Introducció

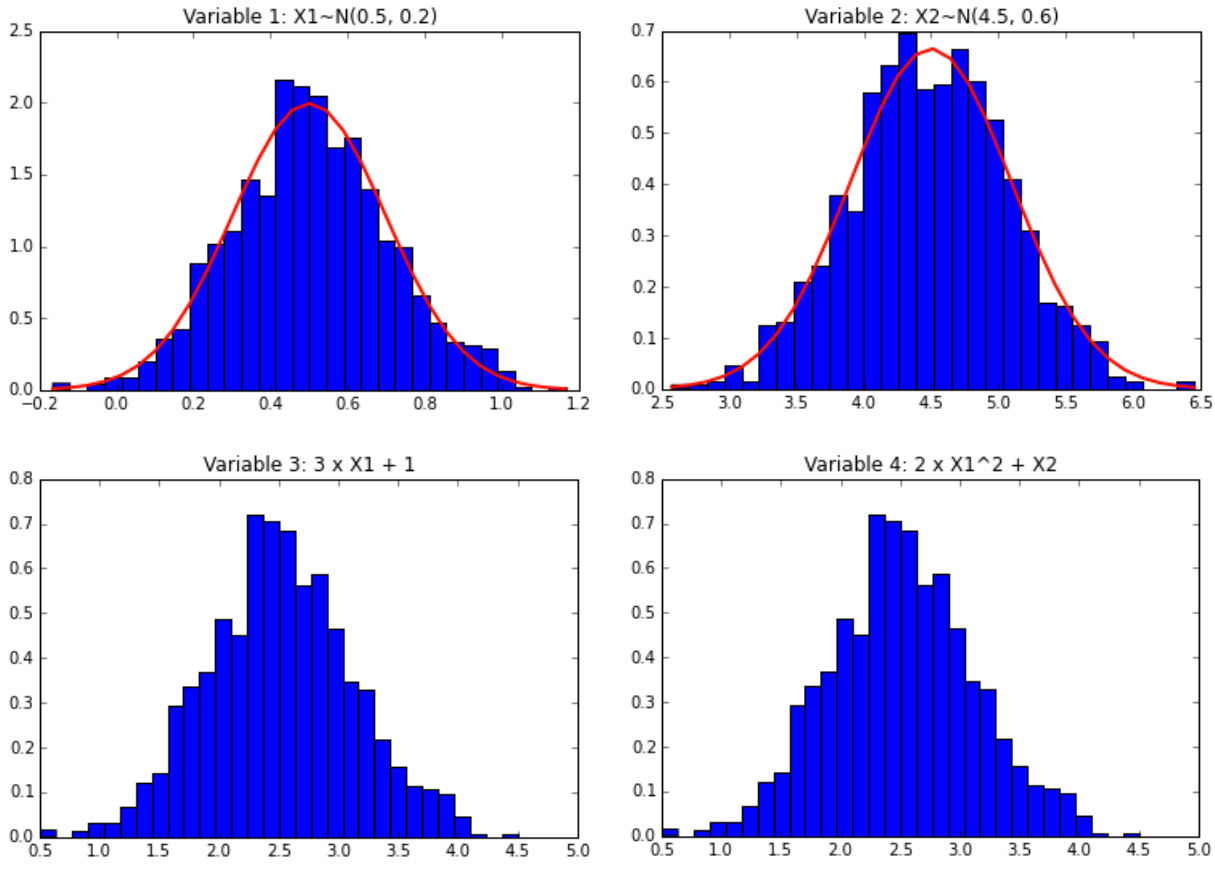
El present informe exposa els procediments i resultats d'execució de la **pràctica final** del curs d'Intel·ligència Artificial Avançada. L'objectiu d'aquesta pràctica és doble: d'una banda, generarem **dades sintètiques** (que no s'han obtingut d'un sistema real) amb unes distribucions determinades i, per tant, amb unes propietats conegudes a priori; d'altra banda, revisarem les tècniques de **reducció de dimensionalitat** i **classificació** utilitzades en la segona i tercera pràctica del curs, respectivament. Per a la realització de la pràctica utilitzarem els mètodes proporcionats per la llibreria **scikit-learn**.

Activitat 1: Reducció de la dimensionalitat

a. Genereu un conjunt de dades sintètiques amb N=1000 observacions i 4 variables. La primera variable (x1) ha de seguir una distribució normal univariada amb mitjana 0.5 i desviació típica 0.2, és a dir x1~N(0.5, 0.2). La segona variables serà x2~N(4.5, 0.6). La tercera variable serà una combinació lineal de la primera x3 = 3*x1 + 1. La quarta, una combinació no lineal de les dues primeres x4 = 2*x12 + x2. Finalment, construïu una matriu de dades A de tamany Nx4 amb les variables x1, x2, x3, x4.

En aquest primer apartat realitzarem la construcció de la matriu sol·licitada utilitzant els mètodes *random* i *concatenate* de la llibreria *numpy*.

Les següents figures recullen la representació gràfica de les 4 variables generades per separat. El codi del programa es recull a l'arxiu *ActivityOneDef.py* i queda documentat després de les figures.



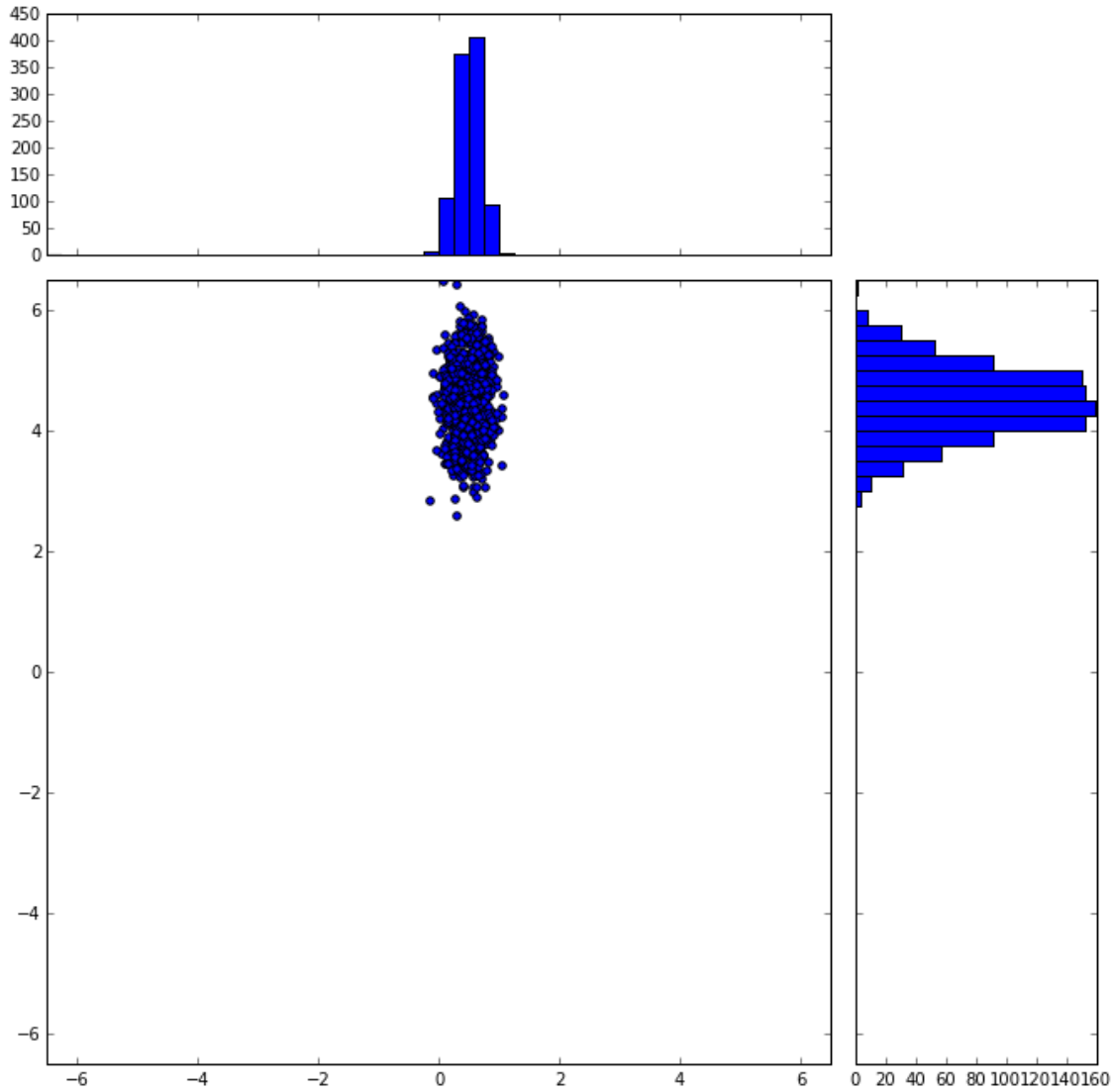
```
    # -*- coding: utf-8 -*-
    """
    @author: Juan Águila Martínez (UOC - 2015)
    """

    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.decomposition import PCA
    from matplotlib.ticker import NullFormatter

    ######################################################################
    # Activitat 1
    ######################################################################

    # Definim els paràmetres de centralitat i dispersió per a la primera varia
    ble
    mu1, sigma1 = 0.5, 0.2
    # i generem un array amb 1000 valors que segueixin una distribució normal
    # parametrada segons els valors anteriors
    s1 = np.random.normal(mu1, sigma1, 1000)
    # Representem el conjunt anterior en un histograma amb 30 bins
    # i grafiquem la funció de distribució normal
    count, bins, ignored = plt.hist(s1, 30, normed=True)
    plt.plot(bins,
             1/(sigma1 * np.sqrt(2 * np.pi)) * np.exp( - (bins -
     mu1)**2 / (2 * sigma1**2) ),
                 linewidth=2, color='r')
    plt.title("Variable 1: X1~N(0.5, 0.2)")
    plt.show()

    # Seguim el mateix procediment per a la segona variable
    mu2, sigma2 = 4.5, 0.6
    s2 = np.random.normal(mu2, sigma2, 1000)
    count, bins, ignored = plt.hist(s2, 30, normed=True)
    plt.plot(bins,
             1/(sigma2 * np.sqrt(2 * np.pi)) * np.exp( - (bins -
     mu2)**2 / (2 * sigma2**2) ),
                 linewidth=2, color='r')
    plt.title("Variable 2: X1~N(0.5, 0.2)")
    plt.show()

    # La tercera variable es construeix com a combinació lineal de la primera
    s3 = 3 * s1 + 1
    count, bins, ignored = plt.hist(s3, 30, normed=True)
    plt.title("Variable 3: 3 x X1 + 1")
plt.show()
    # i la cuarta com a combinació lineal de la primera i la segona
    s4 = 2 * s1**2 + s2
    count, bins, ignored = plt.hist(s3, 30, normed=True)
    plt.title("Variable 4: 2 x X1^2 + X2")
    plt.show()

    # Finalment, utilitzem el mètode `concatenate` de la llibreria numpy per a
    # construir una matriu de dimensió 4x1000
    s = np.concatenate(([s1], [s2], [s3], [s4]), axis=0 ).T
```

b. Per cada combinació de variables, representeu les dades gràficament com a núvols de punts bidimensionals (x1 vs x2, etc.). Podeu fer servir la funció scatter de la llibreria matplotlib.

Les següents figures recullen els diagrames de dispersió de les variables enfrontades una a una. El codi generador es recull després de les mateixes, i està contingut, de nou, a l'arxiu *ActivityOneDef.py*.

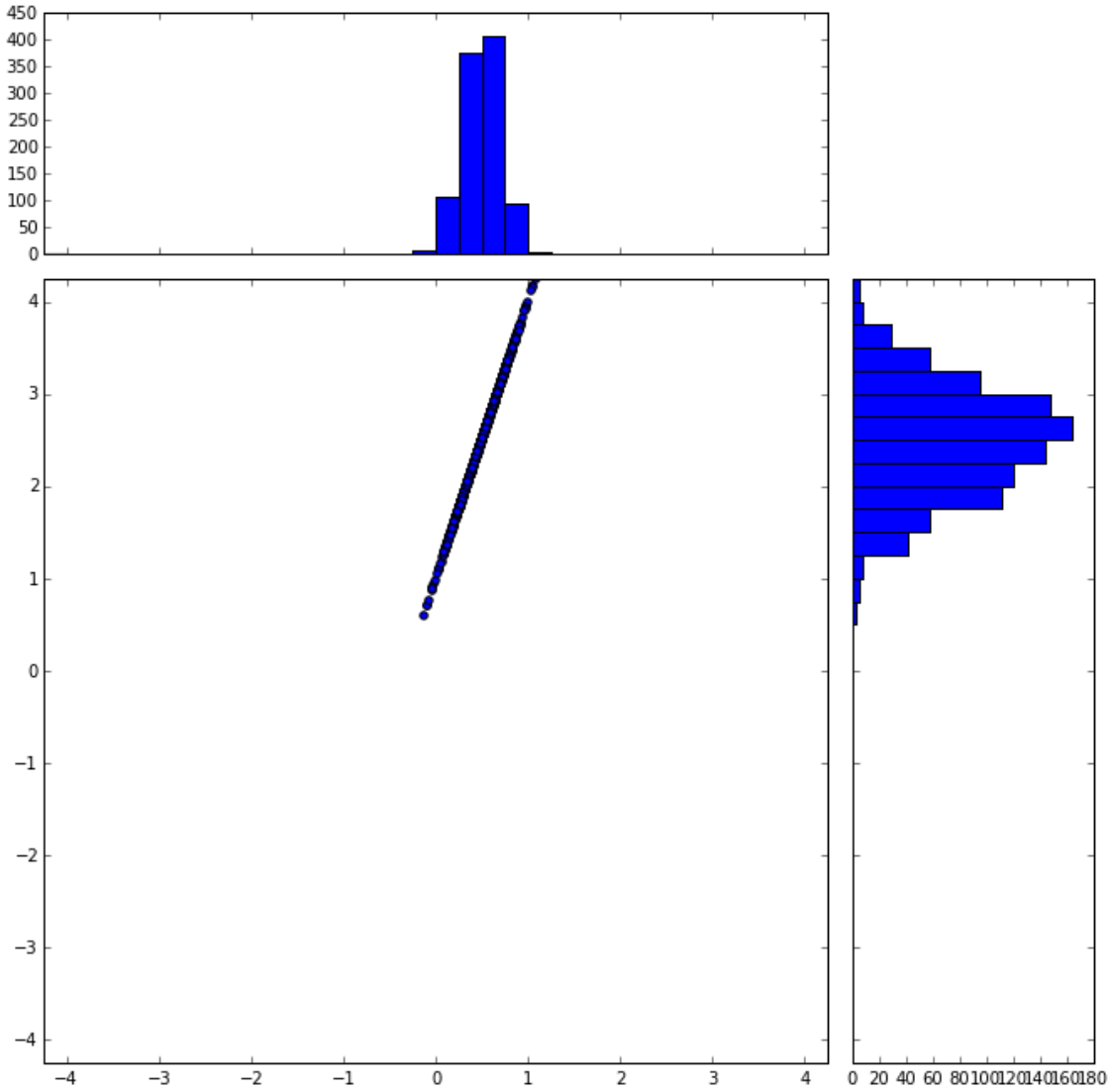
Combinació 1: X1 vs X2

Aquesta primera combinació enfronta les dues variables base. L'absència d'una agrupació al voltant d'una recta o corba indica, com era d'esperar, que les variables X1 i X2 no estan correlades (no existeix cap relació lineal o no de proporcionalitat). Addicionalment, es comprova que la variable X1 (eix d'ordenades) presenta una distribució centrada a 0.5 i que es dispersa fins a, aproximadament, ±3 desviacions tipus (-0.1 a 1.1, aprox.), segons les propietats de la distribució normal. D'altra banda, la variable X2 queda centrada en 4.5 i es dispersa fins a, aproximadament, ±3 desviacions tipus (2.7 a 6.3, aprox.). En aquest cas, al tenir la variable un major rang de valors, es pot observar la presència de valors atípics o *outliers*, que queden fora del rang indicat.



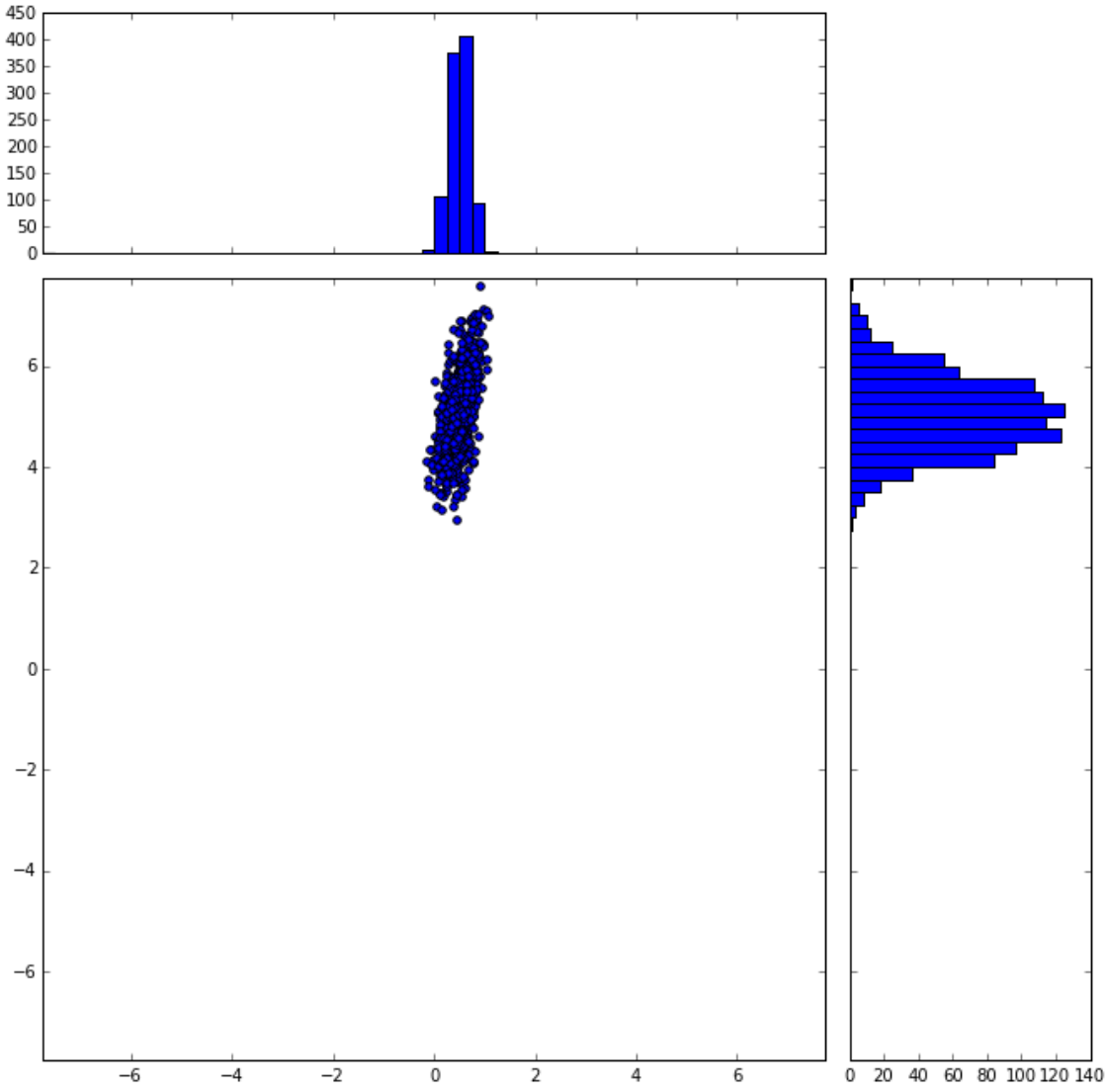
Combinació 2: X1 vs X3

Aquesta combinació recull les variables X1 i X3, relacionades linealment (X3 = 3 x X1 + 1). Com es pot comprovar, ambdues variables presenten una correlació lineal positiva molt forta, atès que els valors del diagrama de dispersió s'organitzen perfectament al voltant d'una recta. La coordenada basal de la recta (0,1) i el seu pendent (3) expressa els valors dels paràmetres de la combinació. Addicionalment, es pot comprovar que els valors de la segona variable queden centrats al voltant de 2,5 (3 x mu1 + 1).



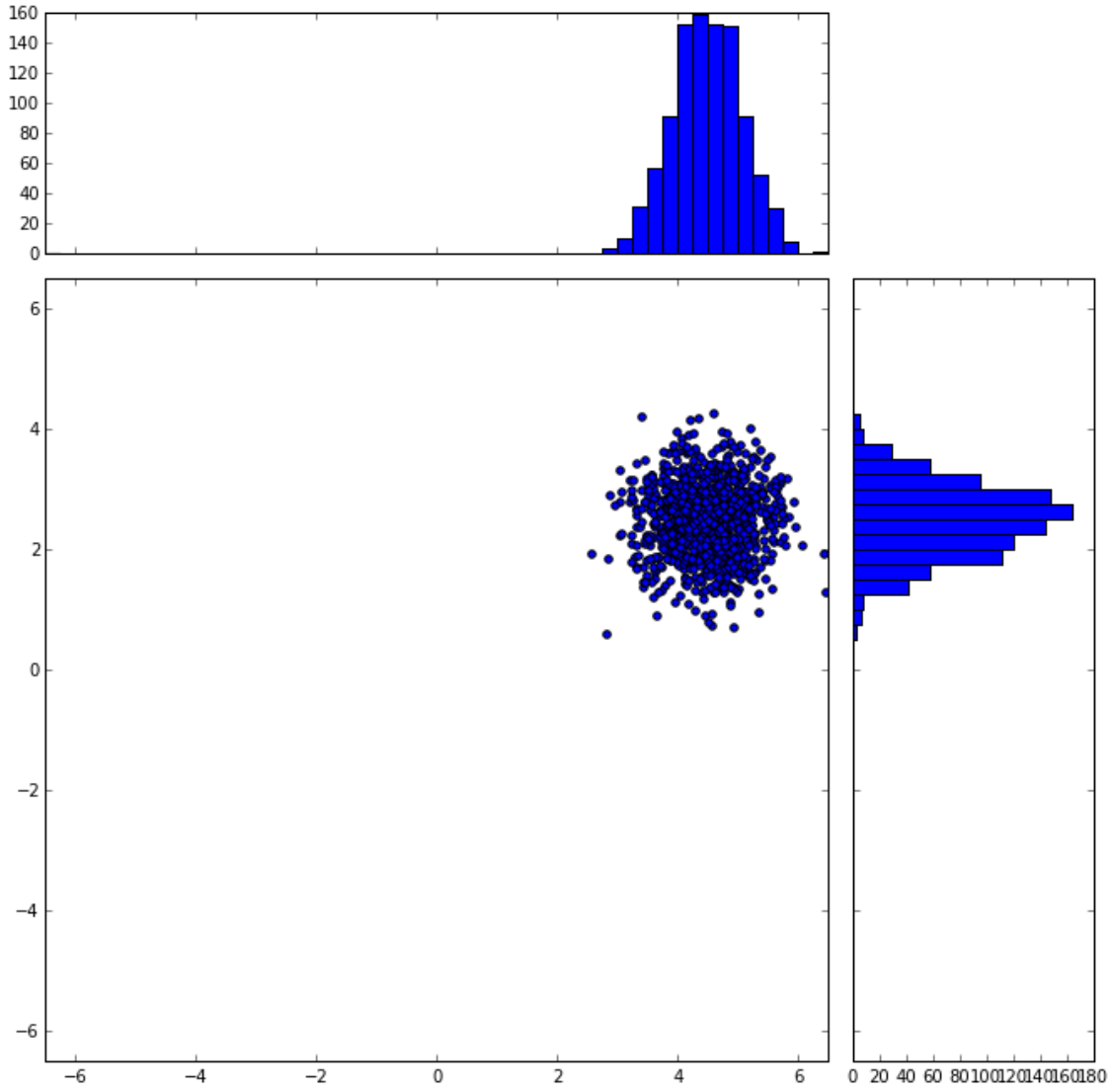
Combinació 3: X1 vs X4

Aquesta combinació enfronta les dues variables correlacionades: X1 i X4. L'agrupament al voltant d'una recta amb pendent positiva comença a indicar la correlació, si bé aquesta es més feble que la expressada en el punt anterior, com indica la dispersió dels punts, al quedar ambdues variables correlacionades a través d'una tercera variable aleatòria (X2).



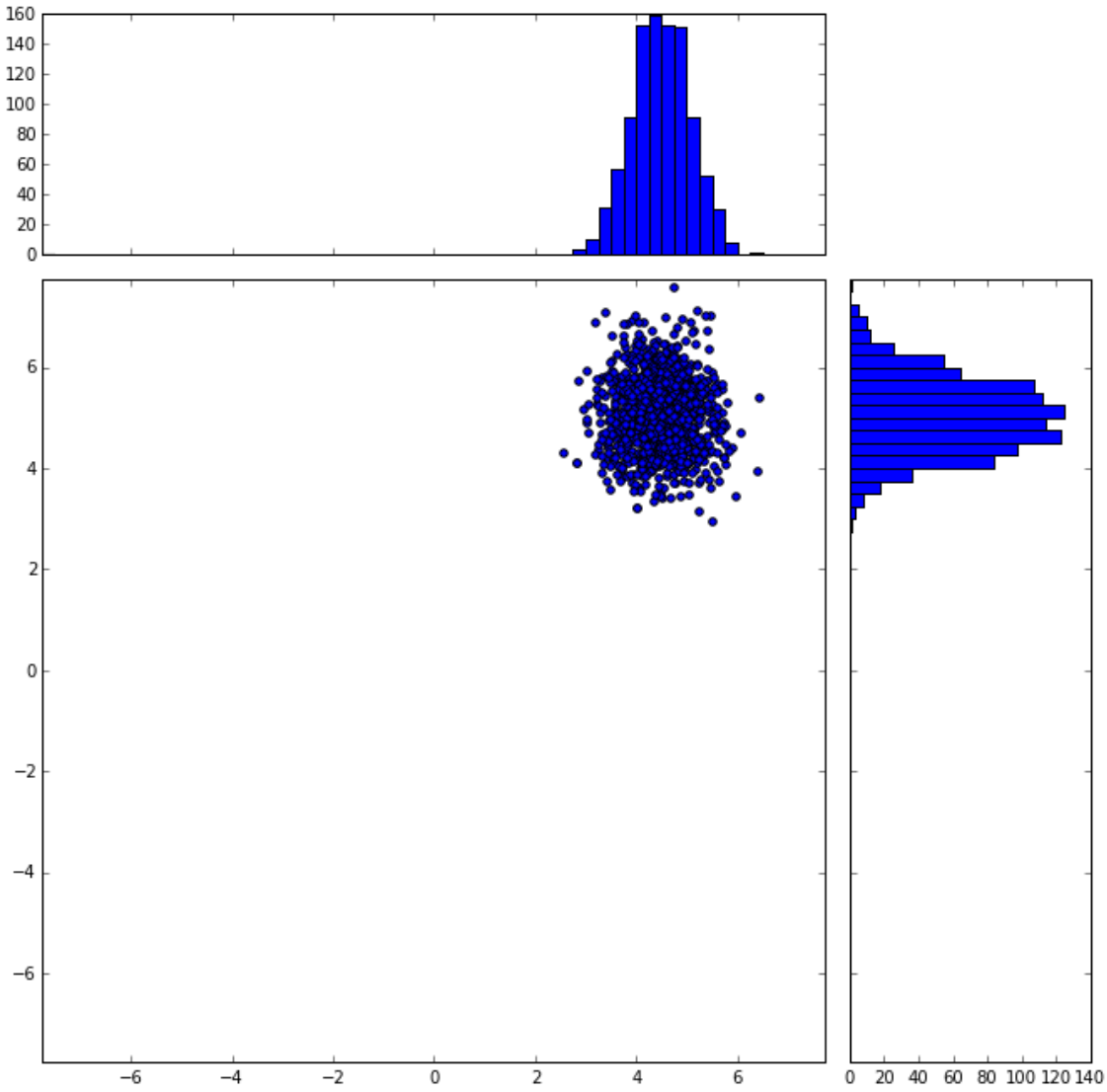
Combinació 4: X2 vs X3

Aquesta combinació torna a enfrontar dues variables no correlacionades, com indica l'absència d'agrupació al voltant d'una recta. S'observa que la variable X2 queda centrada a 4,5, mentre la variable X3 queda centrada a 2,5. L'amplitud del rang permet observar la presència de valors atípics, fora del rang que avarca l'espai comprés en el cercle centrat en (mu2, mu3) i de radi el mòdul de (sigma2, sigma3).



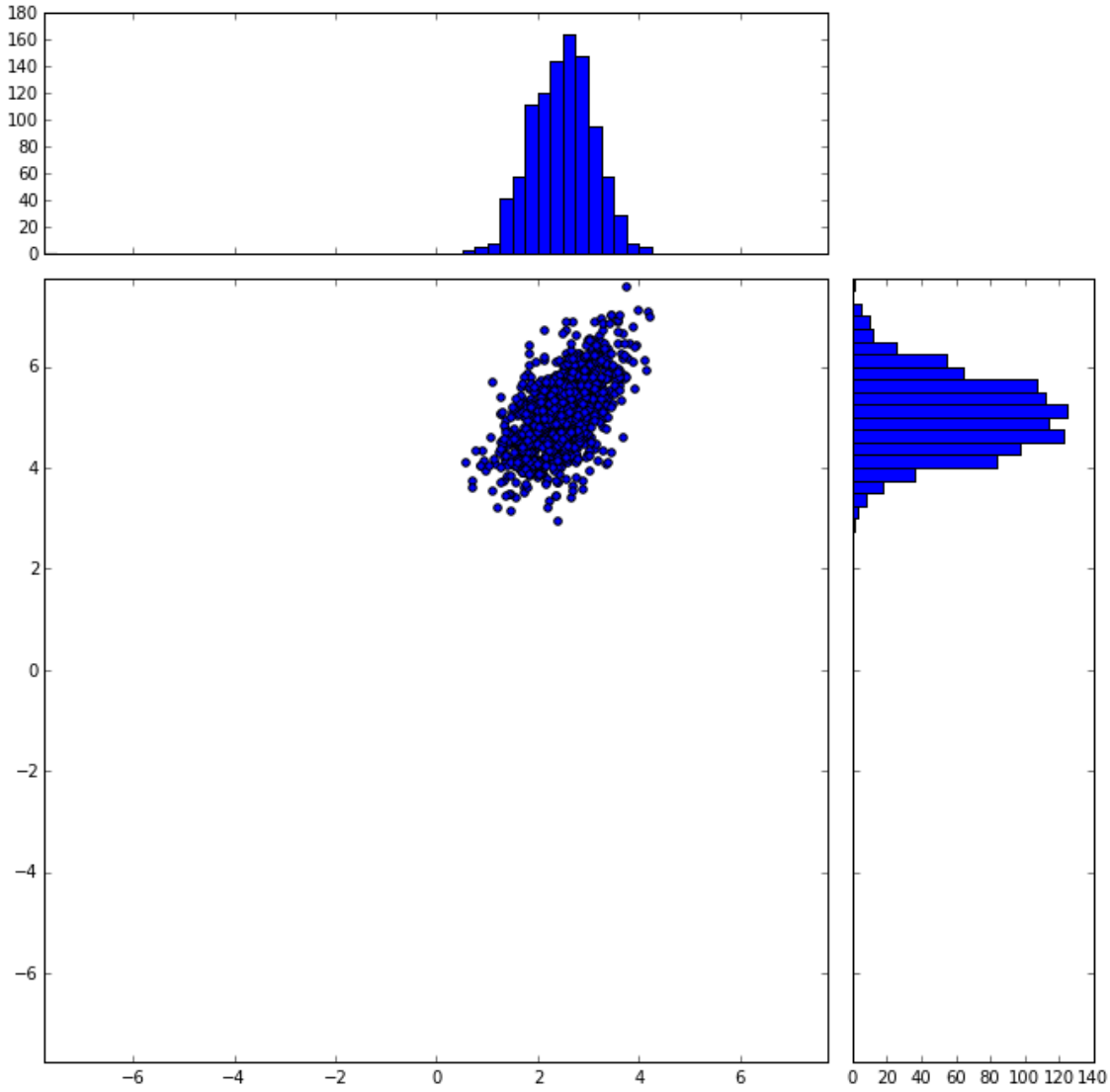
Combinació 5: X2 vs X4

Aquesta combinació enfronta les dues variables correlacionades: X2 i X4. En aquest cas, la correlació es tan feble que l'agrupament al voltant d'una corba es quasi imperceptible (si de cas s'intueix una certa forma ovalada en el diagrama de dispersió).



Combinació 6: X3 vs X4

Aquesta combinació és la més complexa de les mostrades, al enfrontar dues variables relacionades a través de la variable 1. L'equació de correlació és X4 = 2 x (X3/3 - 1)^2 + X2, i es llegeix a la figura en l'agrupació dels punts al voltant d'una corba. La correlació, de nou, es feble, com en el cas de X1 vs X4.



```
# En aquest punt, podem triar una combinació de variables per tal de
# realitzar la seva representació combinada
x = s1 # ORDENADES
y = s2 # ABCISSES

nullfmt = NullFormatter() # Aquest mètode eliminarà  les variables dels ti
cks

# Definicions pels eixos
left, width = 0.1, 0.65
bottom, height = 0.1, 0.65
bottom_h = left_h = left+width+0.02
rect_scatter = [left, bottom, width, height]
rect_histx = [left, bottom_h, width, 0.2]
rect_histy = [left_h, bottom, 0.2, height]

# Comencem amb la figura rectangular
plt.figure(1, figsize=(8,8))
axScatter = plt.axes(rect_scatter)
axHistx = plt.axes(rect_histx)
axHisty = plt.axes(rect_histy)
axHistx.xaxis.set_major_formatter(nullfmt)
axHisty.yaxis.set_major_formatter(nullfmt)

# el gràfic de dispersió:
axScatter.scatter(x, y)

# determinem els límits estètics
binwidth = 0.25
xymax = np.max( [np.max(np.fabs(x)), np.max(np.fabs(y))] )
lim = ( int(xymax/binwidth) + 1) * binwidth
axScatter.set_xlim( (-lim, lim) )
axScatter.set_ylim( (-lim, lim) )
bins = np.arange(-lim, lim + binwidth, binwidth)
axHistx.hist(x, bins=bins)
axHisty.hist(y, bins=bins, orientation='horizontal')
axHistx.set_xlim( axScatter.get_xlim() )
axHisty.set_ylim( axScatter.get_ylim() )

# i mostrem el gràfic
plt.show()
```

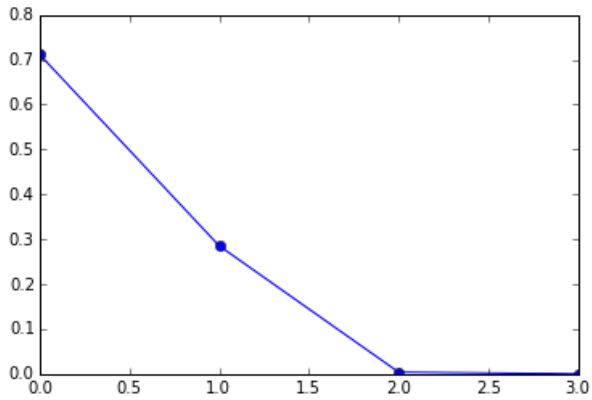
c. Apliqueu PCA a les dades anteriors i comenteu els resultats obtinguts tenint en compte les propietats de les dades. La funció PCA de scikit-learn està descrita a

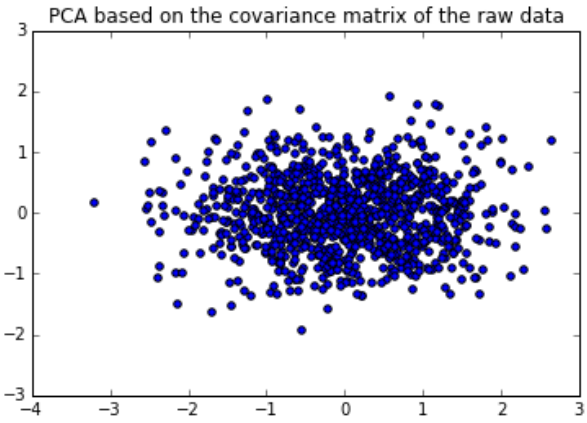
http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Quantes variables descorrelacionades hi han? Quants components principals són necessaris per explicar un 95% de la variabilitat de les dades?

L'anàlisi s'ha realitzat a partir d'un espai original de variables compost per dues variables independents o descorrelacionades i dues transformacions de les mateixes –lineal i quadràtica. És d'esperar, per tant, que la major part de la variància pugui ser explicada amb les dues components principals.

En aquest sentit, i segons es pot observar a l'screeplot que segueix aquestes línies, les dues primeres components del PCA expliquen, respectivament, el 71% i el 29% de la variància (és a dir, el 99% acumulat); les dues components restants gairebé no aporten informació addicional. Addicionalment, es mostra la representació de les dades a l'espai de les variables transformades (dues components principals).

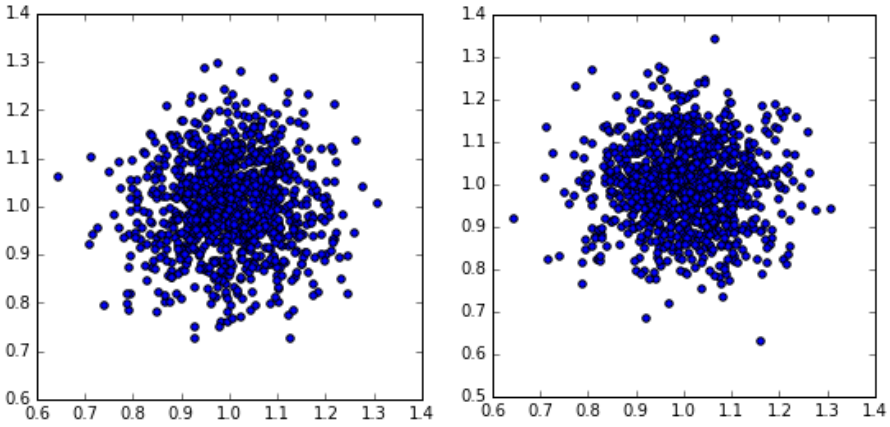




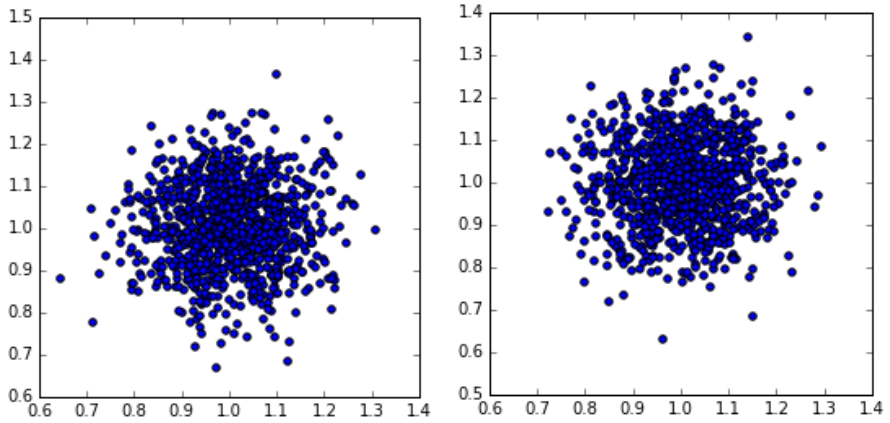

```
# Apliquem l'anàlisi de components principals, amb ncomponents = 2
    pca = PCA()
    pca.fit(s)
    # Imprimim la variància explicada per aquestes 2 components
    acumvar = [sum(pca.explained_variance_ratio_[:i+1])
                   for i in range(len(pca.explained_variance_ratio_))]
    print(list(zip(range(len(acumvar)), acumvar)))
    pylab.plot(pca.explained_variance_ratio_,'o-')
    pylab.show()

    print(pca.get_covariance())

    # Ara representarem la distribució en 2 components
    X_transf = pca.fit_transform(s)
    plt.scatter(X_transf[:,0], X_transf[:,1])
    plt.title('PCA based on the covariance matrix of the raw data')
    plt.show()
```

d. Compareu els resultats obtinguts als apartats anteriors amb els que s'obtindrien en cas que les 4 variables estiguessin distribuïdes amb distribucions independents X~N(1, 0.1). Justifiqueu raonadament els resultats obtinguts.

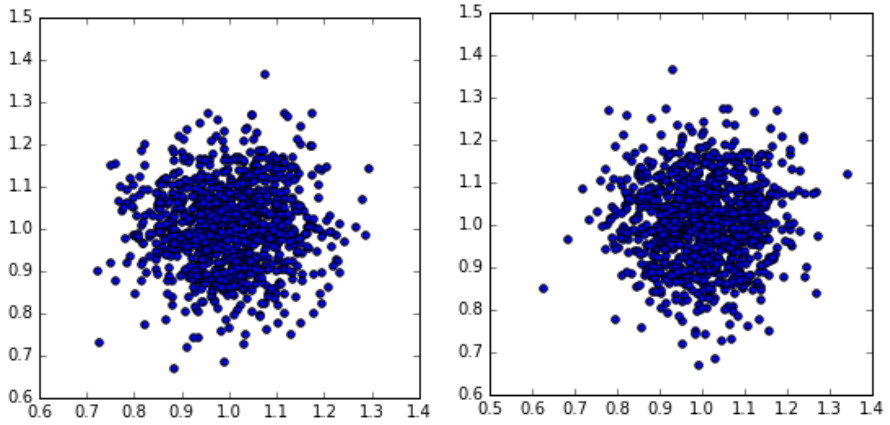
En aquest apartat s'ha construït una matriu de dimensió 4x1000 formada per 4 variables independents distribuïdes normalment tal que X~N(1, 0.1). Les següents figures mostren totes les combinacions de variables una a una. Com es pot observar, en tots els casos s'observa una absència de correlació entre les dues variables mostrades, el que indica que no hi ha dependències, lineals o no, entre les mateixes.



X1 vs X2 i X1 vs X3



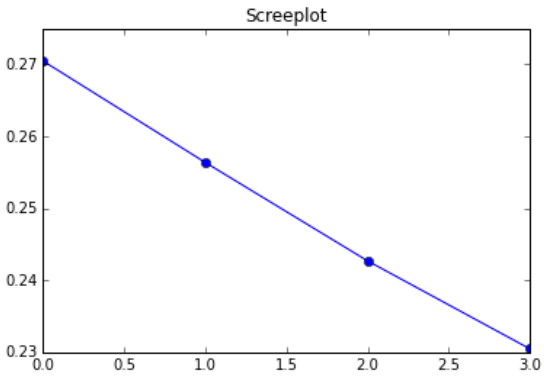
X1 vs X4 i X2 vs X3



X2 vs X4 i X3 vs X4

Addicionalment, es comprova que en aquest cas són necessàries totes les variables per a explicar el 95% de la variància, atès que cadascuna de les quatre components principals ve a explicar, aproximadament, una quarta part de la mateixa, tal i com s'observa en el Screeplot que es recull després de la taula.

Component	Variància explicada acumulada
1era component	27%
2a component	53%
3a component	77%
4a component	100%



Activitat 2: Classificació

a. Genereu un conjunt de dades bidimensional A1 amb N=2000 observacions amb una distribució normal amb mitjana [5, -4] i matriu de covariança [2, -1; -1, 2], i un altre conjunt A2 amb mitjana [1, -3] i matriu de covariança [1, 1.5; 1.5, 3]. Feu servir la següent funció de la llibreria numpy:

http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.multivariate_normal.html

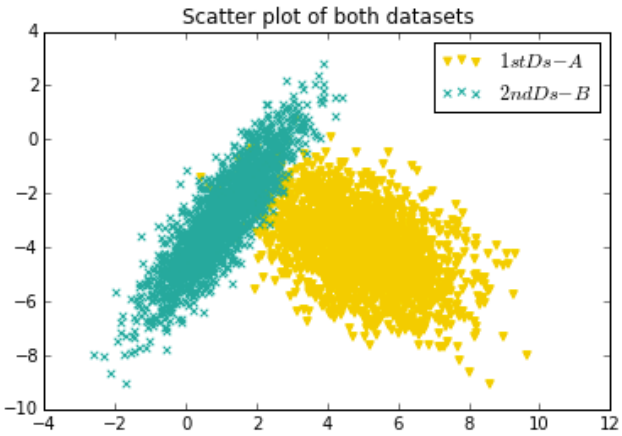
Tal i com indica l'enunciat s'ha generat un conjunt de dades sintètiques bidimensional amb mitjana [-5,4] i matriu de covariància [2, -1; -1, 2] i un altre amb mitjana [1, -3] i matriu de covariància [1, 1.5; 1.5, 3]. A aquest efecte, s'ha utilitzat el mètode *random.multivariate_normal* de la llibreria numpy tal i com recull el codi següent (arxiu de codi *ActivityTwoDef.py*)

```
# -*- coding: utf-8 -*-
"""
@author: Juan Águila Martínez (UOC - 2015)
"""

import numpy as np
import pylab
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.lda import LDA
from random import shuffle
import plotDecisionSurface

########################################################################
# Activitat 2
########################################################################

# Generem dos conjunts de dades bidimensionals, utilitzant
# el mètode random multivariate normal
mean = [5,-4] # Vector de mitjanes
cov = [[2,-1],[-1,2]] # Matriu de covariància
A = np.random.multivariate_normal(mean,cov,2000)

mean = [1,-3]
cov = [[1,1.5],[1.5,3]]
B = np.random.multivariate_normal(mean,cov,2000)
```

b. Representeu les dades de tots dos conjunts en forma de núvol de punts en un únic gràfic. Feu servir simbols de color diferent per representar les dades de cada conjunt.

La figura següent, generada amb la llibreria *pyplot*, mostra la representació gràfica de les dades generades en l'apartat anterior: representada amb triangles grocs la distribució A amb mitjana [-5,4] i matriu de covariància [2, -1; -1, 2]; amb creus blaves la distribució B, amb mitjana [1, -3] i matriu de covariància [1, 1.5; 1.5, 3]. Com es pot comprovar, ambdues distribucions queden creuades en valors de la primera component propers a 0. És d'esperar que aquesta propietat resulti en problemes a l'hora de realitzar una classificació precisa, per la dificultat de separar l'espai entre ambdues distribucions en aquesta regió.

Després de la figura es recull el codi generador (de nou, contingut en *ActivityTwoDef.py*).




```
# Representem els punts en un scatter plot
plt.scatter(A[:,0],A[:,1],color = '#F5CA0C', marker='v', label="$1st Ds -
 A$")
plt.scatter(B[:,0],B[:,1],color = '#00A99D', marker='x', label="$2nd Ds -
 B$")
plt.title("Scatter plot of both datasets")
plt.legend()
plt.show()
```

c. Construïu un conjunt de dades d'entrenament (training) amb les 1000 primeres observacions de cada conjunt de dades i un conjunt de prova (test) amb les altres 1000. Feu servir les següents eines de classificació disponibles a les llibreries scikit-learn per dissenyar un sistema de classificació automàtica dels conjunts A1 i A2 a partir de les dades de test.

Naïve Bayes: Funció GaussianNB del paquet sklearn.naive_bayes.

http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

Anàlisi de Discriminants Lineal: Funció LDA del paquet sklearn.lda.

http://scikit-learn.org/stable/modules/generated/sklearn.lda.LDA.html

En cada cas, determineu el percentatge d'encerts i errors de classificació sobre el conjunt de validació.

La taula següent recull els resultats (precisió i nombre d'errors) d'ambdós classificadors.

Classificador	Accuracy (%)	Errors
Naive Bayes	**96%**	**78/2000**
LDA	**94%**	**101/2000**

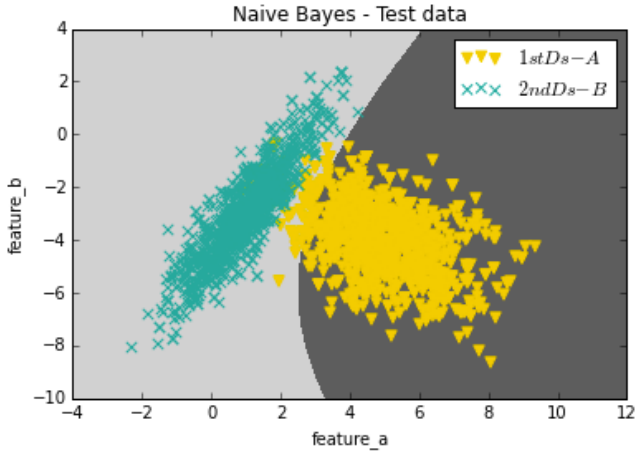
Es pot interpretar, a priori, que l'error es dona en la regió d'intersecció entre ambdues distribucions mencionada a l'apartat anterior. Per tal de comprovar-ho, però, hem pensat en aprofitar que el conjunt de dades en aquest cas és bidimensional per a representar les regions definides pels classificadors en el plànol i els punts de test amb l'etiquetat correcte sobre les mateixes.

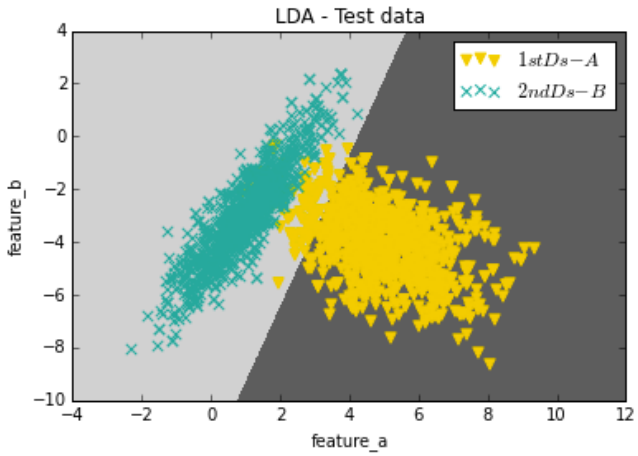
En aquest sentit, hem aprofitat la funció *plot_decision_surface* (recurs web) que, a partir del classificador entrenat, construeix una malla de punts per tal de realitzar la classificació de cada punt del plànol.

A partir de les figures es poden treure diverses conclusions:

- **Efectivament, és la zona d'intersecció la que dona lloc a dificultats en l'etiquetat. Com es pot veure a la figura, existeixen punts de la distribució A en la regió que el classificador entrenat etiqueta com a B.**
- **El classificador bayesià utilitza la probabilitat condicionada per a realitzar la classificació, pel que no pressuposa una geometria definida per l'hiperplànol que divideix l'espai – com sí fa el LDA.**
- **La propietat anterior es tradueix, en aquest cas particular, en un major encert en la classificació, atès que l'absència de curvatura força al classificador lineal a etiquetar de forma incorrecta alguns punts.**

Després de les figures es recull el codi generador.






```
# Afegirem una etiqueta a l'array, per a avaluar la classificació
tmp = np.zeros((2000, 1))
tmp[:] = 1
A = np.append(A,tmp,1)
tmp[:] = 2
B = np.append(B,tmp,1)

shuffle(A)
```

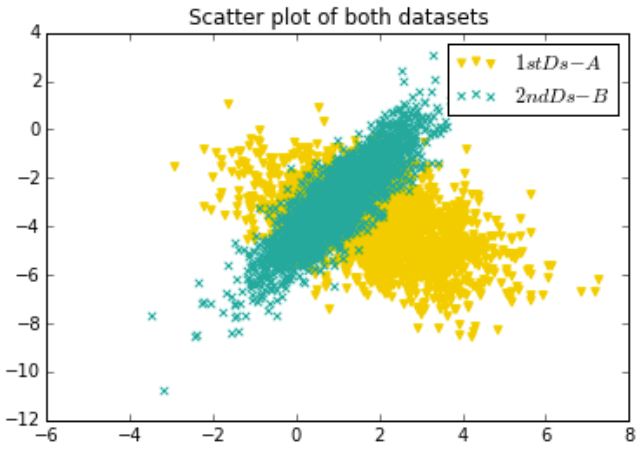
```
shuffle(B)

# I generem els vectors definitius que utilitzarem
X_train = np.concatenate((A[:1000,:2], B[:1000,:2]), axis=0 )
y_train = np.concatenate((A[:1000,2:], B[:1000,2:]), axis=0 )
# Reshape
y_train = np.ravel(y_train)

X_test  = np.concatenate((A[1000:,:2], B[1000:,:2]), axis=0 )
y_test  = np.concatenate((A[1000:,2:], B[1000:,2:]), axis=0 )
# Reshape
y_test = np.ravel(y_test)

# Entrenem el classificador
nb = GaussianNB()
y_pred = nb.fit(X_train, y_train).predict(X_test)

# Avaluem la classificació
acc = 100 * nb.score(X_test, y_test)
print("NB Single-validation")
print("Number of mislabeled points out of a total %d points : %d"
    % (X_test.shape[0],(y_test != y_pred).sum()))
print("Accuracy: %d pct"
    % (acc))
plot_decision_surface(nb, X_test, y_test, 'Naive Bayes - Test data')

# Entrenem el classificador
lda = LDA()
y_pred = lda.fit(X_train, y_train).predict(X_test)

# Avaluem la classificació
acc = 100 * lda.score(X_test, y_test)
print("LDA Single-validation")
print("Number of mislabeled points out of a total %d points : %d"
    % (X_test.shape[0],(y_test != y_pred).sum()))
print("Accuracy: %d pct"
    % (acc))
plot_decision_surface(lda, X_test, y_test, 'LDA - Test data')
```

d. Repetiu l'anàlisi anterior quan la mitjana del primer conjunt de dades A1 és [2, -4] en comptes de [5, -4]. Comenteu raonadament els resultats obtinguts i justifiqueu les diferències observades tenint en compte les característiques de les dades sintètiques utilitzades.

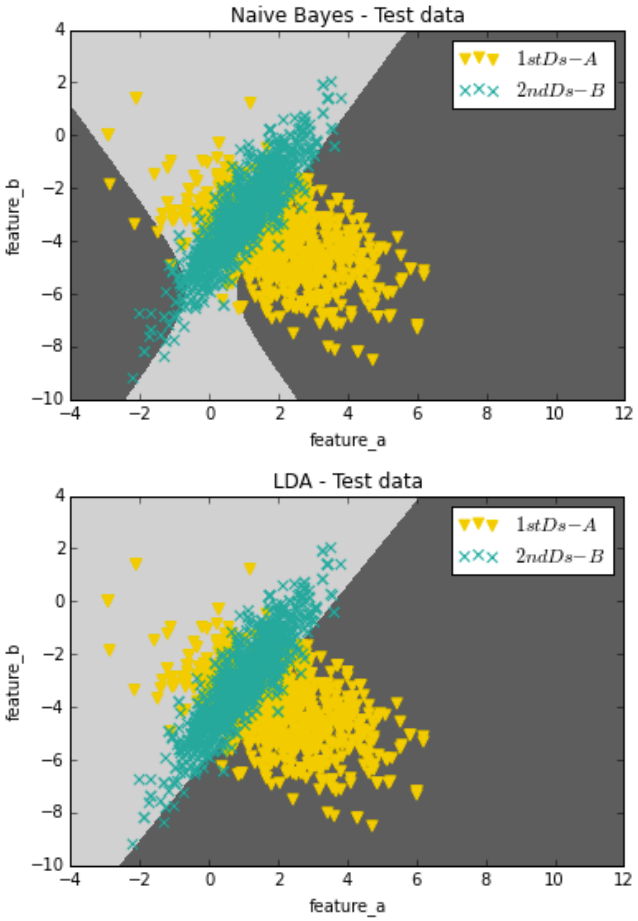
Es recullen a continuació els resultats obtinguts amb aquest canvi en la distribució de A.

Com es pot comprovar en la primera figura, ara existeix una major superfície d'intersecció, el que donarà lloc a una reducció en l'encert dels classificadors, com mostra la taula següent.

Classificador	Accuracy (%)	Errors
Naive Bayes	**78%**	**438/2000**
LDA	**80%**	**399/2000**



Hem tornat a realitzat l'etiquetat del plànol, observant ara la dificultat d'ambdós classificadors per a etiquetar els punts a la zona d'intersecció.



Com es pot comprovar, la intersecció de les distribucions s'interpreta com a ambigüitat en la classificació: no es pot inferir l'etiqueta a partir de les propietats en aquesta regió. Aquesta problemàtica redueix, de manera general, l'encert dels classificadors en el cas de conjunts de dades amb solapaments.