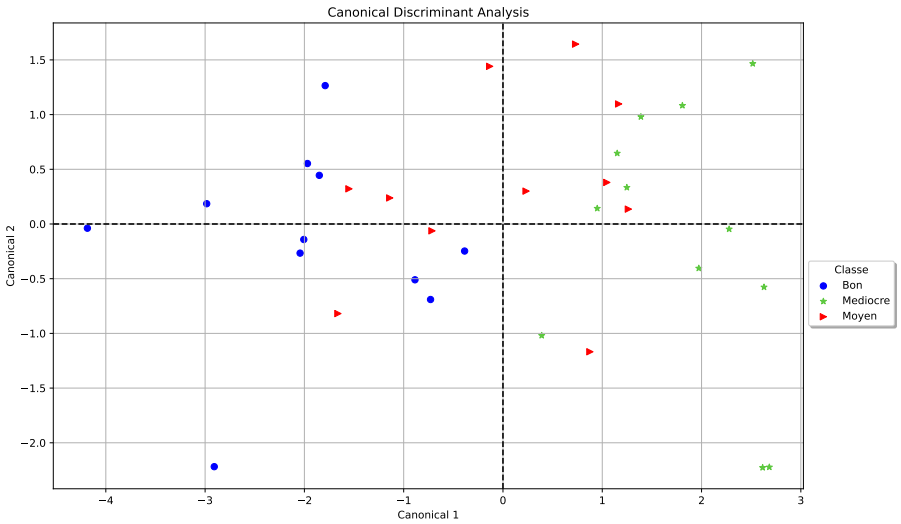
```
# Selection backward
from scientisttools.discriminant_analysis import STEPDISC
stepdisc = STEPDISC(method="backward",alpha=0.01,model_train=True,verbose=True)
stepdisc.fit(my_cda)
```

```
##              Wilks L.  Partial L.         F   p-value
## Temperature  0.213053    0.750263  4.660113  0.017906
## Soleil       0.220777    0.724016  5.336596  0.010876
## Chaleur      0.181627    0.880076  1.907721  0.167217
## Pluie        0.205825    0.776611  4.027032  0.029030
##
##              Wilks L.  Partial L.         F   p-value
## Temperature  0.281219    0.645857  7.950792  0.001766
## Soleil       0.247078    0.735102  5.225166  0.011535
## Pluie        0.229942    0.789886  3.857090  0.032710
##
##              Wilks L.  Partial L.         F   p-value
## Temperature  0.358983    0.640536  8.417873  0.001254
## Soleil       0.339493    0.677309  7.146457  0.002896
##
## STEPDISC(method='backward', model_train=True)
```

```
# Modèle optimal
stepdisc.train_model_
```

```
## CANDISC(features_labels=['Temperature', 'Soleil'],
##         row_labels=Int64Index([1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933
##             1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945,
##             1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956,
##             1957],
##            dtype='int64', name='Annee'),
##         target=['Qualite'])
```

```
# Représentation graphique
fig, axe =plt.subplots(figsize=(16,8))
plotCANDISC(stepdisc.train_model_,
            color=["blue",'#5DC83F','red'],
            marker=['o',"*",'>'],ax=axe)
plt.show()
```




```
# Summary
summaryCANDISC(stepdisc.train_model_,to_markdown=False)
```

```
##                     Canonical Discriminant Analysis - Results
##
##
## Summary Information
##        Total Sample Size  Variables  ...  DF Within Classes  DF Between Classes
## value                 34          2  ...                 31                   2
##
## [1 rows x 6 columns]
##
## Class Level information
##           n(k)      p(k)
## Qualite
## Mediocre    12  0.352941
## Bon         11  0.323529
```

```
## Moyen       11  0.323529
##
## Importance of components
##                             LD1      LD2
## Variance                  2.643    0.053
## Difference                2.590      NaN
## % of var.                98.020    1.980
## Cumulative of % of var.  98.020  100.000
##
## Group means:
##           Temperature    Soleil
## Qualite
## Bon          3306.364  1363.636
## Mediocre     3037.333  1126.417
## Moyen        3140.909  1262.909
##
## Coefficients of canonical discriminants:
##                 LD1     LD2
## Temperature  -0.007  -0.009
## Soleil       -0.007   0.010
## Intercept    32.868  16.032
##
## Classification functions coefficients:
##                  Bon  Mediocre  Moyen
## Temperature   0.016    -0.012 -0.003
## Soleil        0.013    -0.015  0.003
## Intercept   -70.474    54.095  4.269
##
## Individuals (the 10 first)
##
##          LD1    LD2
## Annee
## 1924   1.046  0.380
## 1925   2.629 -0.577
## 1926   0.876 -1.168
## 1927   2.615 -2.227
## 1928  -0.729 -0.690
## 1929  -1.850  0.445
## 1930   2.683 -2.223
## 1931   1.807  1.083
## 1932   1.972 -0.405
## 1933  -1.662 -0.819
##
## Continues variables
##
##              total.1  between.1  within.1  total.2  between.2  within.2
## Temperature   -0.933     -0.995    -0.813   -0.359     -0.149    -0.582
## Soleil        -0.917     -0.993    -0.777    0.399      0.065     0.630
```

Bien qu'il soit possible de déduire un mécanisme de classement en analyse factorielle discriminante, sa finalité est bien différente de l'analyse discriminante linéaire, prédictive. Mais les deux approches se rejoignent.

Pour plus d'informations sur l'AFD sous scientisttools, consulter le notebook https://github.com/enfantbenidedieu/scientisttools/blob/master/notebooks/candisc_wine.ipynb.