TP préparation et visualisation des données

Introduction

Dans ce TP, vous étudierez les notes obtenues pour chaque UE par les étudiants de 3ème année de l'ESEO. L'objectif est d'analyser ces notes pour en déduire différentes informations. Les données sont fournies au format CSV. Les deux premières colonnes sont le nom et le prénom (tous les deux pseudonymisés), et toutes les colonnes restantes correspondent à une UE. Chaque ligne (hormis la première ligne d'entête) correspond aux résultats d'un étudiant.

Partie 1 : préparation des données

L'objectif de cette partie est de « mettre au propre » les données pour pouvoir les exploiter correctement dans les séances suivantes

a) Créez un dossier dans lequel vous réaliserez votre TP. Dans celui-ci, créez un script Python et ajoutez le fichier « notes_E3e.csv » que vous allez étudier
b) Commencez par charger les données du CSV dans un dataframe pandas. Attention, comme le CSV a été créé depuis Excel, le séparateur n'est pas la virgule mais le point-virgule. Affichez votre dataframe pour vérifier que vos données ont bien été chargées.
c) Créez la fonction « affichage_donnees » qui prend en paramètre un dataframe et qui affiche :
    - Les informations générales du dataframe
    - Le dataframe lui-même
d) Testez cette fonction sur votre dataframe. Quels sont les types de données trouvé par pandas ?
e) Ensuite faites un premier nettoyage des données en supprimant toutes les lignes contenant une case vide. Affichez votre dataframe et vérifiez que le nombre de lignes a diminué.
f) Créez la fonction « formatage_donnees » qui prend en paramètre un dataframe et qui renvoie ce même dataframe qui aura subi des modifications. L'objectif de cette fonction est de convertir les notes du format français (avec une virgule comme séparateur décimal) au format américain (avec un point comme séparateur décimal). A cause de cette virgule, toutes les notes sont considérées comme des chaines de caractères, ce qui nous empêche tout calcul mathématique. Cette fonction doit donc effectuer :
    - Pour chaque UE :
        - Créer un tableau de note de vide de la bonne taille pour accueillir les notes corrigées
        - Pour chaque note
            - Remplacer la virgule par un point dans la chaine de caractères
            - Convertir cette chaine de caractères en un nombre décimal
            - Ranger cette note dans le tableau créé précédemment
        - Remplacer la colonne du dataframe par le nouveau tableau
    - Relancer sur le dataframe l'inférence du type de chaque colonne
    - Enfin renvoyer le dataframe corrigé
g) Faites la fonction « suppression_inutiles » qui supprime les lignes et les colonnes inutiles à l'analyse et qui renvoie le dataframe mis à jour. Une ligne est considérée comme inutile si au moins une des notes de la ligne est un zéro (cela correspond à un étudiant n'ayant pas terminé l'année). Une colonne correspondant à une UE est considérée comme inutile si tous les étudiants ont obtenu la même note. Pour vérifier cela vous calculerez la différence entre le minimum et le maximum de la colonne et supprimerez celle-ci si la différence est trop faible. Terminez votre fonction par un reset_index
h) Pour finir, vous allez créer la fonction « extraction_infos_series » qui extrait les informations principales d'une série de données et qui les renvoie sous la forme d'un tableau. Les informations nécessaires sont les suivantes :
    - Nom de la série

- Minimum
- Quartile 1
- Médiane
- Quartile 3
- Maximum
- Moyenne
- Ecart-type
- Ecart-type relatif

i) Appelez cette fonction dans une boucle pour récupérer les infos de chaque matière et stockez les résultats dans un dataframe pour obtenir le résultat suivant :

	Matiere	Minimum	Q1	Mediane	Q3	Maximum	Moyenne	Ecart-type	ETR
0	Programmation Java 1	6.083	10.000	12.019	14.783	18.867	12.738015	2.663477	20.909673
1	Programmation Java 2	7.174	10.000	12.000	14.853	19.303	12.534874	2.602234	20.759952
2	Developpement Web	9.995	10.919	13.096	14.855	18.698	13.153610	2.426194	18.445082
3	Infrastructure et reseaux	9.625	12.375	14.500	15.925	19.063	14.239457	2.189131	15.373696
4	Probabilites	1.000	8.375	10.000	12.000	20.000	10.104684	3.580183	35.430924
5	Electronique analogique et numerique	8.812	11.171	12.409	14.041	17.271	12.614327	1.771743	14.045478
6	Systemes embarques 1	9.560	11.919	13.265	14.589	18.332	13.284115	1.840329	13.853603
7	Systemes embarques 2	5.290	11.000	12.500	14.330	19.500	12.635836	2.256556	17.858385
8	Signaux et systemes	9.059	10.759	11.674	12.630	15.656	11.761145	1.280407	10.886751
9	Traitement du signal	8.518	10.000	10.336	12.087	15.622	11.143606	1.438782	12.911274
10	Radio-frequence	10.000	12.167	13.083	13.917	17.417	13.111732	1.253572	9.560689
11	Communication et langues 1	8.885	11.787	13.044	14.374	17.118	12.978472	1.786176	13.762605
12	Communication et langues 2	7.213	13.158	14.795	16.072	18.120	14.569063	1.973816	13.547997
13	Sciences de l'Homme	10.000	13.250	14.750	15.800	18.630	14.398587	1.915207	13.301355
14	Seminaires	8.633	11.133	12.263	13.300	17.667	12.315859	1.589613	12.907040
15	Projet Professionnel	9.650	13.400	14.300	15.000	16.500	14.064461	1.349200	9.592976

Partie 2 : visualisation des données

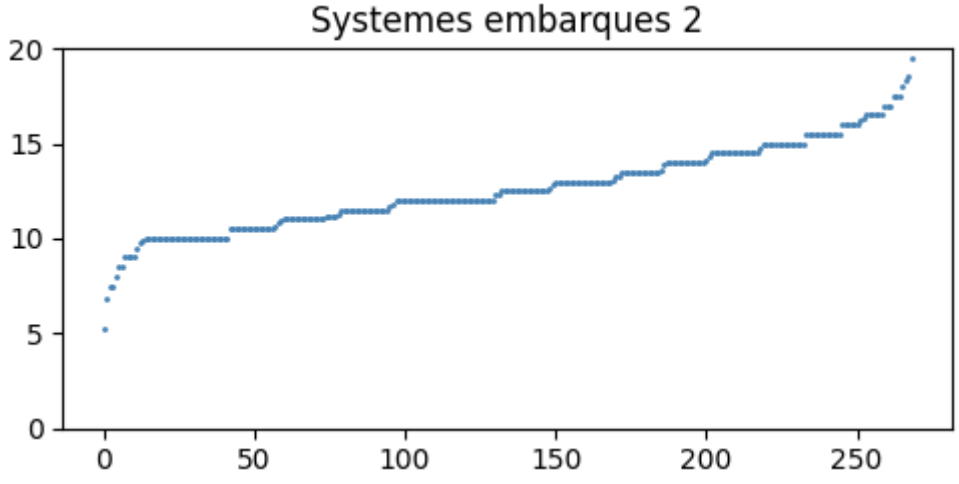
L'objectif de cette partie est de visualiser les données des notes de la manière la plus pertinente possible. Pour vous simplifier la tâche, vous ne travaillerez que sur une seule matière dans un premier temps, et vous généraliserez ensuite le processus à l'ensemble des matières

a) Commencez par faire une fonction "affichage_nuage_points" qui prend en paramètre une dataseries (une colonne de votre dataframe) et qui l'affiche sous la forme d'un nuage de points. Dans la fonction, vous effectuerez les étapes suivantes :
   a. Affichage de la série en paramétrant la taille et la couleur des points, ainsi que l'absence de lignes
   b. L'utilisation du titre de la série comme titre du graphique
   c. Définition de l'intervalle [0;20] comme valeurs possibles
   d. Et pour finir, le lancement de l'affichage

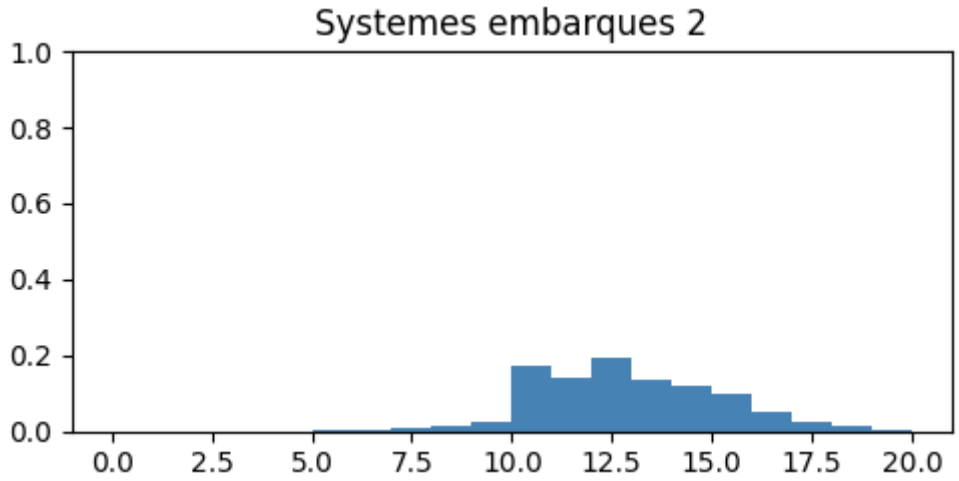
b) Exécutez cette fonction sur une des colonnes de votre dataframe. Qu'observez-vous ?

c) Créez la fonction "tri_serie" qui prend en paramètre une série et qui la renvoie en version triée. Pour cela, vous utiliserez la fonction "sort_values" qui effectue le tri. Ensuite, vous devrez faire un "reset_index", sinon les numéros de lignes seront dans le désordre.

d) Refaites l'affichage sur la version triée de votre série.

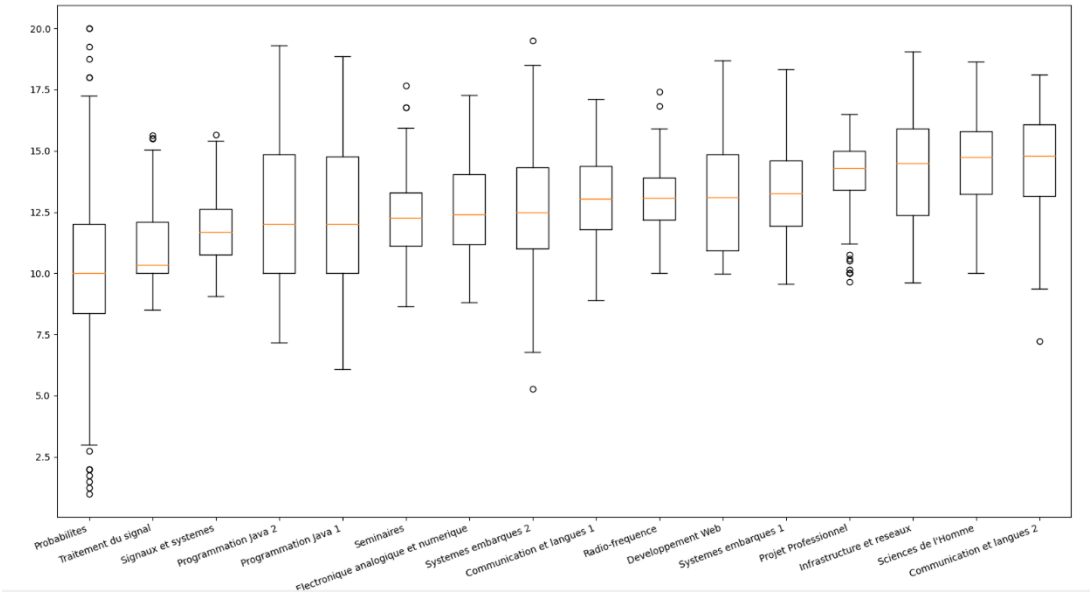


e) Faites la fonction “affichage_histogramme” qui prend en paramètre une dataserie et qui affiche son histogramme. Définissez les catégories comme étant les entiers entre 0 et 20 et le titre du graphique comme le nom de la série. Améliiorez ensuite votre résultat en utilisant le paramètre “weight” de la fonction “hist” pour normaliser votre série (les barres de l’histogramme représentent alors un pourcentage des notes)



f) Appelez les fonctions précédentes dans des boucles pour afficher les résultats pour l’ensemble des matières

g) Faites la fonction “affichage_boite_a_moustaches” qui prend en paramètre un dataframe et qui affiche la boîte à moustache de chaque colonne dans le même graphique. Améliorez l’affichage en triant d’abord les matières par médianes croissante



Partie 3 : approximation par courbes

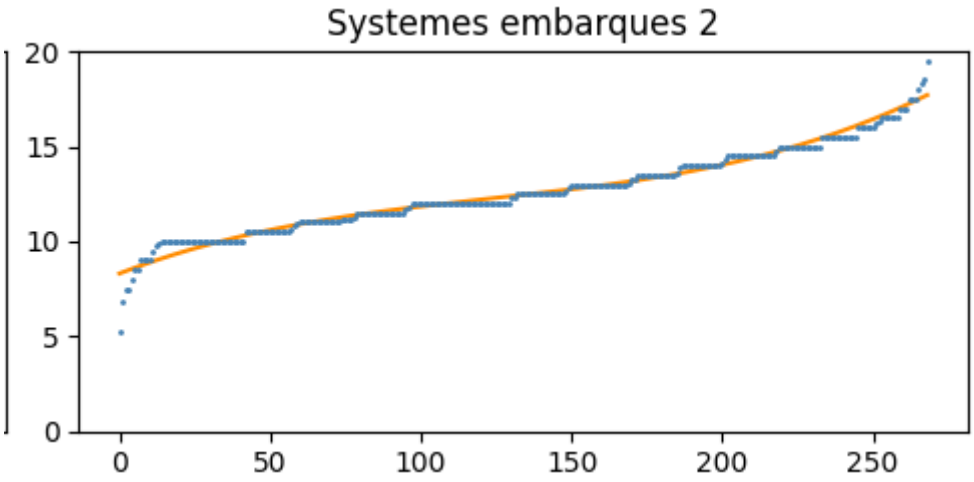
L’objectif de cette partie de la séance est de créer des fonctions permettant de générer des courbes représentatives de nos données.

a) Commencez par définir la fonction “approximation_polynomiale” qui prend en entrée :
   a. X : une liste de x
   b. Y : une liste de y (de même taille que X !)
   c. degre : qui est le degré du polynôme que vous allez utiliser comme approximation

b) Dans cette fonction, appelez la fonction “numpy.polyfit” qui prend exactement les mêmes paramètres et qui renvoie un tableau de coefficients du polynôme

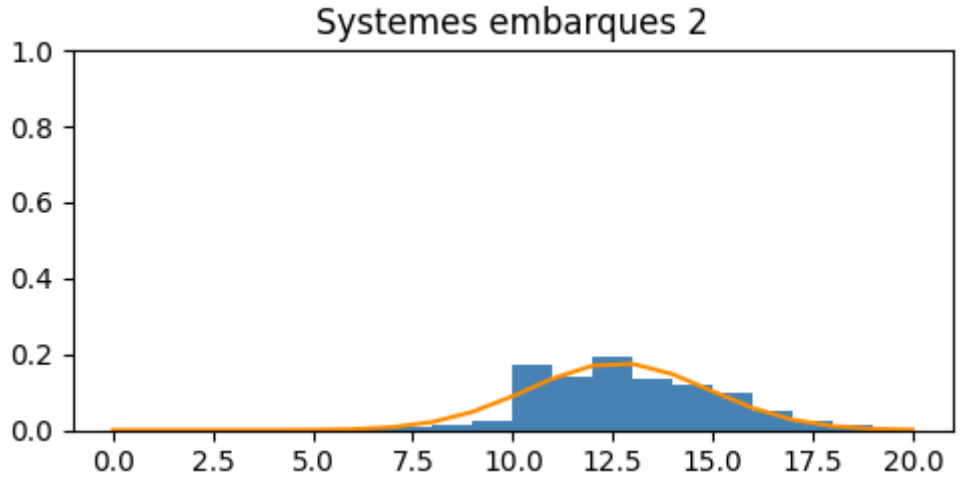
c) Inversez l'ordre des coefficients dans le tableau pour vous simplifier la vie
d) Ensuite, générez un tableau Y_approx qui contient les valeurs f(x) pour chaque x de X (avec f le polynôme que vous avez calculé dans les étapes précédentes
e) Enfin, renvoyez Y_approx

f) Modifiez votre fonction "affichage_nuage_points" pour qu'elle calcule et affiche un polynôme de degré 3 en plus de votre nuage de points. Appelez cette fonction sur une matière pour vérifier que votre polynôme est une bonne approximation de la courbe.



g) Faites la fonction "approximation_gaussienne" qui prend en paramètre une serie de données et une liste de x pour la génération de la courbe.
h) A partir de la série, calculez la moyenne et l'écart-type
i) Ensuite, générez un tableau Y_approx qui contient les valeurs f(x) pour chaque x de X (avec f la formule de la gaussienne que vous pouvez trouver dans le cours et dont vous avez déjà calculé les paramètres)
j) Enfin, renvoyez Y_approx

k) Modifiez votre fonction "affichage_histogramme" pour qu'elle calcule et affiche une gaussienne en plus de vos barres. Appelez cette fonction sur une matière pour vérifier que votre gaussienne est une bonne approximation votre histogramme.



Partie 4 : corrélations

a) Faites la fonction “matrice_correlation” qui prend en paramètre un dataframe et qui affiche la matrice de corrélation de ce dataframe
b) Appelez cette fonction sur le dataframe des notes. Quelles corrélations sont mises en évidence ?
c) Faites la fonction “liste_correlation” qui prend en paramètre un dataframe et un seuil de corrélation et qui renvoie la liste des séries du dataframe dont la corrélation avec au moins une autre série est supérieure au seuil. Pour cela, vous commencerez par déclarer une liste vide. Ensuite, vous calculerez la matrice de corrélation, et vous la parcourrez en regardant chaque corrélation. Si celle-ci est supérieure au seuil, vous ajouterez le nom des deux séries dans la liste. Avant de renvoyer la liste, vous supprimerez les doublons avec la fonction set().
d) Appelez la fonction précédente sur votre dataframe, puis utilisez la liste obtenue pour extraire un sous-dataframe ne contenant que les colonnes ayant des corrélations intéressantes. Terminez par afficher la matrice de corrélation de ce sous-dataframe
e) Faites la fonction « graphe_correlation » qui prend en paramètre un dataframe et un seuil de corrélation et qui renvoie la liste des arêtes du graphe de corrélation (i.e. les paires de séries dont la corrélation est supérieure au seuil)
f) Faites la fonction « composantes_connexes » qui prend en paramètre le résultat de la fonction précédente et qui renvoie une liste de composantes connexes. Une composante connexe est une liste de sommets qui sont tous « accessibles » depuis chaque sommet de la composante en suivant les arêtes.
g) Utilisez les fonctions précédentes pour extraire des sous-dataframes de composantes connexes de votre premier dataframe et afficher les matrices de corrélations de chacune de ces composantes. Ajustez le seuil jusqu’à trouver un résultat qui vous semble convenable (pas trop de séries dans chaque matrice, mais quand même suffisamment d’information)

Partie 5 : Analyse en composantes principales

a) Faites la fonction “affichage_covariance” qui prend en paramètre un dataframe et qui affiche la matrice de covariance de ce dataframe. Que remarquez-vous par rapport à la matrice de corrélation ?
b) Faites la fonction « extraction_par_ACP » qui prend en paramètre un dataframe et un entier. Dans un premier temps, cette fonction applique l’ACP sur le dataframe et affiche les coefficients obtenus. Dans un second temps, ces coefficients sont appliqués sur chaque ligne du dataframe pour extraire les n premières dimension du résultat de l’ACP et les renvoyer sous la forme d’un dataframe.
c) Testez votre fonction et affichez la matrice de corrélation de votre résultat. Que remarquez-vous ?