Intel·ligència Artificial Avançada

PAC 2

Fidel Bonet Vilela

Màster Universitari d'Enginyeria Informàtica

2018-2019, 2n semestre

ÍNDEX

Exercici 1

Apliqueu una anàlisi PCA a les dades de la PAC (recordeu que les dades inclouen valors no vàlids):

a) ***Quantes components principals són necessàries per representar un 95% de la variança de les dades originals?***

Per a poder respondre aquesta pregunta s'han implementat dues funcions:

- *read_and_process_data*:

  Aquesta funció rep el nom de l'arxiu *dermatology.data* i retorna tant els atributs com les classes en dos *arrays* diferents.

  Abans de retornar els valors dels atributs es substitueixen els valors absents marcats amb el caràcter "?" per valors nuls per, a continuació, substituir-los pel valor mitjà de la columna on es troben utilitzant el mètode *SimpleImputer* de la biblioteca *Scikit-learn*.

- *exercise1*:

  Aquesta funció rep els atributs i les classes obtinguts en l'anterior funció i hi aplica una anàlisi PCA.

  En primer lloc, estandarditza les dades ja que és important garantir que les variables presentin un rang de valors comparable abans de realitzar l'anàlisi de components principals evitant que els resultats puguin dependre de factors d'escala. Tot seguit, s'han obtingut 34 components principals ja que és el nombre d'atributs del conjunt de dades original. Finalment, es mostra per la sortida estàndard la variància explicada acumulada.

La taula 1 mostra la variància explicada acumulada obtinguda. Com es pot apreciar, són necessaris 22 components principals per representar un 95% de la variància de les dades originals.

Número de components principals	Variància explicada acumulada
1	0,272
2	0,433
3	0,524
4	0,590
5	0,629
6	0,665
7	0,695
8	0,724
9	0,749
10	0,774
11	0,798
12	0,819
13	0,840
14	0,859
15	0,876
16	0,893
17	0,907
18	0,920
19	0,931
20	0,941
21	0,949
22	**0,957**
23	0,965
24	0,971
25	0,977
26	0,981
27	0,984
28	0,987
29	0,990
30	0,993
31	0,995
32	0,997
33	0,999
34	1,000

Taula 1. Variància explicada acumulada.

b) ***Reconstruïu el conjunt de dades dels 2, 4 i 8 components principals (mitjançant el mètode inverse_transform), i calculeu la pèrdua d'informació respecte al conjunt original per a cada cas. Per a fer-ho, podeu calcular el promig de les diferències elevades al quadrat entre cada element del conjunt reconstruït i l'original. Quina relació tenen aquests valors respecte a les variances acumulades calculades a l'apartat anterior?***

Per a realitzar aquest apartat s'ha continuat amb la implementació de la funció exercise1. Concretament, s'han realitzat anàlisis de components principals de les dades originals per a 2, 4 i 8 components principals. Un cop obtingudes aquestes components principals, s'ha utilitzat la funció *inverse_transform* de la biblioteca *scikit-learn* per a reconstruir el conjunt de dades dels 2, 4 i 8 components principals respectivament.

Un cop reconstruïts els conjunt de dades s'han obtingut els quadrats de les diferències dels elements dels conjunts original i reconstruït amb la finalitat de calcular la pèrdua d'informació en cadascun dels casos. La següent taula mostra els resultats obtinguts.

Components principals	Pèrdua d'informació	Variància explicada acumulada
2	0,567	0,433
4	0,410	0,590
8	0,276	0,724

Taula 2. Pèrdua d'informació dels conjunts de dades reconstruïts.

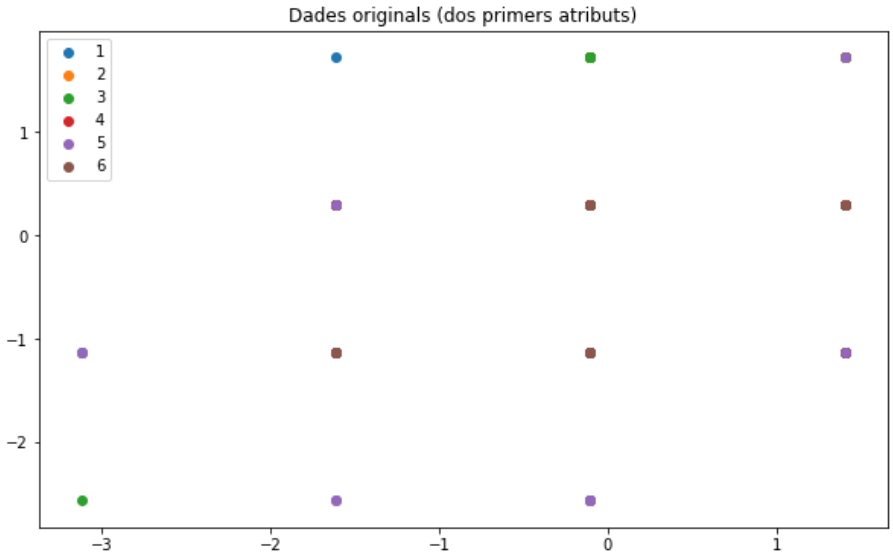
En tots els casos, la projecció de les dades en un subconjunt dels eixos principals simplifica les dades ja que aquestes són representades en un espai de dimensionalitat reduïda. Com s'aprecia en l'anterior taula, en augmentar els components principals utilitzats augmenta el percentatge de variabilitat de les dades. És a dir, quants menys components principals s'utilitzen per a la reconstrucció del conjunt de dades més se simplifiquen aquestes i, per tant, hi ha més pèrdua d'informació. Per contra, en augmentar els components principals utilitzats per a la projecció de les dades, augmenta la variància acumulada i no existeix tanta pèrdua d'informació.

En la taula 2 s'aprecia com hi ha una relació directa entre la pèrdua d'informació (segons el càlcul dut a terme) i la variància explicada acumulada, ja que en augmentar una disminueix l'altra en la mateixa proporció. De fet, la suma d'ambdós paràmetres en els tres casos (per a 2, 4 i 8 components principals) dona 1. Per tant, la relació entre ambdós es manté constant.

c) ***Visualitzeu les dades originals (2 primers atributs) i les dades transformades segons els dos components principals fent servir un color diferent per cada classe. Comenteu el resultat.***

Per implementar aquesta part de l'exercici s'han generat dos gràfics amb la biblioteca *matplotlib*. En ambdós casos es treballa amb dades escalades.

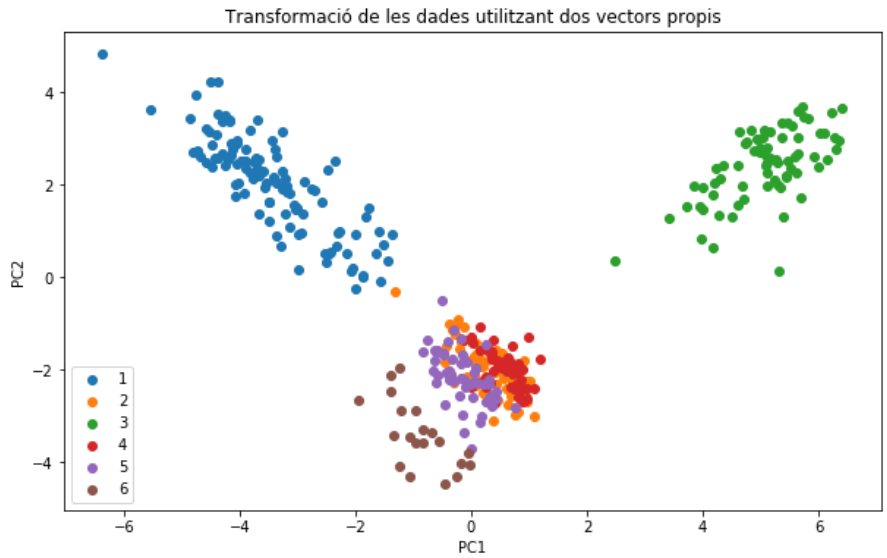
El primer d'aquests gràfics mostra els dos primers atribut de les dades originals en un espai de dos dimensions. S'ha utilitzat un gràfic de punts de colors per a representat aquests elements i les seves 6 classes.



Gràfic 1. Dades originals (dos primers atributs).

En aquest primer gràfic és difícil extreure informació del conjunt d'exemples ja que els dos primers atributs no ens ho permeten. De fet, ni tan sols poden diferenciar clarament les sis classes contingudes en el conjunt.

El segon gràfic mostra la projecció de les dades a al nova base definida pels els dos components principals, és a dir, en un espai PCA 2D.



Gràfic 2. Projecció de les dades en els dos primers components principals PCA.

En aquest cas, la projecció de les dades en l'espai PCA 2D sí que ens permet obtenir informació del conjunt d'exemples ja que hi podem identificar clarament les sis classes del conjunt.

Per altra banda, el fet que la distribució de les dades originals i de les dades projectades en l'espai PCA 2D sigui tan diferent és degut a què en el primer cas tan sols es representen els dos primers atributs del conjunt d'exemples mentre que en el segon es representen dos components principals obtinguts a partir de tots els atributs originals. Per tant, en el segon gràfic no s'estan reconstruint les dades del gràfic 1 sinó del conjunt de les dades originals amb un 43% de la seva variància explicada acumulada.

Exercici 2

En aquest exercici compararem els resultats d'aplicar un classificador simple amb les dades originals i les dades d'acord amb les característiques obtingudes pel PCA fent servir K fold cross-validation (k=8):

a) *Apliqueu KNeighborsClassifier, amb k=3, sobre el conjunt de dades originals*

Per a implementar aquest apartat en primer lloc s'han estandarditzat les dades originals. Tot seguit s'han obtingut els conjunts d'entrenament i de test per al classificador utilitzat el 33% dels exemples per al conjunt de test i la resta per al conjunt d'entrenament. A continuació, s'ha obtingut el model kNN mitjançant el mòdul *KNeighborsClassifier* de la biblioteca *scikit-learn* amb tres veïns més propers i la distància euclidiana com a mesurà de distància.

Un cop obtingut el model, aquest s'ha entrenat amb vuit subconjunts utilitzant validació creuada per avaluar-lo. La següent taula mostra les precisions obtingudes en cadascun dels vuit subconjunts.

Subconjunt	Precisió (%)
1	97,0
2	90,9
3	100,0
4	93,7
5	96,6
6	96,6
7	100,0
8	100,0

Taula 3. Precisió obtinguda per a cadascun dels subconjunts de la validació creuada.

La precisió mitjana ha estat del 96,8%.

b) *Apliqueu KNeighborsClassifier, amb k=3, sobre el conjunt de dades transformades pel PCA amb 8 components*

Per a implementar aquest apartat s'ha partit de les dades estandarditzades de les quals s'han obtingut els 8 components principals i s'ha projectat en el nou espai PCA 8D amb 8 components principals.

Tot seguit, s'ha utilitzat aquest nou conjunt de dades reconstruït per a obtenir els conjunts d'entrenament i de test per al classificador utilitzat el 33% de dels exemples per al conjunt de test i la resta per al conjunt d'entrenament. A continuació, s'ha obtingut el model kNN mitjançant el mòdul *KNeighborsClassifier* de la biblioteca *scikit-learn* com en l'apartat anterior.

Un cop obtingut el model, aquest s'ha entrenat amb vuit subconjunts utilitzant validació creuada per avaluar-lo. La següent taula mostra les precisions obtingudes en cadascun dels vuit subconjunts.

Subconjunts	Precisió (%)
1	97,0
2	90,9
3	90,6
4	100,0
5	100,0
6	90,0
7	96,7
8	92,6

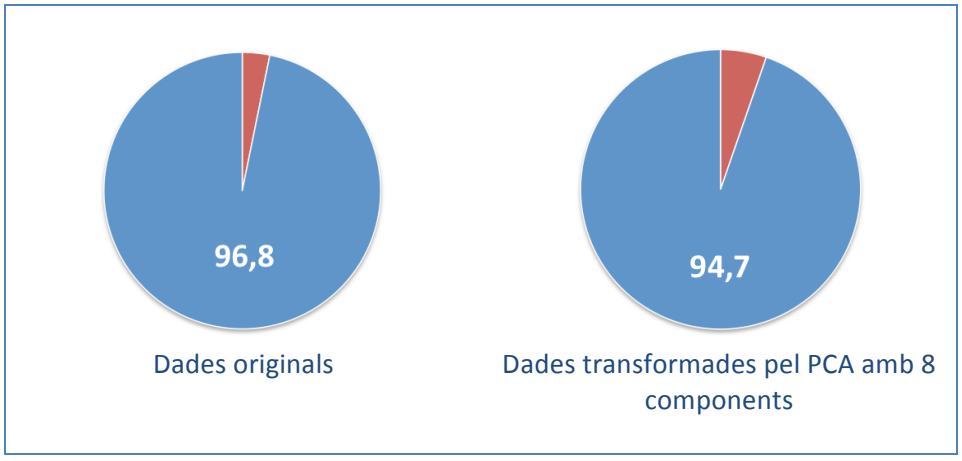
Taula 4. Precisió obtinguda per a cadascun dels subconjunts de la validació creuada.

La precisió mitjana ha estat del 94,7%.

c) *Comenteu els resultats obtinguts*

S'ha obtingut una precisió mitjana en l'avaluació creuada superior en el cas d'utilitzat les dades original que en el cas d'utilitzar les dades transformades.

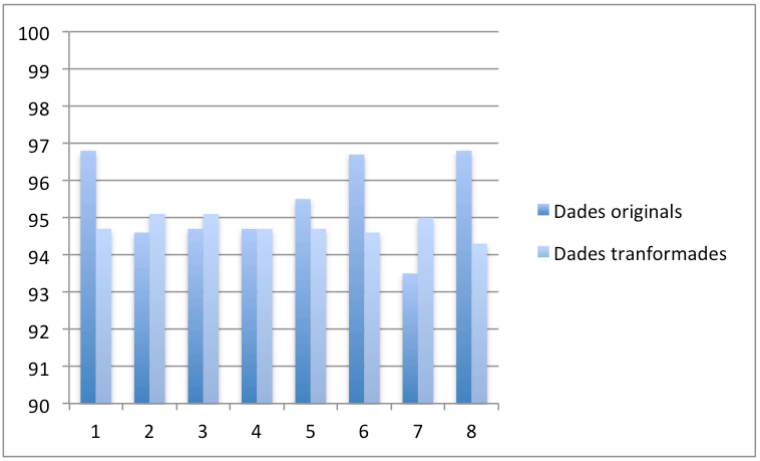
Els següents gràfics mostren el percentatge de precisió obtingut amb l'ús de les dades originals i amb l'ús de des dades transformades pel PCA amb vuit components principals.



Gràfic 3. Precisió mitjana per a les dades originals i per a les dades transformades pel PCA.

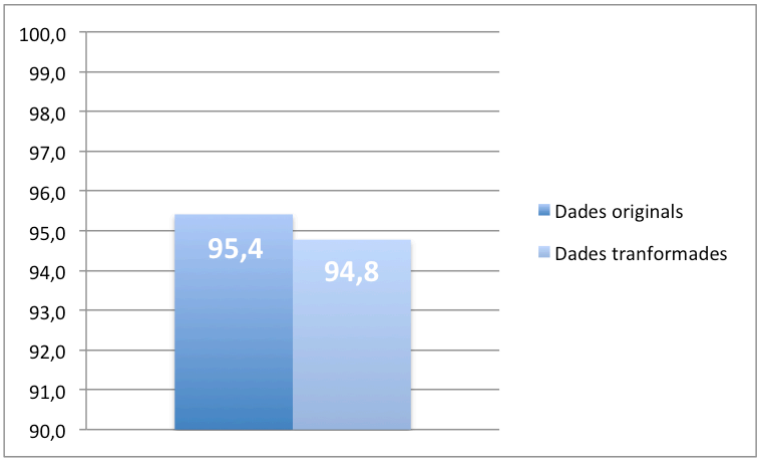
El motiu pel qual la precisió mitjana obtinguda en les dades transformades sigui inferior a l'obtinguda amb les dades originals és que les dades transformades han patit una pèrdua d'informació en reduir-ne la dimensionalitat i, per tant, simplificant aquestes. Concretament l'ús de 8 components principals en el conjunt de dades utilitzat comporta representar el 72,4% de la variància de les dades originals.

Degut al fet que en cada execució de la validació creuada s'escullen vuit subconjunts de dades diferents, el resultat pot variar entre execucions. És per això que s'ha realitzat un conjunt de vit execució per a comprovar que es manté la diferència de precisió obtinguda en tots els casos. El següent gràfic mostra els resultats obtinguts.



Gràfic 4. Resultat d'un conjunt de vuit execucions.

Com es pot observar, alguns cops s'obtenen millors precisions amb les dades originals i en d'altres amb les dades transformades. Tot i així, com es mostra en el gràfic següent la mitjana de les diverses proves realitzades continua indicant que amb les dades originals s'obté més precisió en el classificador tot i que la diferència és molt petita ja que ni tan sols arriba a un punt percentual.



Gràfic 5. Mitjanes.

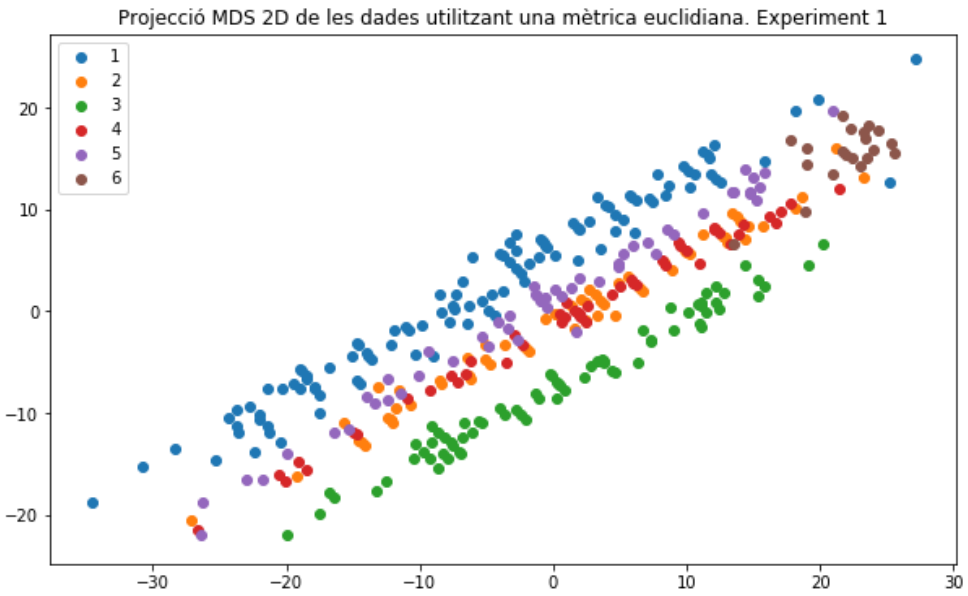
Exercici 3

Apliqueu el mètode "multidimensional scaling" (MDS) a les dades. Repetiu aquest procés 4 cops i mostreu una gràfica en dues dimensions on cada mostra prengui un color diferent segons la seva classe. Comenteu els resultats. Per què cada gràfica és diferent?

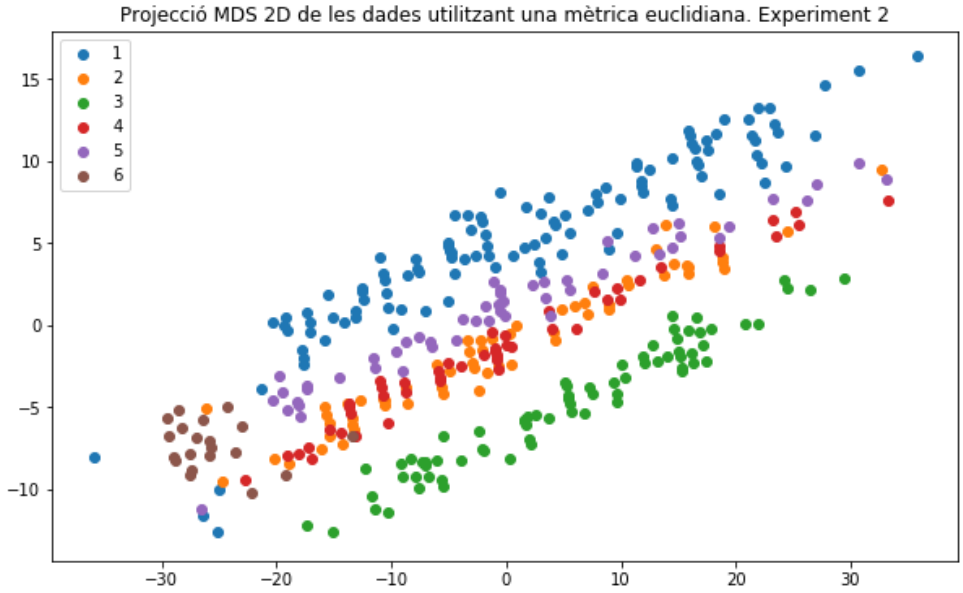
Tenint en compte que no s'ha identificat correlació de les dades en l'espai original, s'ha optat per utilitzar la distància euclidiana com a mètrica.

Per a implementar aquest exercici s'ha utilitzat un recorregut de 4 iteracions per a repetir el procés quatre cops. En cada iteració s'ha generat una llavor nova mitjançant la classe *random* de la biblioteca *numpy* i, tot seguit, s'ha aplicat el mètode d'escalament multidimensional (MDS) sobre les dades originals emprant una mètrica euclidiana per a obtenir les posicions de cada mostra en l'espai 2D.

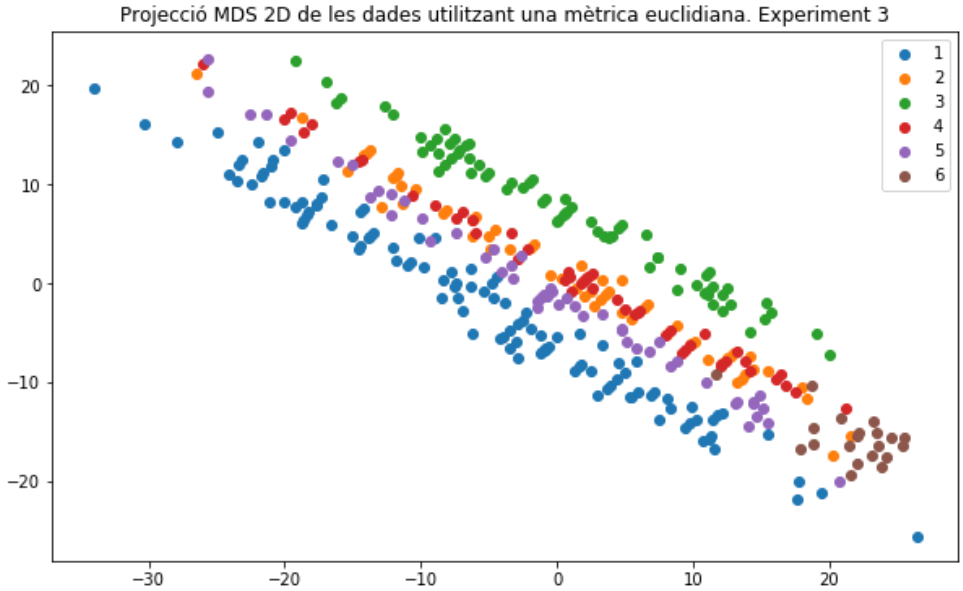
Un cop obtingudes aquestes posicions s'ha realitzat una projecció 2D de les dades multidimensionals assignant un color diferent a cada classe. Les següents quatre gràfiques mostren els resultats obtinguts.



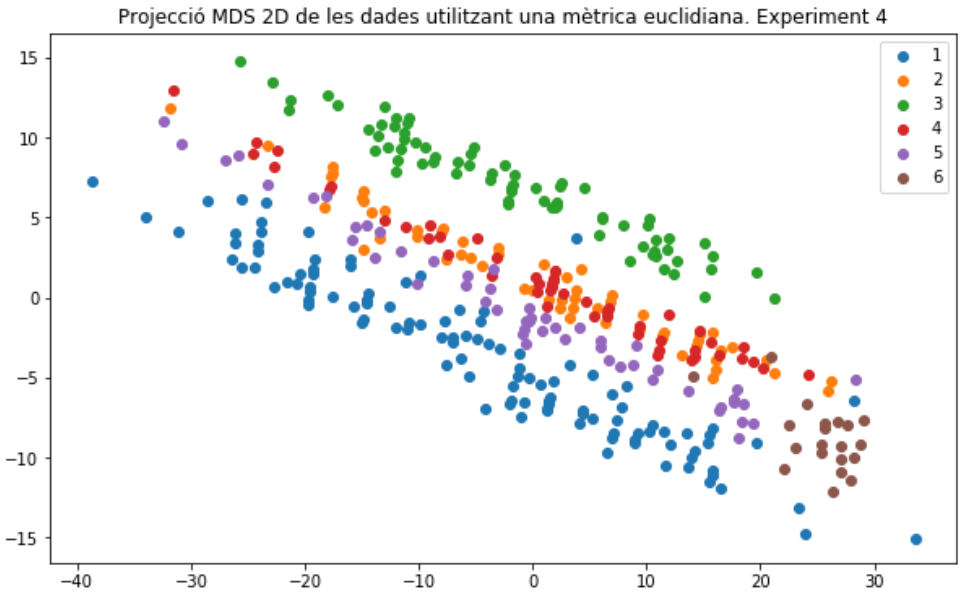
Gràfic 6. Projecció MDS 2D de les dades utilitzant una mètrica euclidiana. Experiment 1.



Gràfic 7. Projecció MDS 2D de les dades utilitzant una mètrica euclidiana. Experiment 2.



Gràfic 8. Projecció MDS 2D de les dades utilitzant una mètrica euclidiana. Experiment 3.



Gràfic 9. Projecció MDS 2D de les dades utilitzant una mètrica euclidiana. Experiment 4.

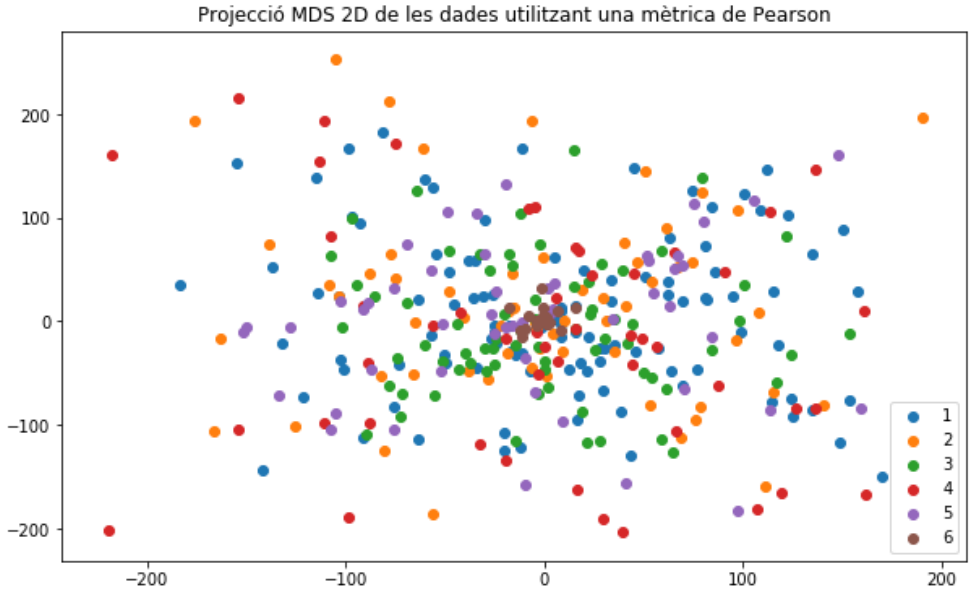
Com es pot observar, en les quatre projeccions 2D de les dades, els sis grups definits en l'espai original formats per les diverses classes existent de 34 dimensions apareixen agrupats en les projeccions MDS 2D. Per tant, s'ha aconseguit l'objectiu d'utilitzat l'escalament multidimensional, que és visualitzar les dades multidimensionals originals en un pla mantenint les distàncies relatives entre els individus del conjunt.

Cada gràfica obtinguda és diferent ja que l'algorisme MDS, un cop generada la matriu de distàncies, distribueix aleatòriament les observacions en un espai bidimensional. Tot seguit es calcula una funció d'error per a cada punt i es corregeix aquesta posició de forma iterativa per a minimitzar l'error respecte a la distància en l'espai original. És la distribució aleatòria a partir de la qual es parteix la que provoca que, tot i que sempre es mantenen agrupades les classes, s'obtinguin resultats diferents cada cop.

Cal tenir en compte que, degut a què els ordinadors binaris no són capaços de generar nombres realment aleatoris sinó que tan sols són capaços de generar nombres pseudo-aleatoris, ha estat necessari canviar la llavor utilitzada per a obtenir aquests valors pseudo-aleatoris en cada iteració. Si s'hagués utilitzat sempre la mateixa llavor s'haurien generat sempre els mateixos nombres pseudo-aleatoris i, per tant, s'haurien obtingut quatre gràfiques idèntiques.

Tot i que s'ha descartat la utilització de Pearson com a mètrica degut a què només és interesant quan les dades presenten correlacions internes entre grups, s'ha realitzat una implementació de prova on s'han utilitzat els coeficients de correlació de Pearson. Per aquesta implementació s'ha reutilitzat el codi dels arxius *scaledown.py* i *pearson.py* proporcionats a l'aula de l'assignatura.

El següent gràfic mostra el resultat obtingut, on s'aprecia clarament que les dades en l'espai projectat no mostren la distribució de les dades en l'espai original.



Gràfic 10. Projecció MDS 2D de les dades utilitzant una mètrica de Pearson.

Exercici 4

Amb els classificadors de scikit-learn i els parámetres indicats a continuació, obteniu l'score, el training time i el prediction time quan s'apliquen a les dades proporcionades fent servir K fold cross-validation (K=8):

Per a implementar aquest exercici, en primer lloc s'han estandarditzat les dades i tot seguit s'han obtingut els conjunts d'entrenament i de test. Tot seguit s'han obtingut vuit *folds* (subconjunts per a la validació creuada).

- k Nearest Neighbors (mòdul KneighborsClassifier de sklearn.neighbors): amb 3, 4 i 5 veïns (primer paràmetre).

Per aquesta part de l'exercici s'ha utilitzat un recorregut per aplicar la classificació al conjunt de dades per a 3, 4 i 5 veïns més propers.

En cadascuna d'aquestes tres iteracions s'ha definit el classificador per 3, 4 i 5 veïns més propers mitjançant el paràmetre *n_neighbors*.

Tot seguit, s'ha implementat un recorregut pels subconjunts de l'avaluació creuada. En cada iteració s'han obtingut els conjunts d'entrenament i de test. Per a calcular posteriorment el temps d'entrenament i de predicció s'ha obtingut el temps mitjançant la funció *time()* abans i després de l'entrenament i de la predicció. Per altra banda, la predicció obtinguda en cada iteració s'ha afegit a una matriu *precisio[]*.

Finalment, es mostra la precisió obtinguda com la mitjana dels resultats de cada subconjunt, el temps d'entrenament i el temps de predicció.

- Linear SVM (mòdul SVC de sklearn.svm): kernel="linear", C=0.025, la resta de paràmetres, valor per defecte.

Un cop definit el classificador amb el mòdul *SVC* i els paràmetres *C=0.025* i *kernel='linear'*, s'ha seguit un procediment similar a l'explicat més amunt per al cas de *k* veïns més propers. En aquest cas, però, no s'ha realitzat un recorregut per a obtenir diverses classificacions ja que només es demanava una classificació amb uns paràmetres.

- Decision Tree (mòdul DecisionTreeClassifier de sklearn.tree): max_depth=5, la resta de paràmetres, valor per defecte.

El classificador s'ha obtingut amb el mòdul *DecisionTreeClassifier* i el paràmetre *max_depth=5*.

- AdaBoost (mòdul AdaBoostClassifier de sklearn.ensemble): paràmetres per defecte.

El classificador s'ha obtingut amb el mòdul *AdaBoostClassifier* i sense cap paràmetre, ja que s'han utilitzat els paràmetres per defecte.

- Gaussian Naive Bayes (mòdul GaussianNB de sklearn.naive_bayes): paràmetres per defecte.

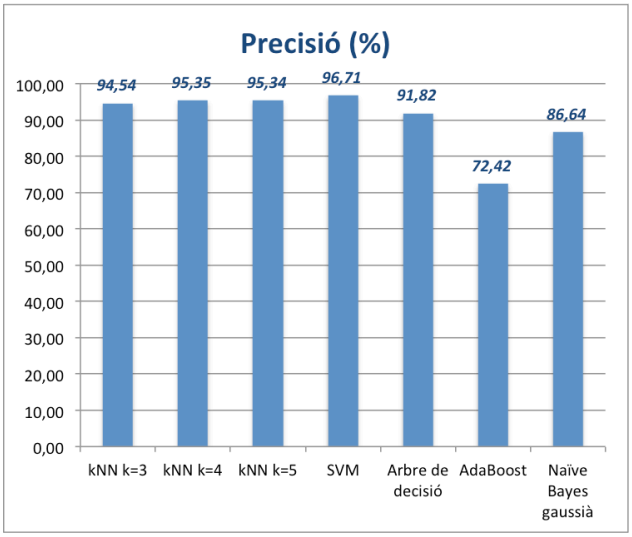
Finalment, aquest darrer classificador s'ha obtingut amb el mòdul *GaussianNB* i no s'ha especificat cap paràmetre ja que, com en el cas anterior, s'han utilitzat els paràmetres per defecte.

La següent taula mostra els resultats pel que fa a la precisió, el temps d'entrenament i el temps de predicció per a cadascun dels classificadors utilitzats.

Classificador	Precisió (%)	Temps entrenament *(segons)*	Temps predicció *(segons)*
kNN k=3	94,535	0,0118	0,0407
kNN k=4	95,350	0,0064	0,0360
kNN k=5	95,344	0,0117	0,0567
SVM	96,709	0,0357	0,0065
Arbre de decisió	91,824	0,0135	0,0029
AdaBoost	72,421	0,6575	0,0774
Naïve Bayes gaussià	86,643	0,0143	0,0071

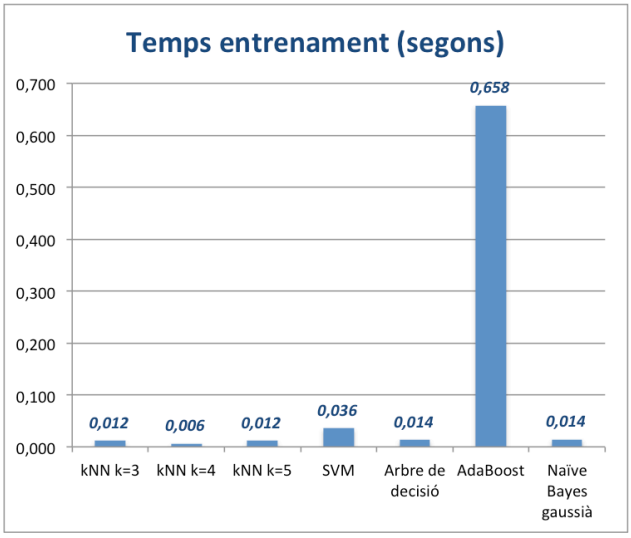
Taula 5. Resultats dels diversos classificadors.

El següent gràfic mostra les precisions obtingudes amb els diversos classificadors emprats. Com es pot observar, el classificador que ha obtingut el millor resultat han estat les màquines de vectors de suport seguides per l'algorisme kNN amb 4, 5 i 3 veïns més propers (en aquest ordre). També s'han obtingut bons resultats amb l'arbre de decisió amb una precisió superior al 91%. S'han obtingut pitjors resultats amb el classificador Naïve Bayes. Finalment, el classificador que ha obtingut pitjors resultats amb el conjunt de dades utilitzat ha estat AdaBoost.



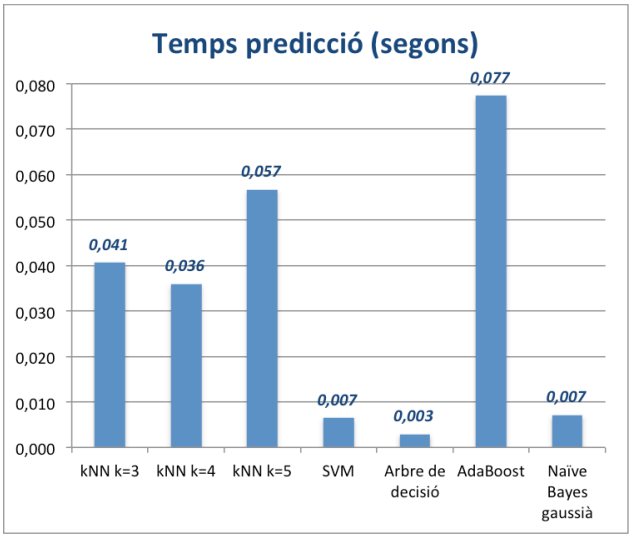
Gràfic 11. Precisions obtingudes.

Per altra banda, el gràfic 12 mostra els temps d'entrenament necessaris en els diversos classificadors utilitzats. Com es pot apreciar, tots els temps d'entrenament estan per sota dels 0,04 segons excepte AdaBoost que ha necessitat més de mig segon per a l'entrenament. De la resta de classificadors, el que ha dut a terme l'entrenament amb menys temps ha estat kNN i concretament en el cas de 4 veïns més propers amb 6 mil·lèsimes de segon.



Gràfic 12. Temps d'entrenament obtinguts.

Finalment, el següent gràfic mostra els temps de predicció obtinguts. Un cop més l'algorisme que ha obtingut els pitjors resultats amb més temps per a la predicció ha estat el classificador AdaBoost. Per contra, l'algorisme amb un temps de predicció menor ha estat l'arbre de decisió seguit de l'SVM i Naïve Bayes. A diferència dels resultats obtinguts per als temps d'entrenament, l'algorisme kNN té uns temps de predicció pitjors que SVM, arbres de decisió i Naïve Bayes.



Gràfic 13. Temps de predicció obtinguts.

Bibliografia

BENÍTEZ, Raúl; CENCERRADO BARRAQUÓ, Andrés; ESCUDERO, Gerard *i altres* (2018). *Intel·ligència artificial avançada* (2a ed.). Barcelona: FUOC.

CENCERRADO BARRAQUÓ, Andrés; MASIP RODÓ, David (2018). *El llenguatge Python* (2a ed.), Barcelona: FUOC.

Matplotlib. [Data de consulta: 25 d'abril de 2019] <https://matplotlib.org/>

NumPy. [Data de consulta: 25 d'abril de 2019] <http://www.numpy.org>

Python 3.7.3rc1 documentation. [Data de consulta: 25 d'abril de 2019] <https://docs.python.org/3/>

Scikit-Learn. [Data de consulta: 25 d'abril de 2019] <https://scikit-learn.org>

Scipy. [Data de consulta: 25 d'abril de 2019] <https://scipy.org>