Rapport du projet

Thème : Analyse Discriminante Linéaire

Cours de Python

Licence 3 Informatique Décisionnelle et Statistique

Par : Aleksandra KRUCHININA, Aymeric DELEFOSSE et Mamadou DIALLO

Table des matières

[Introduction 3](#_Toc61201617)

[Module Calculations 3](#_Toc61201618)

[verification\_NA 3](#_Toc61201619)

[recodification\_var\_expl 3](#_Toc61201620)

[freq\_relat 3](#_Toc61201621)

[means\_class 3](#_Toc61201622)

[cov\_matrix 4](#_Toc61201623)

[pooled\_cov\_matrix 4](#_Toc61201624)

[wilks 4](#_Toc61201625)

[wilks\_log 4](#_Toc61201626)

[p\_value 4](#_Toc61201627)

[Module Reporting 5](#_Toc61201628)

[class HTML 5](#_Toc61201629)

[create\_html\_head 5](#_Toc61201630)

[stepdisc\_html\_output 5](#_Toc61201631)

[discrim\_html\_output 5](#_Toc61201632)

[discrim\_html\_output\_datapane 5](#_Toc61201633)

[class PDF(FPDF) 5](#_Toc61201634)

[footer 5](#_Toc61201635)

[discrim\_pdf\_output 5](#_Toc61201636)

[Module discriminant\_analysis 7](#_Toc61201637)

[class LinearDiscriminantAnalysis 7](#_Toc61201638)

[Attributs 7](#_Toc61201639)

[\_\_init\_\_ 7](#_Toc61201640)

[\_stats\_dataset 7](#_Toc61201641)

[\_stats\_pooled\_cov\_matrix 8](#_Toc61201642)

[\_stats\_classes 8](#_Toc61201643)

[\_stats\_wilks 8](#_Toc61201644)

[fit 8](#_Toc61201645)

[Predict 9](#_Toc61201646)

[confusion\_matrix 10](#_Toc61201647)

[accuracy\_score 10](#_Toc61201648)

[wilks\_decay 11](#_Toc61201649)

[stepdisc 11](#_Toc61201650)

[Pseudo-code 11](#_Toc61201651)

# Introduction

Pour faire notre application on a choisi de créer une classe LDA, LinearDiscriminantAnalysis, cette solution permet de créer des instances, mieux structurer et organiser le code. En plus cela a permis d’utiliser les variables calculées dans certaines fonctions (qui n’étaient pas utilisées dans leurs retours) à partir d’autres fonctions. Par exemple, l’attribut coef\_ qui est construit dans la fonction fit() et qui est utilisé ultérieurement dans la fonction predict().

On a divisé le code en trois modules : calculations, reporting et discriminant\_analysis.

# Module Calculations

C’est le module qui regroupe des fonctions de vérification et de modification de données ainsi que des fonctions permettant d'effectuer les calculs nécessaires pour la pratique de l'analyse discriminante linéaire (lambda de Wilks, matrice de covariance, etc.).

Fonctions :

## verification\_NA

On a repris l’approche de la procédure DISCRIM de SAS : les valeurs nulles ne sont pas prises en compte pour l’analyse, donc les observations précédant ces valeurs sont annulées.

Fonction verification\_NA(inputData : matrice, targetValues  : vecteur) :

Définir n et p

Concaténation des X (inputData) et y (targetValues)

Suppression des observations/lignes avec les valeurs nulles

Calcul du nombre de lignes supprimées

Redéfinition de X et y sans lignes supprimées

Affichage du nombre de lignes supprimées

Renvoyer nouveaux X et y

Fin Fonction

## recodification\_var\_expl :

La fonction *recodage\_var\_expl()* de notre module calculations  permet de convertir des variables qualitatives en variables quantitatives. L’Analyse de  Discriminante Linéaire (ADL) ne cadrant tout simplement pas avec les variables qualitatives, nous oblige de recoder ces variables avant l’application de cette méthode statistique sur nos données. Plusieurs approches existent afin de recoder des variables qualitatives parmi elles : la méthode DISQUAL et la méthode des dummy variables (celle que nous allons utiliser ici) pour ne citer que ces deux là.  La méthode des dummy variables (***page 76 du livre Analyse Discriminante Linéaire de Ricco Rakotomalala, lien en annex*e**)  octroie M-1 variables à chaque variables qualitatives possédant M modalités (notion modalité de référence), à ceux-ci on attribue soit 0 soit 1 représentant la présence ou non de catégories.

À savoir :

* Cette fonction n’est utile qu’en cas de présence de **variables qualitatives** dans les **variables explicatives** des données ;
* Si on est pas en présence que de variables quantitatives il faut directement implémenter la fonction *fit()* de la classe LDA ;

Fonction recodification\_var\_expl(inputData) :

Création d’un dictionnaire vide et lui

Attribuer des valeurs soit vraie ou fausse selon que la variable est positive ou pas

Créer une liste de valeurs booléen

Convertir le type de la liste en type object de python si sa valeur égale à faux

Recoder les variables de type object en tenant compte de la modalité de référence

Jointure des données initiales avec les données recodées

Suppression de variables non numériques de nos données finales

Retourner les données recodées

## freq\_relat

La fonction calcule les fréquences relatives.

Fonction freq\_relat(targetValues, n):

Calcul du nombre d'effectifs par classes (effClassValues)

Calcul des fréquences relatives par classes (freqClassValues)

Renvoyer ces deux vecteurs

Fin Fonction

## means\_class

La fonction calcule des moyennes conditionnelles selon le groupe d'appartenance.

Fonction means\_class(inputData, targetValues):

Définition des noms des classes et conversion des valeurs cibles en numérique

Calcul du nombre d'effectifs par classe

Initiation d'une matrice remplie de 0

Remplir cette matrice avec les effectifs par classe

Division par le nombre d'effectifs par classe

Renvoyer les moyennes conditionnelles

Fin Fonction

## cov\_matrix

## pooled\_cov\_matrix

## wilks

## wilks\_log

## p\_value

# Module Reporting

Ce module regroupe les différentes classes pour la création de reporting automatique, en format HTML et PDF, dont les sorties ressemblent à celles des PROC DISCRIM et STEPDISC de SAS.

## class HTML

Méthodes :

### create\_html\_head

### stepdisc\_html\_output

### discrim\_html\_output

### discrim\_html\_output\_datapane

## class PDF(FPDF)

Cette classe hérite la classe FPDF du package fpdf. Inspiré par le tutoriel : https://pyfpdf.readthedocs.io/en/latest/Tutorial/index.html

Méthodes :

### footer

Footer facilite l'affichage du bas de page lors d’une création d’un fichier pdf.

### discrim\_pdf\_output

Cette méthode créée un fichier pdf, dont l’utilisateur indique le nom. Il contient des sorties ressemblant aux sorties de la procédure DISCRIM de SAS : la description générale de l’échantillon (Taille d'échantillon totale, le nombre des variables et des classes, les DDL), l’information au niveau classe (les fréquences et proportions des classes), l’informations sur la matrice de covariance combinée, les statistiques de lambda de Wilks, les coefficients et l’intercept construits par le modèle.

La matrice de confusion et le taux de précision ne sont pas présentes dans cette méthode car on peut les invoquer par des méthodes séparées - confusion\_matrix() et accuracy\_score().

Fonction discrim\_pdf\_output(ProcDiscrim, fileName) :

Si les 4 derniers caractères ne contiennent pas ".pdf" :

Concaténer le fileName avec ".pdf"

Appel aux différents attributs de l’instance de la classe LinearDiscriminantAnalysis contenant l’information qu’on a besoin de récupérer pour restituer les sorties

Création d’une instance de la classe PDF

Définition des caractères des polices pour le titre

Ajout du titre du paragraphe

Ligne rouge

Définition des caractères des polices pour le texte

Boucle : récupérant l’index et l’élément de l’info général des attributs de LDA

Ajout des cellules de pdf contenant les informations selon l’attribut du modèle choisi

Ligne rouge

Fin Boucle

Idem pour les statistiques des classes, l’information sur la matrice de covariance, sur la fonction de classement, sur le lambda de Wilks

Sauvegarde le fichier

Fin Fonction

# Module discriminant\_analysis

## class LinearDiscriminantAnalysis

### Attributs

dataset : le jeu de données

classEtiquette : le nom de la variable cible

classNames : les noms des valeurs prises pour la variable cible

varNames : les noms des variables explicatives

n : la taille de l'échantillon

p : le nombre de variables explicatives

K : le nombre de classes

V : les matrices de covariance totale

Vb : les matrices de covariance biaisée

W : la matrice de covariance intra-classe

Wb : la matrice de covariance intra-classe biaisée

infoDataset : l’informations de bases sur le jeu de données

infoCovMatrix : les statistiques de la matrice de covariance intra-classe

infoClasses : les effectifs et fréquences relatives des classes

infoWilksStats : les statistiques du Lambda de Wilks

intercept\_ : l’intercept calculé par le modèle

coef\_ : les coefficients du modèle

infoFuncClassement : les valeurs de la fonction de classement

confusionMatrix : la matrice de confusion

confusionMatrixGraph : la matrice de confusion, version « graphique »

accuracy : le taux de précision

Methodes:

On a utilisé un underscore devant les méthodes à « weak internal usage ». En importation de la librairie (import \* from LinearDiscriminantAnalysis) ces méthodes ne seront pas convoquées.

### \_\_init\_\_

### \_stats\_dataset

Sert à créer l’attribut infoDataset qui représentent une matrice contenant l’information suivante : "Taille d'échantillon totale", "Variables", "Classes", "Total DDL", "DDL dans les classes", "DDL entre les classes".

### \_stats\_pooled\_cov\_matrix

Cette méthode calcule des statistiques de la matrice de covariance intra-classes

Fonction \_stats\_pooled\_cov\_matrix() :

Calcul du rang de la matrice de cov. intra-classes (W)

Calcul du logarithme naturel du déterminant de W

Restitution de l’attribut infoCovMatrix qui contient "Rang de la mat. de cov. intra-classes" et "Log. naturel du det. de la mat. de cov. intra-classes"

Fin Fonction

### \_stats\_classes

Cette méthode calcule des statistiques des classes

Fonction \_stats\_classes () :

Restitution de la variable cible

Calcul des effectifs et fréquences relatives des classes en utilisant la fonction du module calculations

Restitution de l’attribut infoClasses qui contient "Effectifs" et "Fréquences"

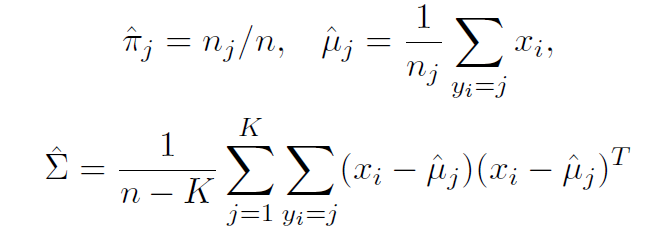
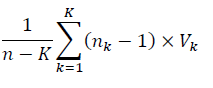
Fin Fonction

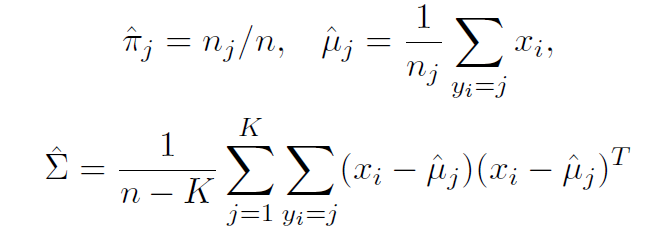
### \_stats\_wilks

### fit

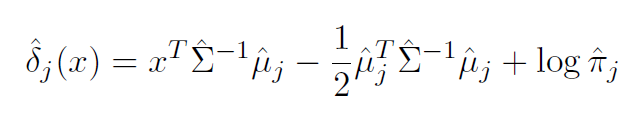
Les valeurs nulles ne sont pas prises en compte et sont supprimées. Le message affiche le nombre d’observations supprimées. La methode renvoie les coefficients et l’intercept du modèle.

D’abord on calcule les moyennes conditionelles, les frequences relatives et la matrice de variance co-variance intra-classes[[1]](#footnote-1) :

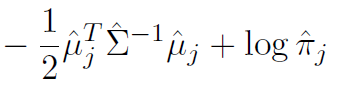




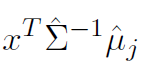
Apres on estime la fonction de classement par la formule :



Où l’intercept :



Et les coefficients :



Fonction fit () :

Sélection des valeurs des variables explicatives et cible

Vérification et suppression des valeurs nulles de dataset

Transformation des données en numpy

              Calcul des effectifs et des fréquences de classes

Calcul des moyennes conditionnelles

Si le déterminant de la matrice de var-covar intra-classes (W) est différent de zero :

Calcul de la matrice inverse de W (invW)

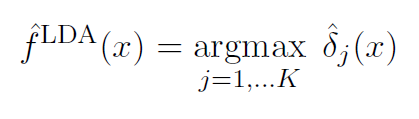
Calcul des coefficients et de l’intercept

Récupération des valeurs de la fonction de classement dans l’attribut infoFuncClassement

Sinon : renvoyer un message d’erreur que la matrice n’est pas inversible

Fin Fonction

### predict

Pour calculer les prédictions il faut prendre le maximum parmi les scores d’appartenance à chaque classe. Ainsi on a finalement la formule pour calculer les prédictions : 

La fonction retourne le vecteur des classes prédites.

Fonction predict(inputData : matrice des valeurs à prédire) :

Création d’une liste vide pour la remplir avec les prédictions

Parcours la matrice inputData

Calcul des probabilités d’appartenir aux classes

Stockage de ces valeurs dans une liste

Prédictions des classes en choisissant le maximum parmi les probabilités

Fin Fonction

### confusion\_matrix

Avoir les prédictions c’est très bien, mais comment on peut savoir ou plutôt mesurer si ces prédictions sont correctes ou pas ? La matrice de confusion et le taux de précision y aident. La première « permet d’obtenir une évaluation non biaisée des performances du modèle en déploiement »[[2]](#footnote-2). Elle prend le vecteur des vraies valeurs de la variable cible qui n’étaient pas utilisées lors de l’apprentissage, et le vecteur des valeurs cibles prédites. Ainsi on peut estimer si la fonction est capable de bien prédire.

« Chaque ligne correspond à une classe réelle, chaque colonne correspond à une classe estimée. La cellule ligne L, colonne C contient le nombre d'éléments de la classe réelle L qui ont été estimés comme appartenant à la classe C »[[3]](#footnote-3).

Cette matrice permet de mesurer plusieurs d’autres indices. Mais comme cela n’est pas l’objectif de notre projet, on a retenu que la matrice de confusion et le taux de précision.

La fonction renvoie la matrice numérique.

En analyse des données c’est toujours utile de construire des graphiques ou d’avoir les données en format plus lisible. Le paramètre de la méthode graphShow est mis à True par défaut, cela nous affiche la matrice en mode plus représentatif (grâce à la librairie seaborn). Si on ne veut pas l’afficher, il faut mettre ce paramètre à False.

Fonction confusion\_matrix(y\_true : vecteur des vrais valeurs de la cible,

y\_pred : vecteur des valeurs de la cible à prédire,

graphShow=True)

Conversion des noms des classes en numérique

Création d’une matrice zéro de taille K x K

Parcours des indices de 0 à K-1 (deux boucles imbriquées) :

Calcul des valeurs de la matrice de confusion

Si le paramètre graphShow est True :

Créer et afficher la matrice de confusion en mode graphique

Renvoyer la matrice de confusion

Fin Fonction

### accuracy\_score

La fonction prend en entrée les vraies valeurs de la variable cible et les prédictions et renvoie la proportion des prédictions correctes.

Accuracy = (True Positive + True Negative) / (Positives + Negatives)

Fonction accuracy\_score(y\_true : vecteur des vrais valeurs de la cible,

y\_pred : vecteur des valeurs de la cible à prédire):

Conversion des noms des classes en numérique

Calcul du taux de précision

Renvoyer le taux de précision

Fin Fonction

### wilks\_decay

### stepdisc

# Pseudo-code

#if we leave pseudo code under each corresponding methode, this section to be deleted

Fonction freq\_relat(y, n; taille de l'échantillon)

Fin Fonction

Fonction means\_class(X,y)

Fin Fonction

Fonction cov\_matrix(DATA)

Fin Fonction

Fonction pooled\_cov\_matrix(DATA, CLASS)

Fin Fonction

Fonction wilks(Vb, Wb)

Fin Fonction

Fonction wilks\_log(Vb, Wb)

Fin Fonction

Fonction wilks\_decay(VAR, Vb, Wb)

Fin Fonction

Fonction p\_value(F, ddl1, ddl2)

Fin Fonction

Fonction createWebFile(filename)

Fin Fonction

**class LinearDiscriminantAnalysis()**

Fonction variables\_explicatives(x)

Fin Fonction

Fonction create\_HTML()

Fin Fonction

1. https://www.stat.cmu.edu/~ryantibs/datamining/lectures/21-clas2.pdf [↑](#footnote-ref-1)
2. http://eric.univ-lyon2.fr/~ricco/cours/cours/Pratique\_Analyse\_Discriminante\_Lineaire.pdf [↑](#footnote-ref-2)
3. https://fr.wikipedia.org/wiki/Matrice\_de\_confusion [↑](#footnote-ref-3)