FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
UNIVERSIDAD DE CHILE

Big Data Analytics en Confiabilidad y Mantenimiento

Aprendizaje de Máquinas Aplicado a Confiabilidad y Mantenimiento

Viviana Meruane N.

Construcción de un modelo de PHM

Aprendizaje de máquinas

El **Aprendizaje de máquinas** (en ingles, machine learning) es la rama de la Inteligencia Artificial que tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender.

De forma más concreta, se trata de crear algoritmos capaces de generalizar comportamientos y reconocer patrones a partir de una información suministrada en forma de ejemplos.

Aprendizaje de máquinas

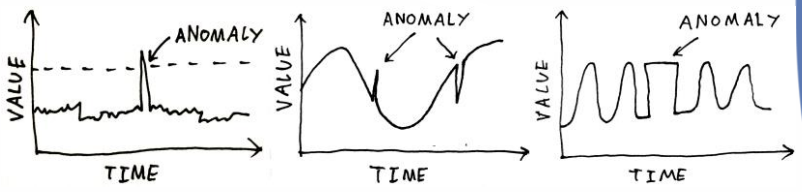
Los algoritmos de aprendizaje automático se clasifican según el tipo:

- Algoritmos para detección de anomalías y/o novedades
- Algoritmos de clasificación
- Algoritmos de regresión
- Algoritmos de agrupamiento

Detección de anomalías y/o novedades

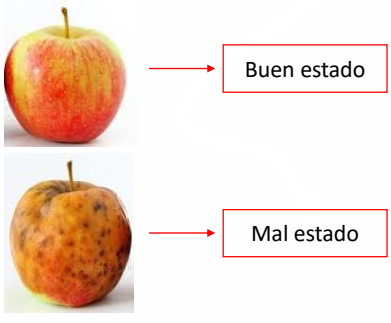
Un grupo de técnicas utilizadas para identificar comportamientos inusuales que no cumplen con el patrón esperado:

- Detección de anomalías: Datos están contaminados con anomalías.
- Detección de novedades: Solo se cuenta con datos normales para entrenar.



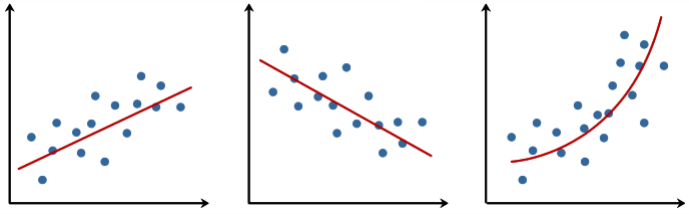
Algoritmos de clasificación

Un problema de clasificación busca encontrar un **modelo** capaz de **identificar** automáticamente para cada **objeto** la **clase** a la cual pertenecen.



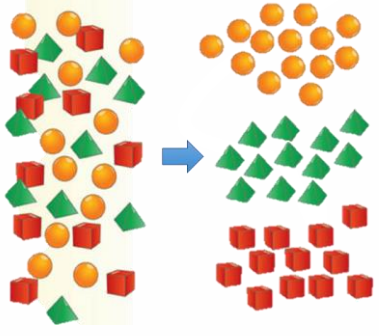
Algoritmos de regresión

Los algoritmos de regresión consisten en un conjunto de métodos que nos permiten **predecir** el valor de una o varias **variables continuas** basada en el valor de una o varias variables predictoras.



Algoritmos de agrupamiento

Un algoritmo de agrupamiento (en inglés, clustering) es un **procedimiento de agrupación** de una serie de datos de acuerdo con un **criterio** usualmente de **distancia o similitud**.



Método de aprendizaje

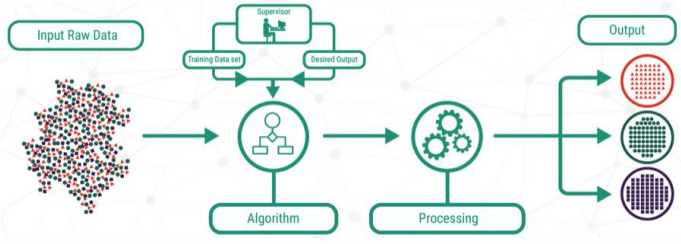
Los algoritmos de aprendizaje de máquinas también se pueden dividir según el método de aprendizaje:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje semi-supervisado
- Aprendizaje por refuerzo

Aprendizaje supervisado

En el aprendizaje supervisado, los algoritmos trabajan con datos "etiquetados" (labeled data), intentado encontrar una función que, dadas las variables de entrada (input data), les asigne la etiqueta de salida adecuada.

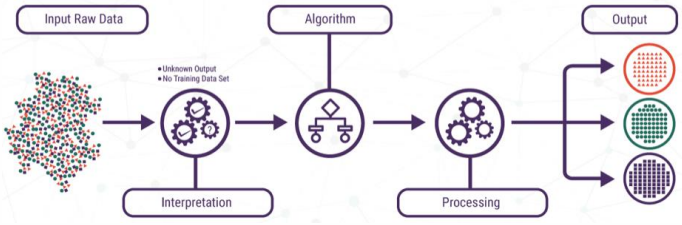
El algoritmo se entrena con un "histórico" de datos y así "aprende" a asignar la etiqueta de salida adecuada a un nuevo valor, es decir, predice el valor de salida (métodos de clasificación o regresión).



Aprendizaje no-supervisado

El aprendizaje no supervisado tiene lugar cuando no se dispone de datos "etiquetados" para el entrenamiento. Sólo conocemos los datos de entrada, pero no existen datos de salida que correspondan a un determinado input.

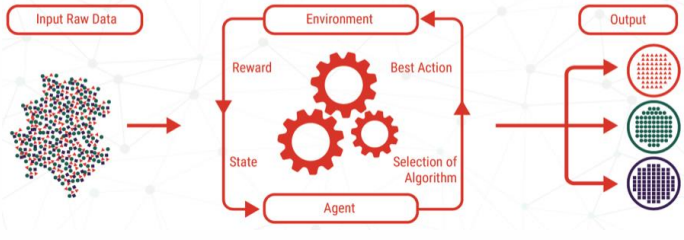
Por tanto, sólo podemos describir la estructura de los datos, para intentar encontrar algún tipo de organización que simplifique el análisis. Por ello, tienen un carácter exploratorio (métodos de reducción de dimensión, métodos de agrupamiento).



Aprendizaje por refuerzo

Este tipo aprendizaje se basa en mejorar la respuesta del modelo usando un proceso de retroalimentación. Se basan en los estudios sobre cómo fomentar el aprendizaje en humanos y ratas basándose en **recompensas** y **castigos**.

El algoritmo aprende observando el mundo que le rodea. Su información de entrada es la retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Por lo tanto, el sistema aprende a base de ensayo-error.



Prognostics and Health Monitoring (PHM)

Una estrategia de PHM está compuesta de tres etapas principales:

1. Detección: Trata de identificar el modo de operación del sistema y su estado.
2. Diagnóstico: Cuando se detecta una anomalía, el diagnóstico identifica el componente que tiene la anomalía y su severidad.
3. Pronóstico: El pronóstico trata de predecir el estado futuro del sistema y su vida útil remanente.

Prognostics and Health Monitoring (PHM)

El objetivo de este curso es conocer y aprender a utilizar las distintas herramientas de aprendizaje de máquinas que nos permiten realizar las primeras dos etapas: detección y diagnostico, a partir de monitoreo de variables físicas y/o operacionales.

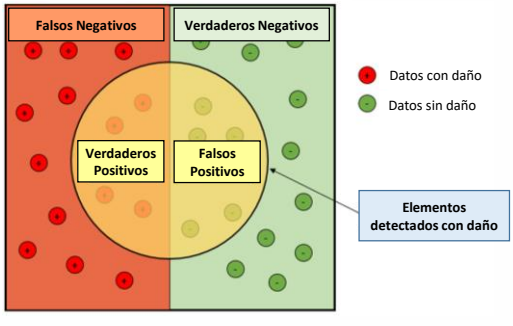
Detección de daño

Existen tres familias de algoritmos que se pueden utilizar para detectar el estado de salud de un sistema:

- No supervisado (detección de anomalías): Datos están contaminados con anomalías y no se saben las etiquetas.
- Semi-supervisado (detección de novedades): Solo se cuenta con datos normales para entrenar.
- Supervisado (clasificación): Se cuenta con las etiquetas de los datos normales y con fallas.

Métricas de evaluación

Supongamos que se tiene un conjunto de datos con daño y un conjunto de datos sin daño. Por medio de un algoritmo se determina cual grupo de datos posee daño.



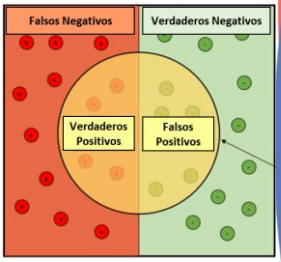
Métricas de evaluación

Se define la exactitud como el porcentaje de elementos clasificados correctamente y la precisión como el porcentaje de daño detectados que correspondían efectivamente a daño.

$$\text{Exactitud (accuracy)} = \frac{VP + VN}{VP + FN + FP + VN}$$

$$\text{Precisión (precision)} = \frac{VP}{VP + FP}$$

$$\text{Sensibilidad (recall)} = \frac{VP}{VP + FN}$$



Métricas de evaluación

- F1-score: media aritmética entre la precisión y la sensibilidad:

$$F_1 - \text{score} = \frac{2}{\frac{1}{\text{precisión}} + \frac{1}{\text{sensibilidad}}} = 2\frac{\text{precisión} \times \text{sensibilidad}}{\text{precisión} + \text{sensibilidad}}$$

Detección de anomalías

Detección de anomalías

Los datos de entrenamiento contienen valores anómalos (outliers) que se definen como observaciones que están lejos de los demás.

Los estimadores de detección de valores anómalos intentan ajustarse a las regiones donde los datos de entrenamiento son los más concentrados, ignorando las observaciones desviadas.

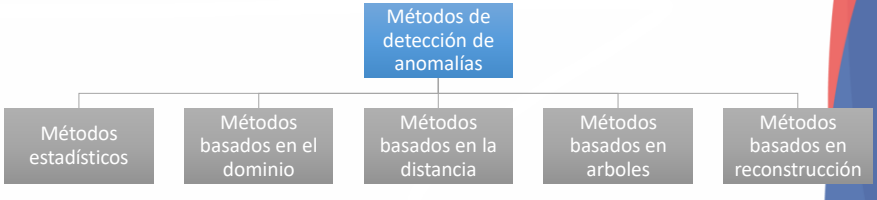
Detección de anomalías

La mayoría de los métodos de detección de anomalías crean un modelo para el comportamiento “normal” de los datos.

Luego, ven que tanto se ajusta un dato particular a este modelo y se le asigna “puntaje de anormalidad”.

Es por tanto, muy relevante la selección de un modelo adecuado.

Detección de anomalías

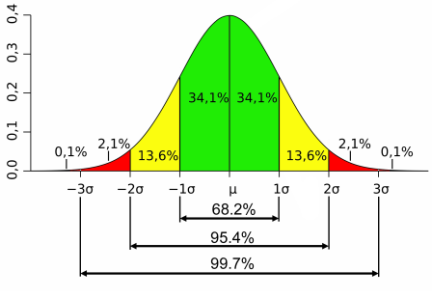


Métodos estadísticos

Métodos estadísticos

Una forma común de realizar una detección anomalías es asumir que los datos normales provienen de una distribución conocida (por ejemplo, una distribución Gaussiana).

Se definen las observaciones anómalas como observaciones que se encuentran lo suficientemente lejos de la forma ajustada.



Métodos estadísticos

La función densidad de probabilidad para una distribución normal con media μ y desviación estándar σ se define como:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Luego se puede calcular el z-score como:

$$z_i = \frac{x_i - \mu}{\sigma}$$

Métodos estadísticos

En el caso de más de un parámetro se puede ajustar a una función Gaussiana multivariable. El vector $\boldsymbol{\mu}$ define a la media de los datos y $\sum$ la matriz de covarianzas. La función densidad de probabilidad viene dada por:

$$f(\mathbf{x}) = \frac{1}{\sqrt{|\sum|}(2\pi)^{d/2}} e\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\sum^{-1}(\mathbf{x}-\boldsymbol{\mu})^T\right]$$

Donde d es la cantidad de parámetros y $|\sum|$ denota al determinante de la matriz $\sum$.

Métodos estadísticos

En este caso se utiliza como puntaje de anomalía a la distancia de Mahalanobis:

$$\text{Mahalanobis}(\mathbf{x}, \boldsymbol{\mu}, \sum) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})\sum^{-1}(\mathbf{x} - \boldsymbol{\mu})^T}$$

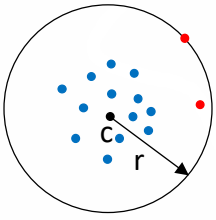
Métodos basados en el dominio

One Class Suport Vector Machine (OC-SVM)

El método de SVM para identificación de una clase trata de ajustar la menor híper-esfera (con radio r y centro c) a todos los datos disponibles.

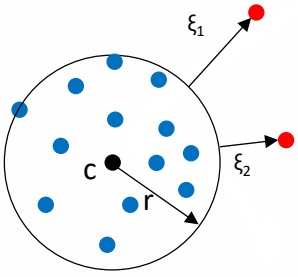
$$\min_{r,c} r^2$$

$$\text{sujeto a: } \|x_i - c\|^2 \leq r^2, \text{para } i = 1,2,\dots,n$$



One Class Suport Vector Machine (OC-SVM)

Una versión más flexible de la formulación permite la presencia de valores anómalos.



One Class Suport Vector Machine (OC-SVM)

Lo que se formula como:

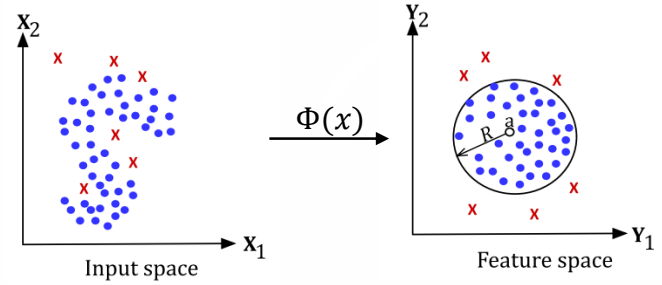
$$\min_{r,c} r^2 + \frac{1}{nv}\sum_{i=1}^{2} \xi_i$$

$$\text{sujeto a: } \|x_i - c\|^2 \leq r^2 + \xi_i, \text{para } i = 1,2,\dots,n$$

nv es un parámetro positivo que especifica la relación entre el volumen de la esfera y el número de valores anómalos.

One Class Suport Vector Machine (OC-SVM)

En muchos casos, la frontera de decisión no tiene la forma de una híper-esfera. En esos casos se puede hacer una transformación no lineal, de forma que los datos proyectados en el nuevo espacio si tengan una frontera similar a una hiper-esfera.



One Class Suport Vector Machine (OC-SVM)

En la práctica el problema se reformula de manera que sea solo necesario evaluar la matriz de kernel:

$$\mathrm{K_{ij}} = \kappa(\mathbf{z}_i, \mathbf{z}_j) = \phi(\mathbf{z}_i)^{\mathrm{T}}\phi(\mathbf{z}_j).$$

Esto es más eficiente!

One Class Suport Vector Machine (OC-SVM)

Algunas funciones típicas son:

- Polinomial:

$$K(x, y) = (1 + x^T y)^d$$

- Radial Basis Function (RBF):

$$K(x, y) = \exp\left[-(x-y)^T(x-y)/2\sigma^2\right]$$

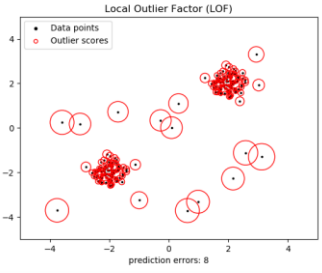
- Sigmoid:

$$K(x, y) = \tanh(\rho_1 x^T y + \rho_2)$$

Métodos basados en distancia

Local Outlier Factor (LOF)

El método Local Outlier Factor (LOF) calcula una puntuación (denominada factor de valores anómalos locales) que refleja el grado de anomalía de las observaciones. Mide la densidad local de un dato dado con respecto a sus vecinos. La idea es detectar las muestras que tienen una densidad sustancialmente menor que sus vecinos.



Local Outlier Factor (LOF)

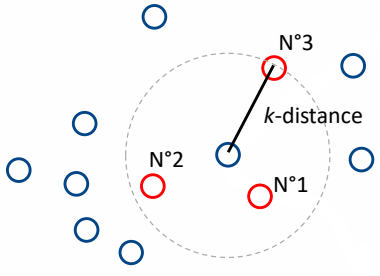
La puntuación LOF de una observación es igual a la razón de la densidad local promedio de sus "k" vecinos más cercanos, y su propia densidad local: **se espera que una instancia normal tenga una densidad local similar a la de sus vecinos, mientras que para los datos anómalos se espera que tengan una densidad local mucho menor**.

El número de vecinos considerados suele ser:

1. Mayor que el número mínimo de muestras que debe contener un grupo.
2. Menor que el número máximo de muestras cercanas que podrían ser datos anómalos.

Local Outlier Factor (LOF)

k-distance: Distancia de un punto a su *k*-esimo vecino. Por ejemplo, si *k*=3 corresponde a la distancia a su tercer vecino.



Local Outlier Factor (LOF)

Distancia de alcance (reach-dist): máximo de la distancia de dos puntos y la *k*-distance del segundo punto.

$$\text{reach}-\text{dist}_k(a,b) = \max[\text{k}-\text{distance}(b), \text{dist}(a,b)]$$

Básicamente, si el punto *a* está dentro de los *k* vecinos del punto *b*, la $\text{reach}-\text{dist}_k(a,b)$ será la *k*-distance de *b*. De lo contrario, será la distancia real entre *a* y *b*.

Local Outlier Factor (LOF)

Se puede definir la distancia de alcance promedio para el punto *a* como:

$$\text{avg}-\text{dist}_k(a) = \frac{1}{k}\sum_{i=1}^{k} \text{reach}-\text{dist}_k(a,i)$$

Por último, la densidad de alcance local para el punto *a* como:

$$lrd_k(a) = \frac{1}{\text{avg}-\text{dist}_k(a)}$$

Local Outlier Factor (LOF)

El local Outlier Factor (LOF) mide la razón entre la densidad de alcance local promedio de los *k* vecinos y la del punto en estudio.

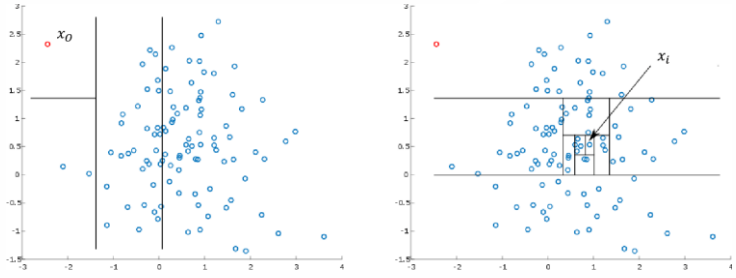
Si la densidad de un punto es mucho menor que las densidades de sus vecinos (LOF $\gg$1), el punto está lejos de las áreas densas y, por lo tanto, es un valor atípico.

Métodos basados en árboles

Isolation Forest

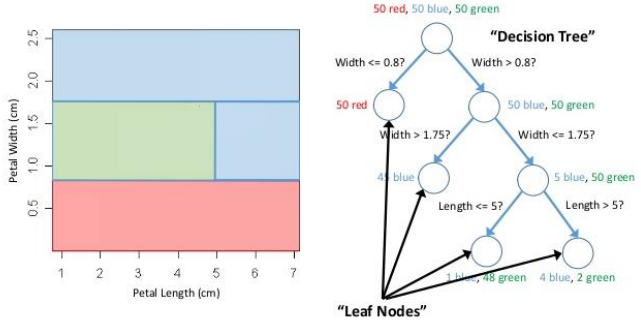
El método “isolation forest” trata de aislar los valores anómalos al hacer sucesivas divisiones de los datos.

Se selecciona un parámetro al azar y luego también al azar se asigna un valor de ese parámetro para dividir los datos en dos grupos.



Isolation Forest

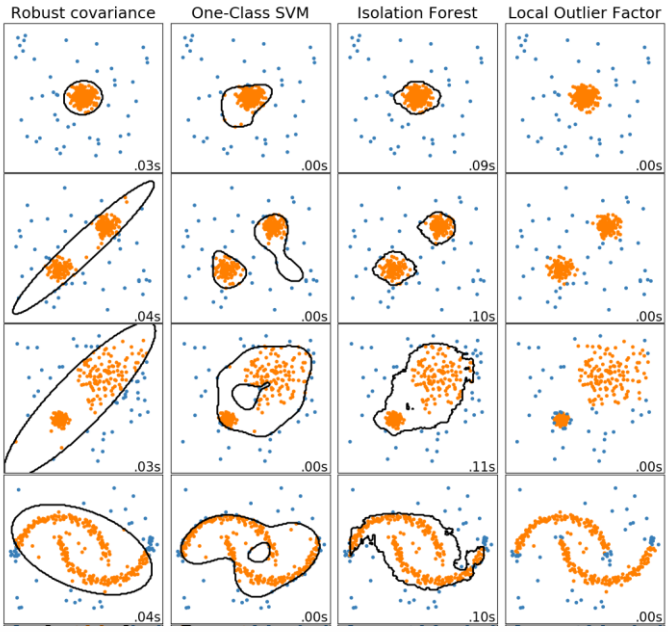
Si representamos este proceso como un árbol, el número de divisiones necesarias para aislar una muestra es equivalente a la longitud de la ruta desde el nodo raíz hasta el nodo de terminación.



Isolation Forest

La longitud de esta ruta, promediada sobre un bosque de árboles aleatorios, es una medida de normalidad.

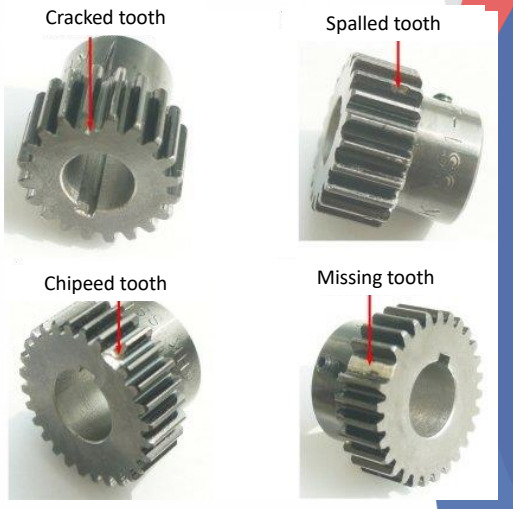
La partición aleatoria produce caminos notablemente más cortos para las anomalías. Por lo tanto, cuando un bosque de árboles aleatorios produce colectivamente longitudes de camino más cortas para muestras particulares, es muy probable que sean anomalías.



Ejemplo 1

Se cuenta con datos de vibración obtenidos en un montaje experimental de una caja de engranajes operando en estado saludable y con distintas fallas en los engranajes.

Provenientes de la siguiente base de datos: https://figshare.com/articles/Gear_Fault_Data/6127874/1



Ejemplo 1

Se usaron los datos temporales. Todos las secuencias sin falla (104) y 10 secuencias con falla. Lo que se guardó en un archivo de 3600x114 (DatosA.mat): son 114 secuencias temporales y cada secuencia tiene 3600 datos adquiridos a una frecuencia de muestreo de 1000 Hz.

Se entrega también un archivo con las etiquetas de cada secuencia (0= sin daño, 1=con daño) (labelA.mat).

Vamos a implementar una estrategia de detección de anomalías.

Ejemplo 1

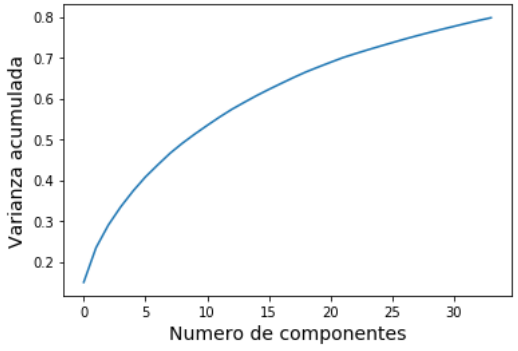
Primero vamos a hacer una extracción de características. Para esto calculamos la transformada de Fourier de cada segmento y luego hacemos una reducción con PCA para mantener un 80% de la varianza.

```
#importar librerías
import scipy.io as sio
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from sklearn.decomposition import PCA
from scipy.fftpack import fft


#Leer datos
Datos=sio.loadmat('DatosA.mat') #Healthy
Datos=Datos['DatosA']
```


Ejemplo 1

```
#vector de tiempo
Fs=1000#sampling rate
dt=1/Fs #paso de tiempo
Ns=114 #numero de segmentos
L=3600 #datos por segmento

P=np.zeros((Ns,int(L/2)))


for i in range(0,Ns):
    x=Datos[:,i]
    F = fft(x)[0:int(L/2)]/(L/2)
    P[i,:]=abs(F).transpose()


# PCA
pca = PCA(n_components=34)
pca.fit(P)
varianza=pca.explained_variance_ratio_.cumsum()

plt.plot(varianza)
plt.xlabel('Numero de componentes', fontsize=14)
plt.ylabel('Varianza acumulada', fontsize=14)
```

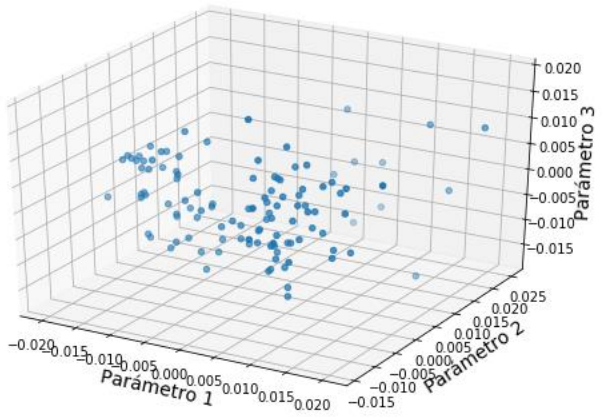
Ejemplo 1



Ejemplo 1

```
Xt=pca.transform(P)

fig = plt.figure(figsize=(9,6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(Xt[:, 0], Xt[:, 1],Xt[:, 2])
ax.set_xlabel('Parámetro 1', fontsize=14)
ax.set_ylabel('Parámetro 2', fontsize=14)
ax.set_zlabel('Parámetro 3', fontsize=14)
plt.show()

np.save('Xt',Xt)
```

Ejemplo 1



Ejemplo 1

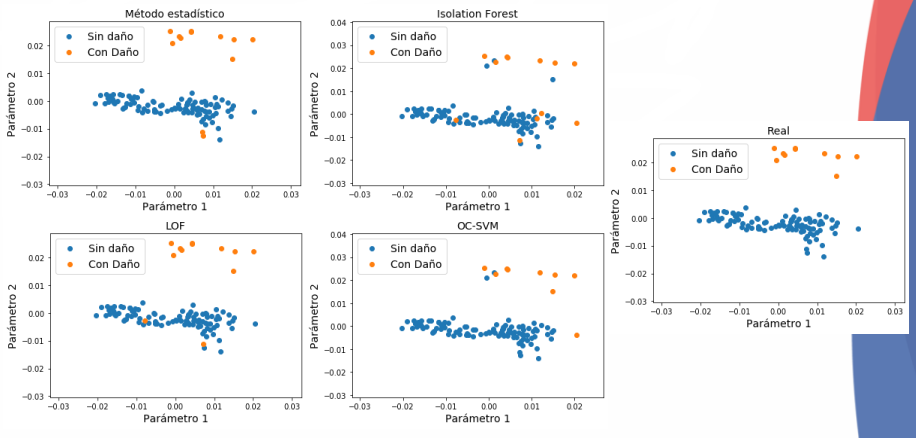
Luego implementamos los algoritmos de detección de anomalías:

```
#importar librerias
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt
from sklearn.covariance import EllipticEnvelope
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn import svm
from sklearn.metrics import precision_score, accuracy_score, recall_score, f1_score, confusion_matrix

#Leer datos
Xt=np.load('Xt.npy')
Datos=Xt

Label=sio.loadmat('labelA.mat') #Healthy
Yt=Label['labelA']


outlier_frac=0.1 #fracción de valores anomalos, valor por defecto es 0.1
Model= EllipticEnvelope(contamination=outlier_frac)
# Model=IsolationForest(n_estimators=100, max_samples='auto', contamination=outlier_frac)
# Model = LocalOutlierFactor(n_neighbors=20, contamination=outlier_frac)
# Model = svm.OneClassSVM(nu=0.4, kernel="rbf", gamma='auto')
Yp=Model.fit_predict(Datos)
```

Ejemplo 1

```
Yp=np.where(Yp==1, 0, Yp)
Yp=np.where(Yp==-1, 1, Yp)

Xp0=Datos[np.where(Yp == 0)[0],:]
Xp1=Datos[np.where(Yp == 1)[0],:]

Xt0=Datos[np.where(Yt == 0)[0],:]
Xt1=Datos[np.where(Yt == 1)[0],:]


plt.scatter(Xp0[:, 0], Xp0[:, 1])
plt.scatter(Xp1[:, 0], Xp1[:, 1])
plt.xlabel('Parámetro 1', fontsize=14)
plt.ylabel('Parámetro 2', fontsize=14)
plt.legend(('Sin daño','Con Daño'), fontsize=14)
plt.title('Detectado', fontsize=14)
plt.show()

plt.scatter(Xt0[:, 0], Xt0[:, 1])
plt.scatter(Xt1[:, 0], Xt1[:, 1])
plt.xlabel('Parámetro 1', fontsize=14)
plt.ylabel('Parámetro 2', fontsize=14)
plt.legend(('Sin daño','Con Daño'), fontsize=14)
plt.title('Real', fontsize=14)
plt.show()
```


Ejemplo 1



Ejemplo 1

```
Exactitud=accuracy_score(Yt, Yp)
Presicion=precision_score(Yt, Yp)
Sensibilidad=recall_score(Yt, Yp)
F1score=f1_score(Yt, Yp)

print("Exactitud= ", '{:.2f}'.format(Exactitud))
print("Presicion= ", '{:.2f}'.format(Presicion))
print("Sensibilidad= ", '{:.2f}'.format(Sensibilidad))
print("F1 score= ", '{:.2f}'.format(F1score))
```


Ejemplo 1

Método	Exactitud	Precisión	Sensibilidad	F1-score
Estadístico	0.98	0.83	1.00	0.91
OC-SVM	0.97	0.89	0.80	0.84
Isolation Forest	0.91	0.50	0.60	0.55
LOF	0.98	0.83	1.00	0.91

Detección de novedades

Los datos de entrenamiento no están contaminados con datos anómalos y estamos interesados en detectar si una nueva observación es un valor anómalo. En este contexto, un valor anómalo también se denomina novedad.

Los métodos vistos anteriormente también se pueden utilizar como detectores de novedades.

Ejemplo 2

Se entregan los datos del Ejemplo 1, pero con datos solo sin daño. Hacemos el mismo procedimiento de extracción de caracteristicas:

```
#importar librerías
import scipy.io as sio
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from sklearn.decomposition import PCA
from scipy.fftpack import fft


#Leer datos
Datos=sio.loadmat('DatosN.mat') #Healthy
Datos=Datos['DatosN']

#vector de tiempo
Fs=1000#sampling rate
dt=1/Fs #paso de tiempo
Ns=104 #numero de segmentos
L=3600 #datos por segmento

P=np.zeros((Ns,int(L/2)))
```

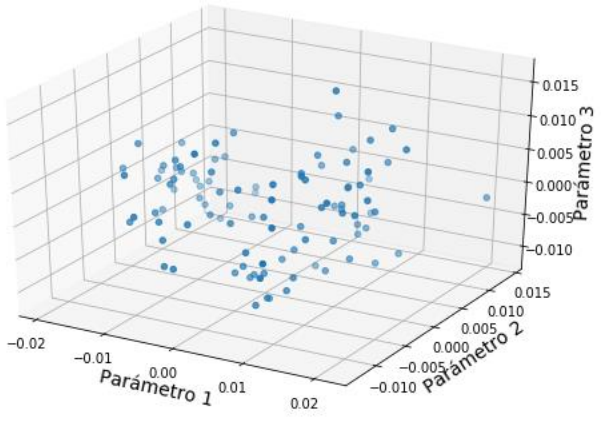
Ejemplo 2

```
for i in range(0,Ns):
    x=Datos[:,i]
    F = fft(x)[0:int(L/2)]/(L/2)
    P[i,:]=abs(F).transpose()


# PCA
pca = PCA(n_components=34)
pca.fit(P)
varianza=pca.explained_variance_ratio_.cumsum()

plt.plot(varianza)
plt.xlabel('Numero de componentes', fontsize=14)
plt.ylabel('Varianza acumulada', fontsize=14)

Xn=pca.transform(P)
```


Ejemplo 2

```
fig = plt.figure(figsize=(9,6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(Xn[:, 0], Xn[:, 1],Xn[:, 2])
ax.set_xlabel('Parámetro 1', fontsize=14)
ax.set_ylabel('Parámetro 2', fontsize=14)
ax.set_zlabel('Parámetro 3', fontsize=14)
plt.show()

np.save('Xn',Xn)
```

Ejemplo 2



Ejemplo 2

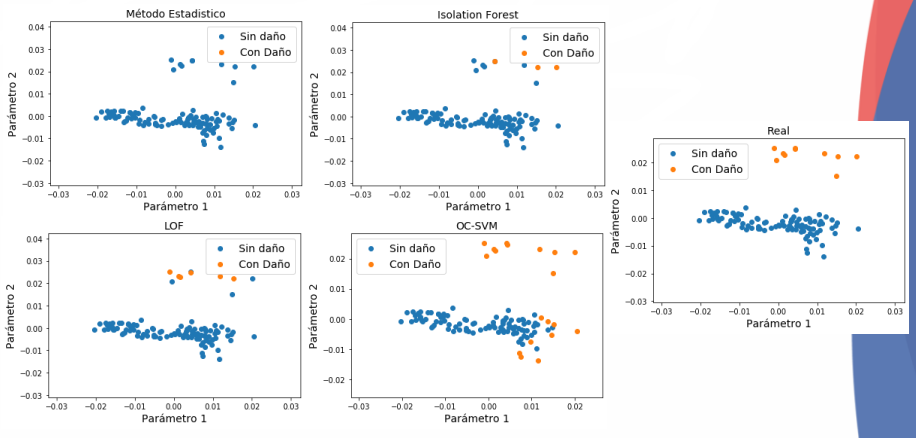
Luego implementamos los algoritmos de detección de novedades:

```
#importar librerias
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt
from sklearn.covariance import EllipticEnvelope
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn import svm
from sklearn.metrics import precision_score, accuracy_score, recall_score, f1_score, confusion_matrix

#Leer datos
Xt=np.load('Xt.npy')
Xn=np.load('Xn.npy')

Datos_train=Xn
Datos_test=Xt

Label=sio.loadmat('labelA.mat') #Healthy
Yt=Label['labelA']


outlier_frac=0.001 #fracción de valores anomalos, valor por defecto es 0.1
Model= EllipticEnvelope(contamination=outlier_frac)
# Model=IsolationForest(n_estimators=100, max_samples='auto', contamination=outlier_frac)
# Model = LocalOutlierFactor(n_neighbors=20, contamination=outlier_frac,novelty=True)
# Model = svm.OneClassSVM(nu=0.1, kernel="rbf", gamma='auto')
Model.fit(Datos_train)
Yp=Model.predict(Datos_test)
```

Ejemplo 2

```
Yp=np.where(Yp==1, 0, Yp)
Yp=np.where(Yp==-1, 1, Yp)

Xp0=Datos_test[np.where(Yp == 0)[0],:]
Xp1=Datos_test[np.where(Yp == 1)[0],:]

Xt0=Datos_test[np.where(Yt == 0)[0],:]
Xt1=Datos_test[np.where(Yt == 1)[0],:]


plt.scatter(Xp0[:, 0], Xp0[:, 1])
plt.scatter(Xp1[:, 0], Xp1[:, 1])
plt.xlabel('Parámetro 1', fontsize=14)
plt.ylabel('Parámetro 2', fontsize=14)
plt.legend(('Sin daño','Con Daño'), fontsize=14)
plt.title('Método Estadistico', fontsize=14)
plt.show()

plt.scatter(Xt0[:, 0], Xt0[:, 1])
plt.scatter(Xt1[:, 0], Xt1[:, 1])
plt.xlabel('Parámetro 1', fontsize=14)
plt.ylabel('Parámetro 2', fontsize=14)
plt.legend(('Sin daño','Con Daño'), fontsize=14)
plt.title('Real', fontsize=14)
plt.show()
```


Ejemplo 2



Ejemplo 2

```
Exactitud=accuracy_score(Yt, Yp)
Presicion=precision_score(Yt, Yp)
Sensibilidad=recall_score(Yt, Yp)
F1score=f1_score(Yt, Yp)

print("Exactitud= ", '{:.2f}'.format(Exactitud))
print("Presicion= ", '{:.2f}'.format(Presicion))
print("Sensibilidad= ", '{:.2f}'.format(Sensibilidad))
print("F1 score= ", '{:.2f}'.format(F1score))
```


Ejemplo 2

Método	Exactitud	Precisión	Sensibilidad	F1-score
Estadístico	0.91	0.00	0.00	0.00
OC-SVM	0.92	0.53	1.00	0.69
Isolation Forest	0.94	1.00	0.30	0.46
LOF	0.96	1.00	0.60	0.75

Clasificación

Identificación de daño

Se busca determinar el tipo de daño y su severidad. Esto se puede realizar mediante algoritmos de clasificación.

La clasificación puede ser:

- No hay daño o hay daño
- No hay daño, hay daño tipo 1, hay daño tipo 2,…
- No hay daño, hay daño tamaño 1, hay daño tamaño 2,…

Métricas de Evaluación Multi-clase

Las métricas típicas que se utilizan en la multi-clase son las mismas que se utilizan en el caso de la clasificación binaria. La métrica se calcula para cada clase al procesarla como un problema de clasificación binaria después de agrupar todas las otras clases como pertenecientes a la segunda clase.

Ejemplo

Real: A A A A C C C B BB

Predicho: A B A B C C C A BB

		Predicho A	Predicho B	Predicho C	
Real	A	2	2	0	4
	B	1	2	0	3
	C	0	0	3	3
		3	4	3	Total

Ejemplo

Clase	Precisión	Sensibilidad	F1-score
A	0.67	0.50	0.57
B	0.50	0.67	0.57
C	1.00	1.00	1.00

	A	B	C	
A	2	2	0	$\frac{2}{4}$ Sensibilidad
B	1	2	0	
C	0	0	3	
	$\frac{2}{3}$			
	Precisión			

$$\text{Exactidud} = \frac{7}{10} = 0.7$$

Algoritmos de clasificación

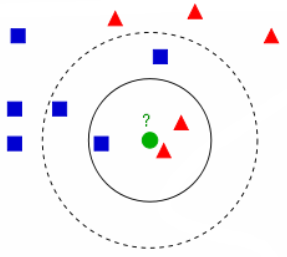
- K-nearest neighbors algorithm (k-NN)
- Arboles de decisión
- Random Forest
- Maquinas de vectores de soporte

K-nearest neighbors algorithm (k-NN)

- Es muy método muy simple para clasificar datos
- Solo requiere de un hiperparámetro
- Funciona bien en varias situaciones

K-nearest neighbors algorithm (k-NN)

El objeto se asigna a la clase más común entre sus k-vecinos más cercanos (k es un número entero positivo, típicamente pequeño).



En el ejemplo, para k = 3 el circulo verde es clasificado con la clase triángulo, ya que hay solo un cuadrado y 2 triángulos, dentro del círculo que los contiene. Si k = 5 este es clasificado con la clase cuadrado, ya que hay 2 triángulos y 3 cuadrados, dentro del círculo externo.

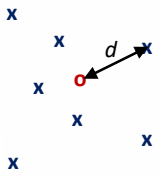
K-nearest neighbors algorithm (k-NN)

Cada objeto se caracteriza un conjunto de *n* parámetros:

$$x_i = \{x_i^1, x_i^2, x_i^3, \dots, x_i^n\}$$

La distancia se mide usualmente mediante la distancia euclidiana:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (x_i^r - x_j^r)^2}$$

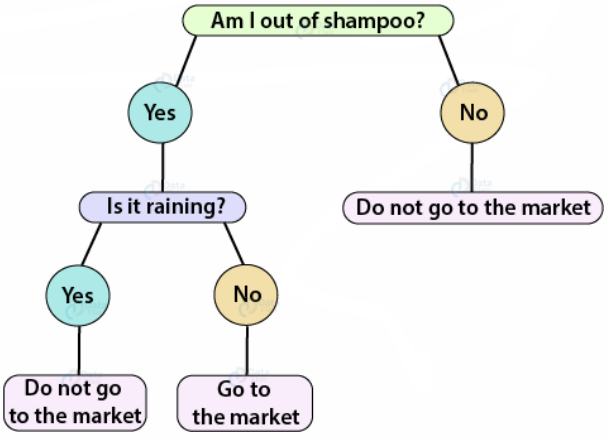


K-nearest neighbors algorithm (k-NN)

La mejor elección de k depende fundamentalmente de los datos:

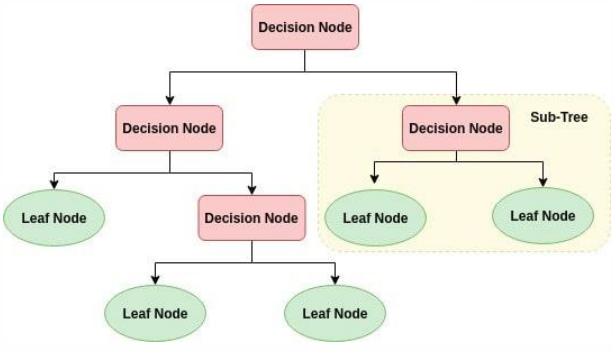
- Valores pequeños de k son sensibles al ruido en los datos.
- Valores grandes de k reducen el efecto de ruido, pero crean límites entre clases parecidas.

Arboles de decisión



Arboles de decisión

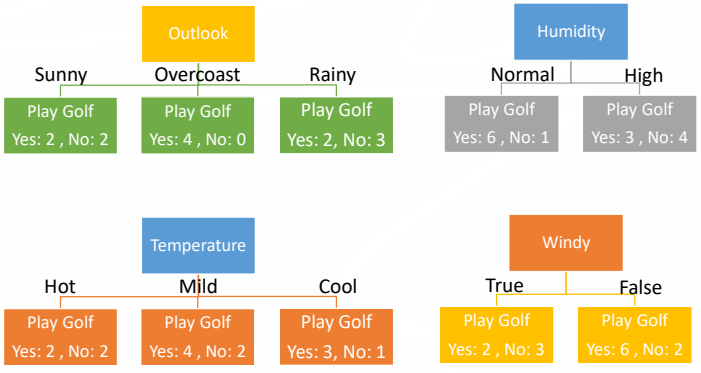
Un árbol de decisión es una estructura de similar a un diagrama de flujo donde un nodo interno representa una característica (o atributo), la rama representa una regla de decisión y cada nodo hoja (leaf) representa el resultado.



Arboles de decisión

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcoast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcoast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcoast	Mild	High	True	Yes
Overcoast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Arboles de decisión



Lo ideal sería encontrar una pregunta que separe complemente las clases. ¿Como podemos medir si una pregunta hace una mejor separación de clases que otra?

Medidas de Selección de Atributos

La medidas de selección de atributos (ASM por sus siglas en ingles) permiten seleccionar el parámetro que divide los datos de la mejor manera posible, para realizar esto hace un ranking entre los distintos parámetros.

En el caso de un parámetro de valor continuo, también se debe definir el valor para dividir las ramas.

Las medidas de selección más populares son ganancia de información e índice de Gini.

Entropía

La Entropía mide cuan homogénea es un conjunto de datos. Si la muestra es completamente homogénea (misma clase), la entropía es cero y si la muestra está dividida en partes iguales, entonces tiene una entropía de uno.

- División con un parámetro:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

Entropía

- División con dos parámetros:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Ganancia de Información

La ganancia de información se basa en la disminución de la entropía después de que un conjunto de datos se divide en un atributo.

La construcción de un árbol de decisión consiste en encontrar un atributo que devuelva la mayor ganancia de información (es decir, las ramas más homogéneas).

Ganancia de Información

1. Calcular la entropía del objetivo.

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

2. El conjunto de datos se divide en los diferentes atributos y se calcula la entropía de cada rama. La entropía resultante se resta de la entropía antes de la división. El resultado es la ganancia de información, o disminución de la entropía.

$$\text{Gain}(T, X) = \text{Entropy}(T) - \text{Entropy}(T, X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)
= 0.940 – 0.693 = 0.247

Ganancia de la Información

Se selecciona el parámetro con la mayor ganancia. El proceso se repite en cada rama hasta llegar a entropía 0 (hoja).

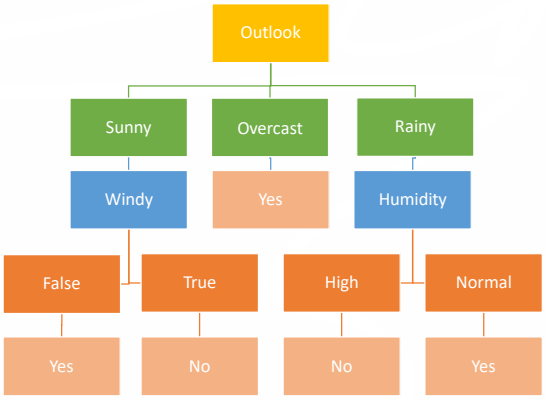
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

Ganancia de la Información



Índice de diversidad de Gini

El índice de Gini también mide la homogeneidad de un conjunto. Un mayor índice implica menos homogeneidad.

$$G(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) G(C|A_{ij})$$

- A_i es el atributo para ramificar el árbol.
- M_i es el número de valores diferentes del atributo A_i.
- $p(A_{ij})$ es la probabilidad de que A_i tome su j-ésimo valor $(1 <= j <= M_i)$.

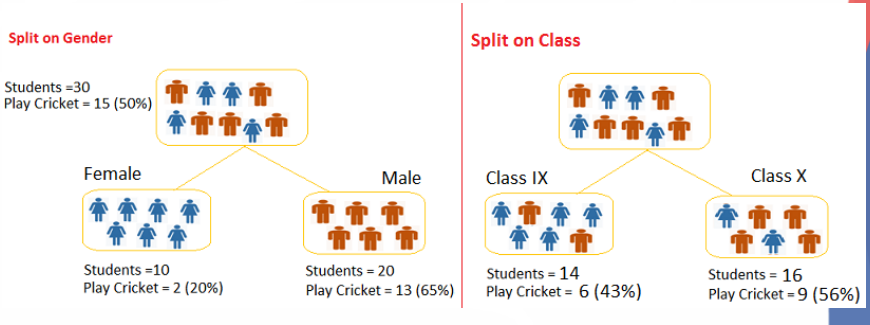
Índice de diversidad de Gini

$$G(C|A_{ij}) = -\sum_{k=1}^{J} p(C_k|A_{ij})p(\neg C_k|A_{ij}) =$$

$$= 1 - \sum_{k=1}^{J} p^2(C_k|A_{ij})$$

- $p(C_k|A_{ij})$ es la probabilidad de que un ejemplo pertenezca a la clase C_k cuando su atributo A_i toma su j-ésimo valor.
- $p(\neg C_k|A_{ij})$ es 1 - $p(C_k|A_{ij})$.

Ejemplo



Ejemplo

$$G(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) G(C|A_{ij})$$

- **Según el genero:**
  - Gini para sub-node Femenino =1-[(0.2)*(0.2)+(0.8)*(0.8)]=0.32
  - Gini para sub-node Masculino = 1-[(0.65)*(0.65)+(0.35)*(0.35)]=0.45
  - Índice para el parámetro genero= (10/30)*0.32+(20/30)*0.45 = **0.41**

- **Según la clase:**
  - Gini para sub-node Class IX =1-[(0.43)*(0.43)+(0.57)*(0.57)]=0.49
  - Gini para sub-node Class X =1-[(0.56)*(0.56)+(0.44)*(0.44)]=0.49
  - Índice para el parámetro clase = (14/30)*0.49+(16/30)*0.49 = **0.49**

El índice de Gini para dividir según género es menor que dividir según clase, por lo tanto, la división se hace según género.

Random Forest

Random forest también conocidos en castellano como "Bosques Aleatorios" es una combinación de árboles de decisión.

Cada árbol es entrenado con un sub-conjunto de los datos de entrenamiento.

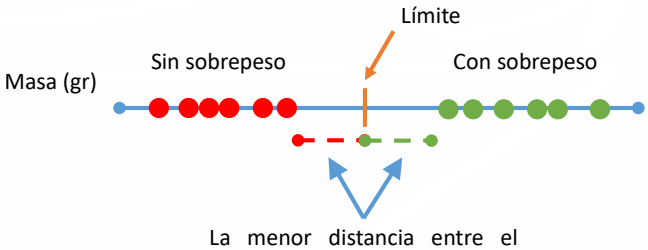
Además, durante la construcción del árbol, la división seleccionada ya no es la mejor división entre todos los parámetros. Sino que la división seleccionada es la mejor división entre un sub-conjunto aleatorio de los parámetros.

Random Forest

Como resultado cada árbol de decisión es ligeramente diferente al resto.

Para la predicción de un nuevo dato, este es clasificado por cada árbol de decisión. La clase que se repita más veces es la predicción del bosque.

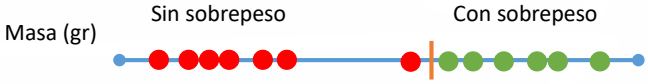
Maquinas de vectores de soporte



La menor distancia entre el límite y las observaciones se denomina el margen.

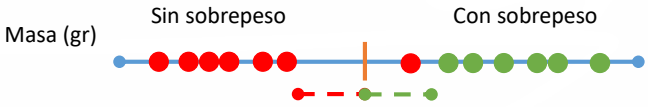
El margen se maximiza en el punto medio.

Maquinas de vectores de soporte



Este método es muy sensible a los valores atípicos en los datos de entrenamiento.

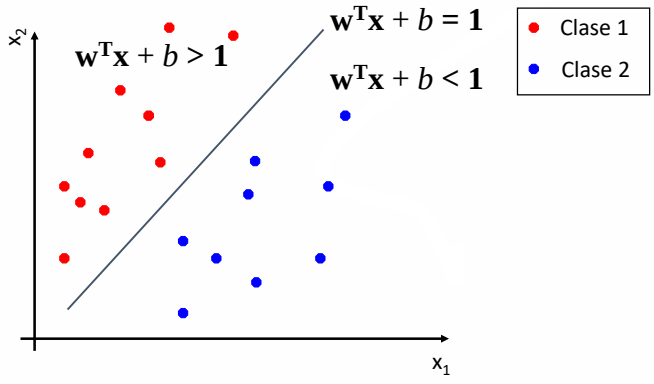
Maquinas de vectores de soporte



Para definir un margen que no sea tan sensible a los valores atípicos debemos permitir clasificaciones erróneas en el entrenamiento.

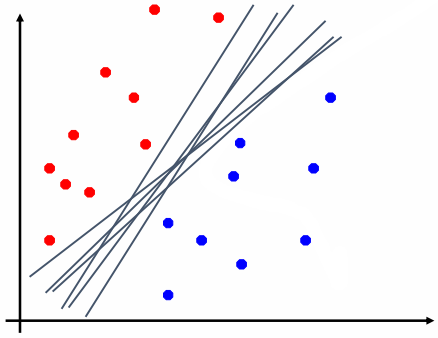
Maquinas de vectores de soporte

La clasificación binaria se puede ver como la tarea de separar clases en un espacio de parámetros:



Maquinas de vectores de soporte

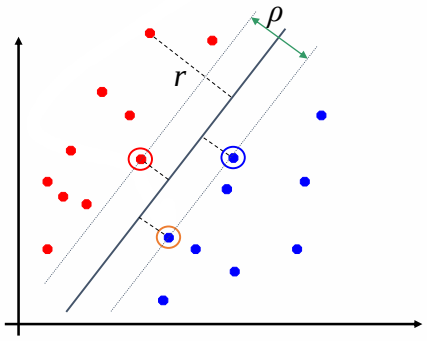
¿Cuál de estos separadores lineales es el mejor?



Maquinas de vectores de soporte

- *r* es la distancia del punto al plano separados
- Los datos más cercanos al plano se denominan vectores de soporte
- ρ es el margen de separación entre los vectores de soporte

Las maquinas de vectores de soporte seleccionan el plano que maximice el margen ρ.



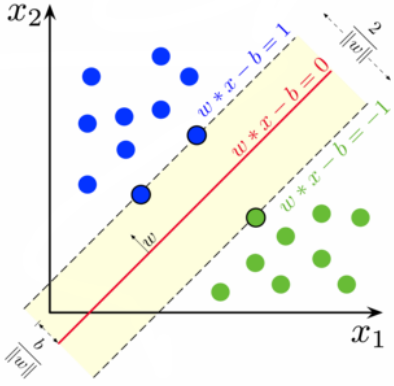
Formulación

$$\min_{\mathbf{w},\mathbf{b}} \|\mathbf{w}\|^2$$

sujeto a:

$$y_i \cdot (\mathbf{w}^T \mathbf{\Phi}(\mathbf{x}) + \mathbf{b}) \geq 1$$

y_i es igual a 1 para la clase 1 e igual a -1 para la clase 2.



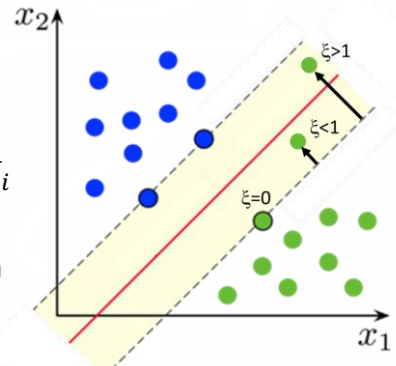
Formulación

$$\min_{\mathbf{w},\mathbf{b},\boldsymbol{\xi}} \|\mathbf{w}\|^2 + C\sum \xi_i$$

sujeto a:

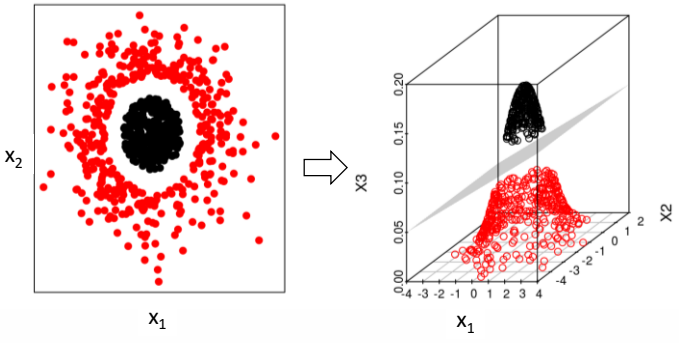
$$y_i \cdot (\mathbf{w}^T \Phi(\mathbf{x}) + \mathbf{b}) \geq 1 - \xi_i$$

C es un factor de penalización para el error.



Maquinas de vectores de soporte

Si los datos no son linealmente separable, se realiza una transformación del espacio a uno de mayor dimensión en donde si son separables por un plano.



Maquinas de vectores de soporte

Esta transformación de realiza por medio de funciones Kernel de manera que $K(x,y) =< \varphi(x), \varphi(y) >$. Algunas funciones típicas son:

- Polinomial:

$$K(x,y) = (1 + x^T y)^d$$

- Radial Basis Function (RBF):

$$K(x,y) = \exp(^{-(x-y)^T(x-y)}/_{2\sigma^2})$$

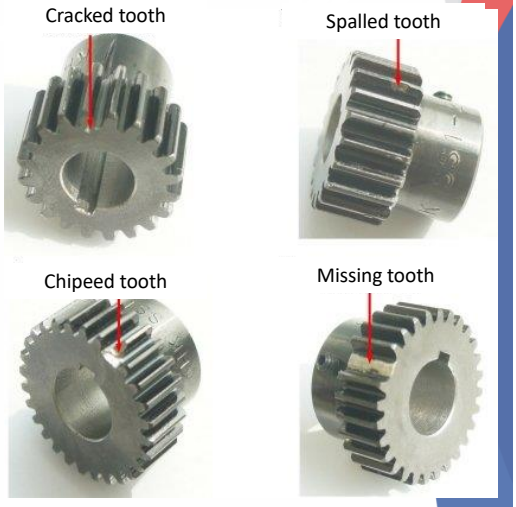
- Sigmoid:

$$K(x,y) = \tanh(\rho_1 x^T y + \rho_2)$$

Ejemplo 3

Se cuenta con datos de vibración obtenidos en un montaje experimental de una caja de engranajes operando en estado saludable y con distintas fallas en los engranajes.

Provenientes de la siguiente base de datos: https://figshare.com/articles/Gear_Fault_Data/6127874/1

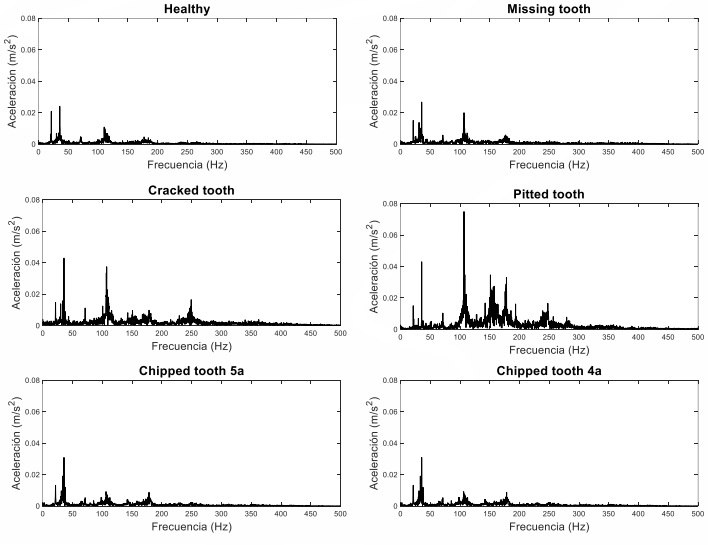


Ejemplo 3

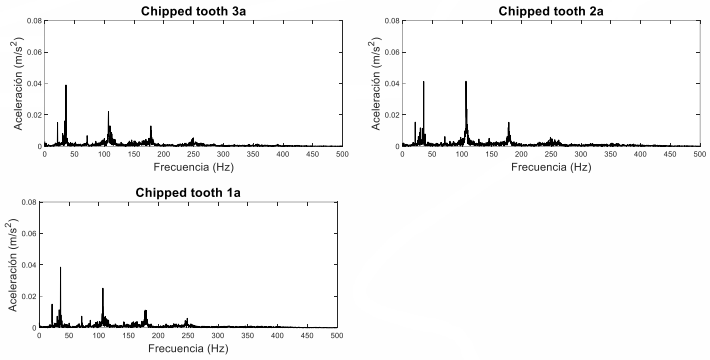
La base de datos considera 9 estados de salud:

1. Healthy
2. Missing
3. Crack
4. Spall
5. Chip5a
6. Chip4a
7. Chip3a
8. Chip2a
9. Chip1a

Ejemplo 3 – Extracción de parámetros



Ejemplo 3 – Extracción de parámetros



Ejemplo 3

Para cada estado se tienen 104 mediciones, cada medición consiste en 3600 datos de vibración temporales. Para cada medición se calculó la transformada de Fourier y luego se extrajeron 34 parámetros utilizando PCA. Los resultados se entregan en las matrices:

- Xtodos.npy: array de Python de dimensión 936x34 con los 34 parámetros para las 104x9 mediciones.
- Label.npy: array de 936x1 con las etiquetas de las mediciones.

Ejemplo 3

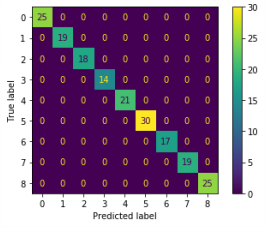
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report
from sklearn.metrics import plot_confusion_matrix

Xt=np.load('Xtodos.npy')
Yt=np.load('Label.npy')

X_train, X_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.20)
```


Ejemplo 3

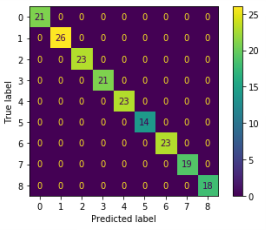
```
# Model = KNeighborsClassifier(n_neighbors=5)
# Model=DecisionTreeClassifier(criterion='entropy') #el criterio puede ser gini o entropy
# Model=RandomForestClassifier(n_estimators=10)
Model = SVC(kernel='rbf',C=2) #'linear', 'poly', 'rbf', 'sigmoid'
Model.fit(X_train, y_train)
Yp=Model.predict(X_test)

plot_confusion_matrix(Model, X_test, y_test)
plt.show()

target_names = ['Healthy', 'Missing', 'Crack', 'Spall', 'chip5a', 'chip4a', 'chip3a', 'chip2a', 'chip1a']
print(classification_report(y_test, Yp, target_names=target_names))
```

Ejemplo

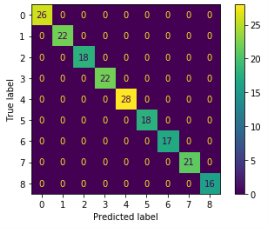
K-nearest



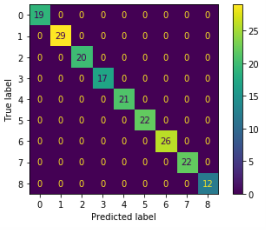
Árbol de decisión



Random Forest



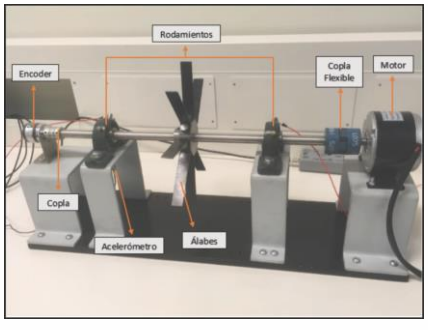
SVM



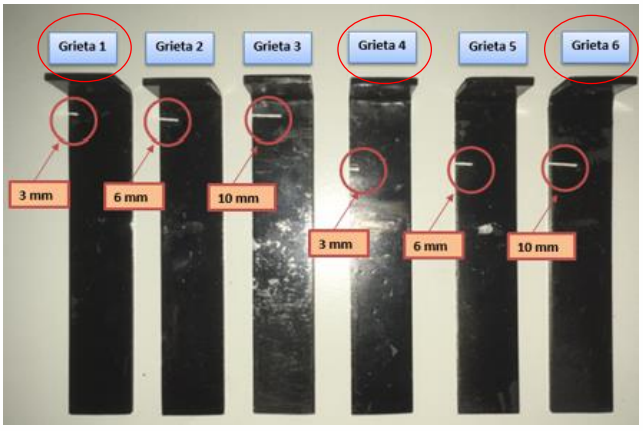
Exactitud=1.00

Tarea 1

Se cuenta con el sistema rotor de la figura. Este sistema consiste en un motor eléctrico acoplado a un eje soportado por dos rodamientos. Entre los rodamientos se encuentra una turbina con álabes intercambiables. Bajo el rodamiento izquierdo se encuentra instalado un acelerómetro piezoeléctrico.



Tarea 1



Tarea 1

- De los datos experimentales, se extrajeron 8 parámetros estadísticos y 8 parámetros asociados a bandas de frecuencias (Fourier).
- Luego se utilizó el método de Kernel PCA para obtener un set de 3 parámetros.
- Los parámetros asociados a los cuatro casos se entregan en los siguientes archivos:
  - Pa0.npy: parámetros del caso sin daño
  - Pa1.npy: parámetros del caso Grieta 1
  - Pa2.npy: parámetros del caso Grieta 4
  - Pa3.npy: parámetros del caso Grieta 6
- Implemente los distintos métodos de clasificación vistos en clases y evalué sus resultados.

Tarea 1

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


Pa0=np.load('Pa0.npy')
Pa1=np.load('Pa1.npy')
Pa2=np.load('Pa2.npy')
Pa3=np.load('Pa3.npy')
Xt=np.concatenate((Pa0,Pa1,Pa2,Pa3),axis=0)

#etiquetas
y0=np.zeros(23);
y1=1*np.ones(23);
y2=2*np.ones(23);
y3=3*np.ones(23);
Yt=np.concatenate((y0,y1,y2,y3),axis=0)
```


Selección del Modelo

Entrenamiento y pruebas

Entrenar un modelo y evaluarlo con los mismos datos es un error metodológico: un modelo que simplemente repite las etiquetas de las muestras que acaba de ver tendría una puntuación perfecta pero fallaría en predecir datos no vistos. Esta situación se llama sobreajuste.

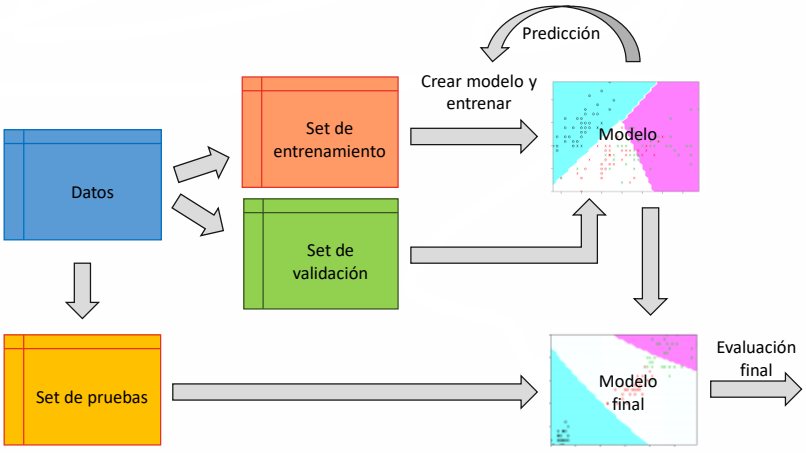
Para evitar este problema, una práctica común cuando se entrena un modelo de aprendizaje supervisado es separar los datos disponibles en un conjunto para entrenar y otro para evaluar (conjunto de prueba).

Entrenamiento y pruebas

Al evaluar diferentes configuraciones (hiperparámetros) para los modelos, todavía existe el riesgo de que el conjunto de prueba se sobreajuste porque los parámetros se pueden seleccionar hasta que el estimador funcione de manera óptima para ese set.

Para resolver este problema se usa un tercer set de datos, denominado datos de validación. El modelo se entrena con los datos de entrenamiento, y se realizan pruebas con distintas configuraciones evaluando en el set de validación, cuando se encuentra la mejor configuración se hace la evaluación final en el conjunto de prueba.

Entrenamiento y pruebas



Validación cruzada

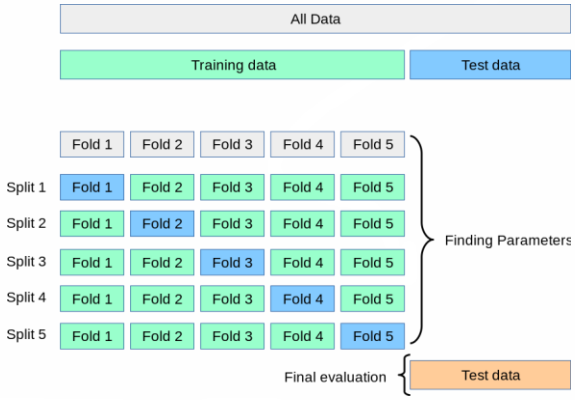
Sin embargo, al dividir los datos disponibles en tres conjuntos, reducimos el número de datos que se pueden usar para entrenar el modelo, y los resultados pueden depender de la elección para los conjuntos de entrenamiento y validación.

Una solución a este problema es un procedimiento llamado validación cruzada (CV). Los datos se separan en conjunto de entrenamiento y prueba. En el enfoque básico, llamado k-fold CV, el conjunto de entrenamiento se divide en k conjuntos más pequeños y se realiza lo siguiente:

- Se entrena un modelo utilizando k-1 conjuntos como datos de entrenamiento.
- El modelo se evalúa con el conjunto restante.
- Se repite el procedimiento para todas las combinaciones y se calcula el rendimiento promedio.

Validación cruzada

Este enfoque puede ser computacionalmente costoso, pero desperdicia pocos datos, lo cual es una ventaja importante en problemas donde el número de muestras es muy pequeño.



Ejemplo 4

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_validate


Xt=np.load('Xtodos.npy')
Xt=Xt[:,1:5]
Yt=np.load('Label.npy')

X_train, X_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.20)

Model = KNeighborsClassifier(n_neighbors=10)
# Model=DecisionTreeClassifier(criterion='entropy') #el criterio puede ser gini o entropy
# Model=RandomForestClassifier(n_estimators=10)
# Model = SVC(kernel='rbf',C=2) #'linear', 'poly', 'rbf', 'sigmoid'

scores = cross_validate(Model, X_train, y_train,cv=5)

print(scores['test_score'])
```

Ejemplo 4

	1	2	3	4	5	Promedio
SVC	0.97	0.95	0.97	0.96	0.95	0.96
Random Forest	0.93	0.93	0.91	0.92	0.94	0.93
Árbol de decisión	0.92	0.87	0.86	0.86	0.91	0.88
K-nearest	0.92	0.93	0.92	0.97	0.91	0.93

Métodos combinados

En el campo del aprendizaje automático, los métodos combinados (métodos de ensemble) utilizan múltiples algoritmos de aprendizaje para obtener un rendimiento predictivo que mejore el que podría obtenerse por medio de cualquiera de los algoritmos de aprendizaje individuales que lo constituyen.

Métodos combinados

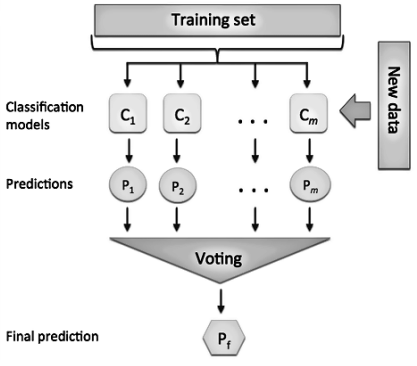
Los algoritmos de aprendizaje supervisado buscan en un espacio de hipótesis la más adecuada que haga buenas predicciones con un problema en particular. En general, esta tarea es muy complicada.

La idea de los métodos combinados es considerar múltiples hipótesis simultáneamente para formar una hipótesis que, con suerte (y la ayuda de algunos teoremas esenciales), se comporte mejor.

Clasificación por votación

La idea es combinar distintos métodos de clasificación y usar un voto mayoritario para predecir una clase.

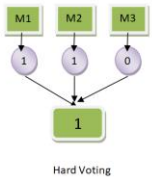
Un clasificador de este tipo puede ser útil cuando se tiene un conjunto de modelos de rendimiento similar con el fin de equilibrar sus debilidades individuales.



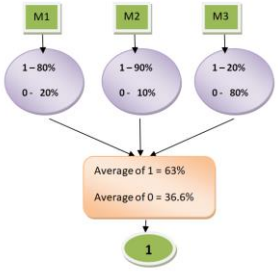
Clasificación por votación

Votación dura: la salida de cada algoritmo es una clase y la predicción es la clase con la mayoría más alta de votos.

Votación suave: la clase de salida es la predicción basada en el promedio de probabilidad dado a esa clase.



Hard Voting



Soft Voting

Ejemplo 5

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_validate
from sklearn.ensemble import VotingClassifier


Xt=np.load('Xtodos.npy')
Xt=Xt[:,1:5]
Yt=np.load('Label.npy')

X_train, X_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.20)

Model1 = KNeighborsClassifier(n_neighbors=10)
Model2=RandomForestClassifier(n_estimators=10)
Model3 = SVC(kernel='rbf',C=2) #'linear', 'poly', 'rbf', 'sigmoid'

Model= VotingClassifier(estimators=[('Knearest', Model1), ('RandomForest', Model2), ('svc',
Model3)],voting='hard')
```

Ejemplo 5

```
Model1 = Model1.fit(X_train, y_train)
Model2 = Model2.fit(X_train, y_train)
Model3 = Model3.fit(X_train, y_train)

scores1 = cross_validate(Model1, X_train, y_train,cv=5)
print("model1= ", scores1['test_score'], "promedio= ",np.mean(scores1['test_score']))

scores2 = cross_validate(Model2, X_train, y_train,cv=5)
print("model2= ", scores2['test_score'], "promedio= ",np.mean(scores2['test_score']))

scores3 = cross_validate(Model3, X_train, y_train,cv=5)
print("model3= ", scores3['test_score'], "promedio= ",np.mean(scores3['test_score']))

scores = cross_validate(Model, X_train, y_train,cv=5)
print("Combinados= ", scores['test_score'], "promedio= ",np.mean(scores['test_score']))
```


Ejemplo 5

	1	2	3	4	5	Promedio
Model 1	0.91	0.93	0.94	0.95	0.91	0.93
Model 2	0.89	0.91	0.95	0.94	0.89	0.91
Model 3	0.93	0.96	0.96	0.96	0.94	0.95
Combinados	0.93	0.95	0.95	0.97	0.92	0.94

Selección de hiperparámetros

Los hiperparámetros son parámetros que no se aprenden directamente dentro de los modelos y que se deben especificar.

Algoritmo	Hiperparámetros
K-nearest neighbors algorithm (k-NN)	Numero de vecinos, función para los pesos (uniforme, basada en la distancia)
Arboles de decisión	Criterio de selección: Índice de Gini o Ganancia de la Información
Random Forest	Numero de arboles, criterio de selección.
Maquinas de vectores de soporte	Parámetro de penalización C, función de kernel, parámetros de la función de kernel.

Selección de hiperparámetros

Se recomienda seleccionar el conjunto de hiperparámetros que entreguen los mejores resultados de validación cruzada. Para encontrar el mejor conjunto de parámetros hay dos estrategias frecuentes:

- Grid Search: se evalúan todas las combinaciones de parámetros dentro de ciertos rangos.
- Randomized Search: se evalúan un numero fijo de combinaciones seleccionadas aleatoriamente.

Ejemplo 6

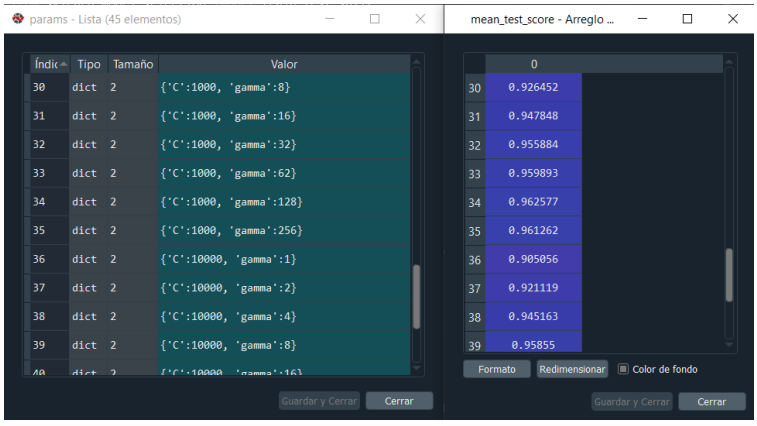
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from scipy import stats


Xt=np.load('Xtodos.npy')
Xt=Xt[:,1:5]
Yt=np.load('Label.npy')

X_train, X_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.20)

Model= SVC(kernel='rbf')
```


Ejemplo 6

```
#Grid Search
param_grid = {'C': [1, 10, 100, 1000, 10000], 'gamma': [1, 2, 4, 8, 16, 32, 62, 128, 256]}

grid_search = GridSearchCV(Model, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
Resultados_GridSearch=grid_search.cv_results_;
print(max(Resultados_GridSearch['mean_test_score']))

#Random Search
param_distr = {'C': stats.randint(1, 10000), 'gamma': stats.uniform(1, 256)}

random_search = RandomizedSearchCV(Model, param_distributions=param_distr,n_iter=50, cv=5)
random_search.fit(X_train, y_train)
Resultados_RandomSearch=random_search.cv_results_;
print(max(Resultados_RandomSearch['mean_test_score']))
```

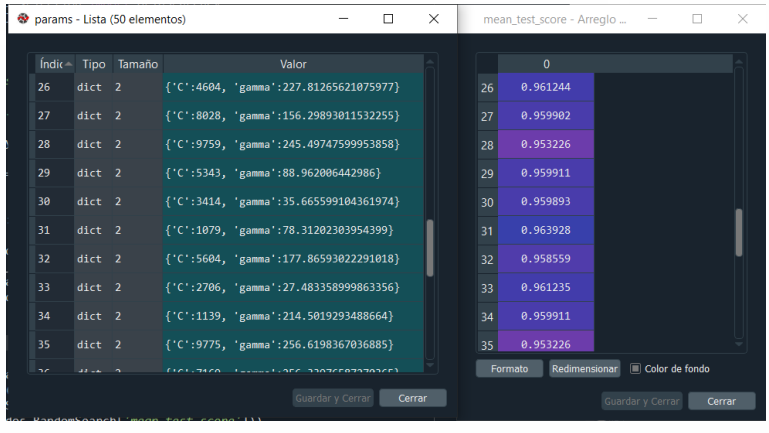
Ejemplo 6

- Resultados Grid Search



Ejemplo 6

- Resultados Random Search



Tarea 2

Utilice los mismos datos de la tarea 1 y el método de SVC. Determine el kernel y los parámetros óptimos haciendo un Grid Search.

- Lineal: $x^T y$
- Radial Basis Function (RBF): $\exp(-\gamma(x-y)^T(x-y))$
- Polinomial: $(r + \gamma\, x^T y)^d$
- Sigmoid: $\tanh(\gamma\, x^T y + r)$

Kernel	Parámetros
Lineal	'C'
Radial	'C', 'gamma'
Polinomial	'C', 'coef0' , 'gamma', 'degree'
Sigmoid	'C', 'gamma', 'coef0'

Métodos de agrupamiento

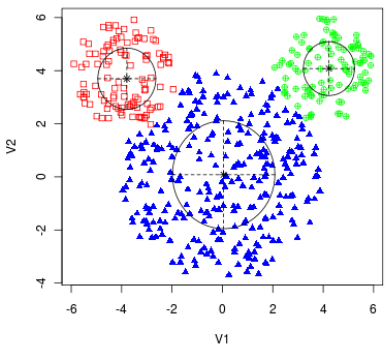
Métodos de agrupamiento

- La mayoría de los sistemas críticos están siendo monitoreados de forma continua -> millones de datos que procesar
- Etiquetar un conjunto de datos de esta magnitud requiere un uso intensivo de recursos.
- Además, existe la suposición de que se conoce todo acerca de las fallas preestablecidas. Esto restringe la capacidad del modelo supervisado para generalizar.

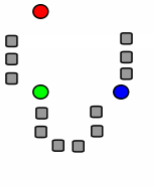
Métodos de agrupamiento

- La agrupación se realiza mediante algoritmos de agrupamiento (clustering), como son k-means, DBSCAN y agrupamiento jerárquico, entre otros.
- Estos algoritmos agrupan los datos utilizando, en general, criterios de distancia o similitud.
- Los datos de un mismo grupo comparten propiedades comunes.

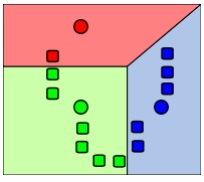
Métodos de agrupamiento



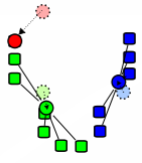
K-Means



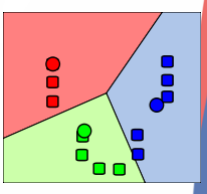
1) K centroides iniciales



2) *k* grupos son generados asociándole los puntos más cercanos

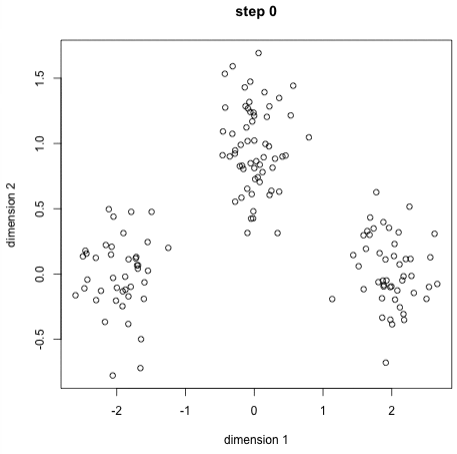


3) El centroide de cada grupo se recalcula



4) Se repiten los pasos 2 y 3 hasta que converga.

K-Means



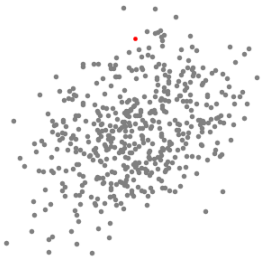
K-Means

- **Ventajas**: Es bastante rápido, ya que lo único que estamos haciendo es calcular las distancias entre los puntos y los centros grupales.

- **Desventajas**: (1) Se debe seleccionar cuántos grupos / clases hay. Esto no siempre es trivial. (2) K-means comienza con una elección aleatoria de los centroides y, por lo tanto, puede producir diferentes resultados en diferentes ejecuciones del algoritmo. Esto implica que los resultados pueden no ser repetibles.

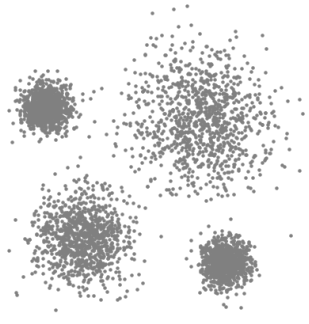
Mean-Shift Clustering

- Comenzamos con una ventana deslizante circular centrada en un punto (seleccionado al azar) y con radio r.
- En cada iteración, la ventana deslizante se desplaza hacia regiones de mayor densidad al desplazar el punto central al valor medio dentro de la ventana.
- Continuamos cambiando la ventana deslizante según el promedio hasta que no haya ninguna dirección en la que un cambio pueda acomodar más puntos dentro de la ventana.



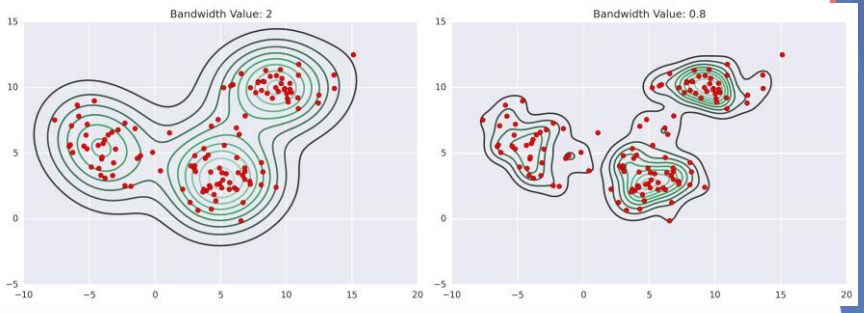
Mean-Shift Clustering

- El proceso anterior se repite para múltiples ventanas. Cuando varias ventanas deslizantes se superponen, la ventana que contiene la mayor cantidad de puntos se conserva.



Mean-Shift Clustering

- **Ventajas**: No es necesario definir previamente el número de agrupamientos.
- **Desventajas**: No es trivial elegir el tamaño/radio de la ventana.

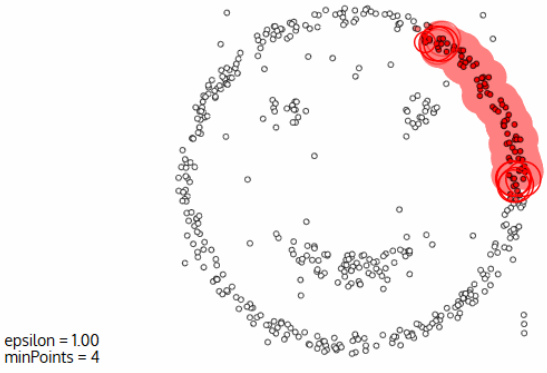


DBSCAN

DBSCAN es un algoritmo de agrupamiento basado en densidad similar al Mean-Shift, pero con un par de ventajas. Su funcionamiento es el siguiente:

1. Se selecciona un dato cualquiera, se determinan todos los vecinos de ese punto que estén a un distancia ε.
2. Si hay suficientes datos dentro de la ventana (de acuerdo a lo especificado) entonces ese dato y sus vecinos son etiquetados como dentro del grupo. En caso contrario se etiqueta como ruido.
3. El proceso 1-2 se repite para todos los datos que fueron añadidos al grupo.
4. El proceso finaliza cuando todos los puntos en el grupo han sido "visitados" y etiquetados como dentro del grupo o como "ruido".
5. Una vez que se han etiquetado todos los datos de un grupo, el proceso se repite para otro dato fuera de ese grupo que no haya sido "visitado", hasta que todos los datos sean "visitados".

DBSCAN



DBSCAN

- **Ventajas**: (1) No es necesario definir previamente el número de agrupamientos. (2) Permite identificar “outliers” como ruido (a diferencia de Mean Shifting). (3) Puede identificar grupos de formas arbitrarias.

- **Desventajas**: No funciona muy bien en casos con densidad variable. Esto es porque la configuración de la distancia ε y el número mínimo de datos en la ventana podría variar de un grupo a otro.

Gaussian Mixture Models

Uno de los problemas de K-means es que sirve solo para grupos de forma circular.

Los Gaussian Mixture Models (GMM) dan más flexibilidad que K-Means, ya que asumen que los datos están distribuidos con una Gaussiana. De esa manera, tenemos dos parámetros para describir la forma de los grupos: la media y la desviación estándar.

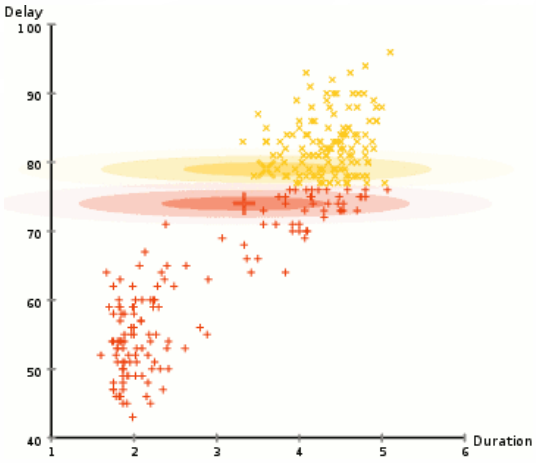
Si consideramos un caso dos dimensiones, esto significa que los grupos pueden tomar cualquier tipo de forma elíptica (ya que tenemos una desviación estándar en las direcciones x e y).

Gaussian Mixture Models

Para encontrar los parámetros de cada grupo (la media y la desviación estándar) utilizaremos un algoritmo de optimización llamado Expectation–Maximization (EM).

1. Comenzamos seleccionando el número de grupos (a igual que con K-Means) y definiendo de forma aleatoriamente los parámetros para cada grupo.
2. Dadas las distribuciones gaussianas para cada grupo, se calcula la probabilidad de que cada dato pertenezca a un grupo en particular.
3. Se calcula un nuevo conjunto de parámetros para las distribuciones gaussianas, de modo de maximizar las probabilidades de los datos dentro de los grupo.
4. Se repiten los pasos 1-3 hasta la convergencia.

Gaussian Mixture Models



Gaussian Mixture Models

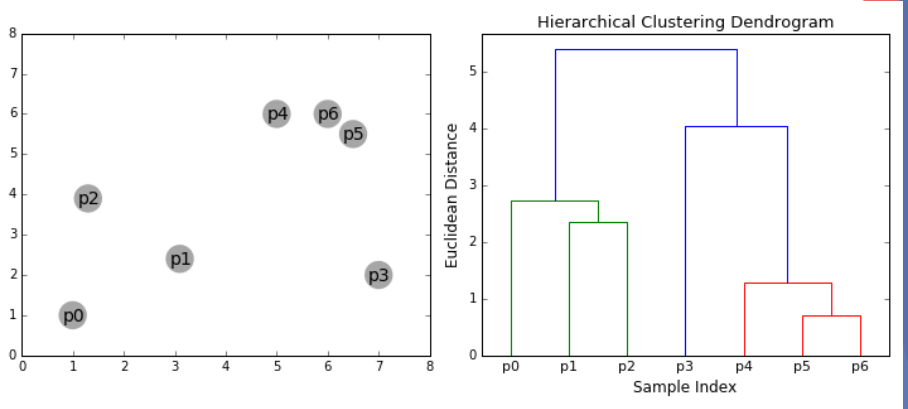
- **Ventajas**: (1) GMM son más flexibles que K-Means al poder tomar cualquier forma elíptica, de hecho K-Means es un caso particular de GMM. (2) Dado que los GMM utilizan probabilidades, un dato puede pertenecer a más de un grupo (con distintas probabilidades).
- **Desventajas**: (1) Se debe seleccionar cuántos grupos / clases hay. Esto no siempre es trivial. (2) CMM comienza con una elección aleatoria de las distribuciones y, por lo tanto, puede producir diferentes resultados en diferentes ejecuciones del algoritmo.

Agglomerative Hierarchical Clustering

El algoritmo Agglomerative Hierarchical Clustering (HAC) trata cada dato como un solo grupo en un principio y luego combina (o aglomera) pares de grupos hasta que todos se agrupan en un solo grupo que contiene todos los datos. Sigue los siguientes pasos:

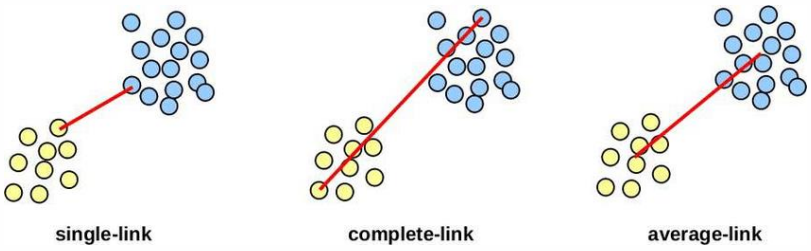
1. Comenzamos tratando cada dato como un único grupo. Luego seleccionamos una métrica que mida la distancia entre dos grupos.
2. En cada iteración se combinan dos grupos un uno, se combinan siempre los grupos con la menor distancia entre ellos.
3. Se repite el paso 2 hasta que se llega a un único grupo.

Agglomerative Hierarchical Clustering



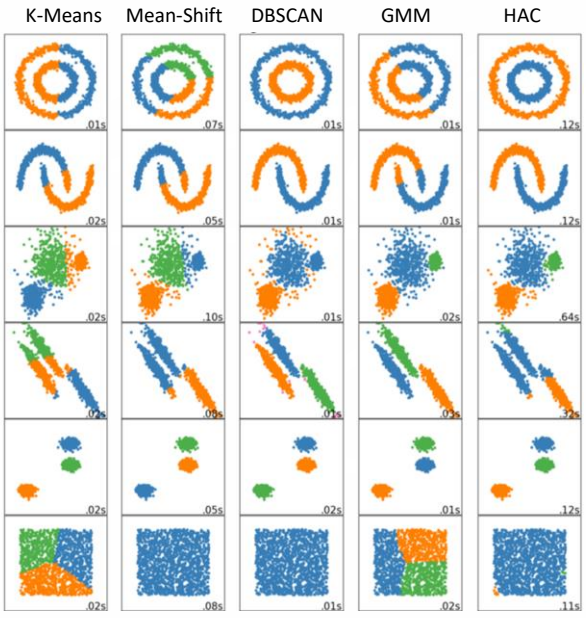
Agglomerative Hierarchical Clustering

- **Enlace único**: distancia menor entre dos puntos en cada grupo.
- **Vinculación completa**: distancia mayor entre dos puntos en cada grupo.
- **Vinculación promedio**: distancia promedio entre cada punto en un grupo a cada punto en el otro grupo.



Agglomerative Hierarchical Clustering

- **Ventajas**: (1) No es necesario especificar la cantidad de grupos e incluso podemos seleccionar qué cantidad de grupos se ve mejor. (2) El algoritmo funciona bien con distintas métricas para la distancia.
- **Desventajas**: Es muy costoso computacionalmente.



Ejemplo 7

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.cluster import MeanShift
from sklearn.cluster import DBSCAN
from sklearn import mixture
from sklearn.cluster import AgglomerativeClustering
from mpl_toolkits.mplot3d import Axes3D
from itertools import cycle

Xt=np.load('Xtodos.npy')
Yt=np.load('Label.npy')[:,0]

#Metodo de agrupamiento

Model=KMeans(n_clusters=9)
nombre_mod='Kmeans'

# Model=MeanShift(bandwidth=0.06)
# nombre_mod='Mean Shift'

# Model=DBSCAN(eps=0.02, min_samples=5)
# nombre_mod='DBSCAN'
```

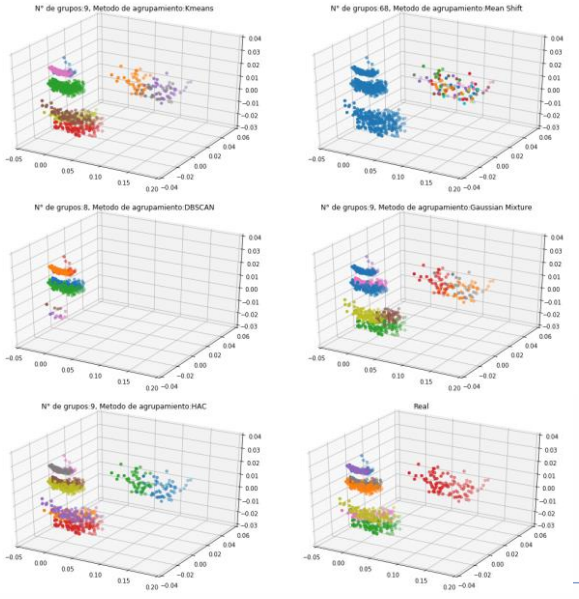
Ejemplo 7

```
# Model=mixture.GaussianMixture(n_components=4)
# nombre_mod='Gaussian Mixture'

# Model=AgglomerativeClustering(n_clusters=4)
# nombre_mod='HAC'

Yp=Model.fit_predict(Xt)
labels_unique = np.unique(Yp)
n_clusters_ = len(labels_unique)

fig = plt.figure(figsize=(9,6))
ax = fig.add_subplot(111, projection='3d')
colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
for k, col in zip(range(n_clusters_), colors):
    my_members = Yp == k
    ax.scatter(Xt[my_members, 0], Xt[my_members, 1], Xt[my_members, 2], col + '.')
plt.title('N° de grupos:' + str(n_clusters_) + ', Metodo de agrupamiento:' + nombre_mod)
plt.show()
```


Ejemplo 7

```
fig = plt.figure(figsize=(9,6))
ax = fig.add_subplot(111, projection='3d')
colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
for k, col in zip(range(9), colors):
    my_members = Yt == k
    ax.scatter(Xt[my_members, 0], Xt[my_members, 1], Xt[my_members, 2], col + '.')
plt.title('Real')
plt.show()
```

Ejemplo 7



Evaluación de los métodos de agrupamiento

Evaluación de los métodos de agrupamiento

Evaluar el rendimiento de un algoritmo de agrupamiento no es tan trivial como calcular la exactitud de un algoritmo de clasificación supervisado. En particular, cualquier métrica de evaluación no debe tener en cuenta los valores absolutos de las etiquetas del clúster, sino que si esta agrupación define separaciones de los datos similares a algún conjunto de clases de verdad básica o que satisfacen que los miembros pertenecientes a la misma clase son más similares que miembros de diferentes clases según alguna métrica de similitud.

Métricas supervisadas

Índice Rand ajustado

Si se conocen las agrupaciones reales de los datos y las asignaciones obtenidas con un algoritmo de agrupamiento, el índice Rand ajustado es una función que mide la similitud de las dos asignaciones, ignorando las permutaciones.

Índice Rand ajustado

Consideremos dos agrupaciones distintas para los datos: $X = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots \boldsymbol{X}_n\}$ e $Y = \{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots \boldsymbol{Y}_n\}$. La superposición entre X e Y se puede resumir en una tabla de contingencia, donde cada entrada n_{ij} denota el número de objetos en común entre $\boldsymbol{X}_i$ e $\boldsymbol{Y}_j$.

$X \backslash Y$	Y_1	Y_2	$\cdots$	Y_s	sums
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
sums	b_1	b_2	$\cdots$	b_s	

Índice Rand ajustado

El índice de Rand ajustado que utiliza el modelo de permutación es

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}$$

Índice Rand ajustado

Características:

- Las asignaciones de etiquetas aleatorias (uniformes) tienen una puntuación ARI cercana a 0.0
- Rango acotado [-1, 1]: los valores negativos son malos, agrupaciones similares tienen un ARI positivo, 1.0 es la puntuación de coincidencia perfecta.
- No se hace ninguna suposición sobre la estructura del clúster.

Información Mutua Normalizada

La información mutua es una función que mide la similitud entre dos conjuntos, ignorando las permutaciones. Para evaluar los métodos de agrupamiento de utiliza la Información Mutua Normalizada (NMI).

Información Mutua Normalizada

Consideremos dos asignaciones de etiquetas (de los mismos N objetos) U y V. Las entropías de U y V vienen dadas por:

$$H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i)) \qquad H(V) = -\sum_{j=1}^{|V|} P'(j)\log(P'(j))$$

La información mutua (MI) entre U y V se calcula como:

$$\mathrm{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

Información Mutua Normalizada

La información mutua normalizada se define como:

$$\mathrm{NMI}(U, V) = \frac{\mathrm{MI}(U, V)}{\mathrm{mean}(H(U), H(V))}$$

Características:

- Las asignaciones de etiquetas aleatorias (uniformes) tienen un puntaje cercano a 0.0.
- Límite superior de 1: los valores cercanos a cero indican dos asignaciones de etiquetas que son en gran medida independientes, mientras que los valores cercanos a uno indican un acuerdo significativo.

Ejemplo 8

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.cluster import MeanShift
from sklearn.cluster import DBSCAN
from sklearn import mixture
from sklearn.cluster import AgglomerativeClustering
from sklearn import metrics

Xt=np.load('Xtodos.npy')
Yt=np.load('Label.npy')[:,0]

#Metodo de agrupamiento
Model=KMeans(n_clusters=9)
# Model=MeanShift(bandwidth=0.02)
# Model=DBSCAN(eps=0.02, min_samples=5)
# Model=mixture.GaussianMixture(n_components=9)
# Model=AgglomerativeClustering(n_clusters=9)

Yp=Model.fit_predict(Xt)

ARI=metrics.adjusted_rand_score(Yt,Yp)
NMI=metrics.normalized_mutual_info_score(Yt,Yp)

print("ARI= ", '{:.2f}'.format(ARI))
print("NMI= ", '{:.2f}'.format(NMI))
```

Ejemplo 8

	K-means	Mean Shift	DBSCAN	Gaussian Mixture	HAC
ARI	0.85	0.56	0.44	0.75	0.85
NMI	0.94	0.65	0.74	0.89	0.94

Métricas no supervisadas

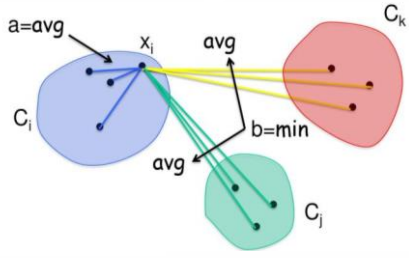
Coeficiente de silueta

Si no se conocen las etiquetas, la evaluación debe realizarse utilizando el modelo mismo. El coeficiente de silueta es un ejemplo de dicha evaluación, donde una puntuación más alta del coeficiente de silueta se relaciona con un modelo con grupos mejor definidos.

Coeficiente de silueta

El coeficiente de silueta se define para cada muestra y se compone de dos puntajes:

- **a**: La distancia media entre un dato y todos los demás en la misma clase.
- **b**: La distancia media entre un dato y todos los demás en el siguiente grupo más cercano.



Coeficiente de silueta

El coeficiente de silueta s para un dato se calcula como:

$$s = \frac{b - a}{max(a, b)}$$

Características:

- La puntuación está limitada entre -1 para la agrupación incorrecta y +1 para la agrupación altamente densa. Los puntajes alrededor de cero indican grupos superpuestos.
- La puntuación es mayor cuando los grupos son densos y están bien separados.

El índice de Calinski-Harabasz

El índice de Calinski-Harabasz es la relación entre la suma de la dispersión entre grupos y la dispersión entre grupos para todos los grupos (donde la dispersión se define como la suma de las distancias al cuadrado). Una puntuación más alta de Calinski-Harabasz se refiere a un modelo con grupos mejor definidos.

El índice de Calinski-Harabasz

Para un conjunto de datos de tamaño n_E que se ha agrupado en k grupos, el índice de Calinski-Harabasz se define como la relación entre la dispersión media entre grupos y la dispersión dentro del grupo:

$$s = \frac{\mathrm{tr}(B_k)}{\mathrm{tr}(W_k)} \times \frac{n_E - k}{k - 1}$$

donde $\mathrm{tr}(B_k)$ es la traza de la matriz de dispersión entre grupos y $\mathrm{tr}(W_k)$ es la traza de la matriz de dispersión dentro del grupo.

El índice de Calinski-Harabasz

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T$$

Donde $C_q$es el conjunto de datos en el grupo q, c_q es el centro del grupo q, c_E es el centro de E (todos los datos), y n_q es el numero de datos en el grupo q.

Característica: El índice es mayor cuando los grupos son densos y están bien separados.

Ejemplo 9

Se cuenta con una base de datos de análisis de aceite en descansos de motores y bombas.

Equipo	Desc Componente	Tag	Correlativo Muestra	Fecha	Fierro	Cromo	Aluminio	Cobre	Plomo	Nickel	Plata	Estano	Titanio	Vanadio	Cadmio	Manganeso
BBA 163 63056-57	DESCANSO BBA 163 LADO ACOPLE	MCM-AR09-63056	305780	02-05-2019	3,7	0,1	0,1	1,2	0,6	0,1	0,1	0,5	0,1	0,1	0,1	0,1
BBA 163 63056-57	DESCANSO BBA 163 LADO LIBRE	MCM-AR09-63057	305781	02-05-2019	17,9	0,1	0,1	0,1	0,1	0,1	0,1	0,7	0,1	0,1	0,1	0,1
BBA 172 63060-61	DESCANSO BBA 172 LADO ACOPLE	MCM-AR09-63060	305782	02-05-2019	1,3	0,1	0,1	1,7	1,0	0,1	0,1	0,7	0,1	0,1	0,1	0,2
BBA 172 63060-61	DESCANSO BBA 172 LADO LIBRE	MCM-AR09-63061	305783	02-05-2019	9,5	0,1	0,1	0,1	2,2	0,1	0,1	1,3	0,1	0,1	0,1	0,1
BBA 182 63066-67	DESCANSO BBA 182 LADO ACOPLE	MCM-AR09-63066	305784	02-05-2019	9,7	0,1	1,3	0,1	0,8	0,1	0,1	0,7	0,1	0,1	0,1	0,1
BBA 182 63066-67	DESCANSO BBA 182 LADO LIBRE	MCM-AR09-63067	305785	02-05-2019	270,0	3,3	0,7	9,5	0,1	0,5	0,1	1,2	0,1	0,1	0,1	2,4
MTR 163 63082-83	DESCANSO MTR 163 LADO ACOPLE	MCM-AR09-63082	305786	02-05-2019	1,1	0,1	0,1	8,3	0,1	0,1	0,1	5,0	0,1	0,1	0,1	0,1
MTR 163 63082-83	DESCANSO MTR 163 LADO LIBRE	MCM-AR09-63083	305787	02-05-2019	0,4	0,1	0,1	0,7	0,1	0,1	0,1	2,5	0,1	0,1	0,1	0,1
MTR 172 63086-87	DESCANSO MTR 172 LADO ACOPLE	MCM-AR09-63086	305788	02-05-2019	3,4	0,1	0,1	5,7	0,1	0,1	0,1	3,1	0,1	0,1	0,1	0,3
MTR 172 63086-87	DESCANSO MTR 172 LADO LIBRE	MCM-AR09-63087	305789	02-05-2019	2,3	0,1	0,1	8,3	0,1	0,1	0,1	2,3	0,1	0,1	0,1	0,2
MTR 182 63092-93	DESCANSO MTR 182 LADO ACOPLE	MCM-AR09-63092	305790	02-05-2019	0,1	0,1	0,1	0,1	0,1	0,1	0,1	4,9	0,1	0,1	0,1	0,1
MTR 182 63092-93	DESCANSO MTR 182 LADO LIBRE	MCM-AR09-63093	305791	02-05-2019	2,4	0,1	0,1	1,1	0,1	0,1	0,1	5,8	0,1	0,1	0,1	0,2
BBA 161 63052-53	DESCANSO BBA 161 LADO ACOPLE	MCM-AR09-63052	307323	17-05-2019	51,8	0,1	0,1	2,2	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,3
BBA 161 63052-53	DESCANSO BBA 161 LADO LIBRE	MCM-AR09-63053	307324	17-05-2019	4,7	0,1	0,1	2,7	0,8	0,1	0,1	0,1	0,1	0,1	0,1	0,1
BBA 171 63058-59	DESCANSO BBA 171 LADO ACOPLE	MCM-AR09-63058	307325	17-05-2019	6,3	0,1	0,1	34,9	10,9	0,1	0,1	0,3	0,1	0,1	0,1	0,1
BBA 171 63058-59	DESCANSO BBA 171 LADO LIBRE	MCM-AR09-63059	307326	17-05-2019	68,4	0,1	0,1	17,9	2,3	0,1	0,1	0,1	0,1	0,1	0,1	0,1
BBA 181 63064-65	DESCANSO BBA 181 LADO ACOPLE	MCM-AR09-63064	307327	17-05-2019	8,1	0,1	0,1	2,4	1,2	0,1	0,1	0,1	0,1	0,1	0,1	0,1
BBA 181 63064-65	DESCANSO BBA 181 LADO LIBRE	MCM-AR09-63065	307328	17-05-2019	44,3	0,2	0,1	6,7	1,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
MTR 161 63078-79	DESCANSO MTR 161 LADO ACOPLE	MCM-AR09-63078	307329	17-05-2019	0,4	0,1	0,1	8,0	0,1	0,1	0,1	10,2	0,1	0,1	0,5	0,1
MTR 161 63078-79	DESCANSO MTR 161 LADO LIBRE	MCM-AR09-63079	307330	17-05-2019	2,4	0,1	0,1	7,4	0,1	0,1	0,1	8,2	0,1	0,1	0,1	0,1
MTR 171 63084-85	DESCANSO MTR 171 LADO ACOPLE	MCM-AR09-63084	307331	17-05-2019	0,1	0,1	0,1	8,9	0,1	0,1	0,1	5,8	0,1	0,1	0,1	0,1
MTR 171 63084-85	DESCANSO MTR 171 LADO LIBRE	MCM-AR09-63085	307332	17-05-2019	0,1	0,1	0,1	0,7	0,1	0,1	0,1	2,5	0,1	0,1	0,1	0,1
MTR 181 63090-91	DESCANSO MTR 181 LADO ACOPLE	MCM-AR09-63090	307333	17-05-2019	2,7	0,1	0,1	0,9	0,1	0,1	0,1	4,0	0,1	0,1	0,1	0,1
MTR 181 63090-91	DESCANSO MTR 181 LADO LIBRE	MCM-AR09-63091	307334	17-05-2019	0,1	0,1	0,1	1,1	0,1	0,1	0,1	2,6	0,1	0,1	0,1	0,1
BBA 161 63052-53	DESCANSO BBA 161 LADO ACOPLE	MCM-AR09-63052	297625	10-01-2019	45,4	0,1	0,1	0,5	0,1	0,1	0,1	0,5	0,1	0,1	0,1	0,1

Ejemplo 9

Los datos no están clasificados. Por lo tanto se busca realizar un análisis exploratorio mediante algoritmos de agrupamiento que permita visualizar y agrupar datos con características comunes que podrían estar asociados a la condición de salud del descanso.

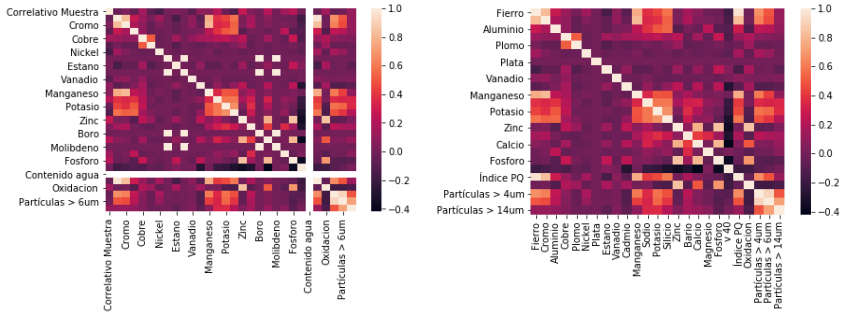
Ejemplo 9

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.feature_selection import VarianceThreshold
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.cluster import MeanShift
from sklearn.cluster import DBSCAN
from sklearn import mixture
from sklearn.cluster import AgglomerativeClustering
from mpl_toolkits.mplot3d import Axes3D
from itertools import cycle
from sklearn import metrics

data = pd.read_excel('datos_aceite.xlsx')
print(data.columns)

#remover variables con alta correlación
corr = data.corr()
ax=sns.heatmap(corr)
plt.show()

print(corr.unstack().sort_values(ascending=False).head(30))
```


Ejemplo 9

```
columnas = ['Fierro', 'Cromo', 'Aluminio', 'Cobre', 'Plomo', 'Nickel',
    'Estano', 'Vanadio', 'Cadmio', 'Manganeso', 'Sodio',
    'Potasio', 'Silicio', 'Zinc', 'Bario', 'Boro', 'Calcio',
    'Magnesio', 'Fosforo', 'v 40', 'Contenido agua', 'Partículas > 4um',
    'Partículas > 14um']

Xs = data[columnas]
corr = Xs.corr()
ax=sns.heatmap(corr)
plt.show()

print(corr.unstack().sort_values(ascending=False).head(30))

#normalizar max min
scaler = MinMaxScaler()
scaler.fit(Xs)
Xs=scaler.transform(Xs)
Xs.var()

#filtrar datos con baja varianza
sel = VarianceThreshold(0.01)
Xf=sel.fit_transform(Xs)
```

Ejemplo 9



Ejemplo 9

```
#Método de reducción
pca = PCA(n_components=10)
nombre_red='PCA'
Xt = pca.fit_transform(Xf)
varianza=pca.explained_variance_ratio_.cumsum()

#Metodo de agrupamiento
Model=KMeans(n_clusters=2)
nombre_mod='Kmeans'

# Model=MeanShift(bandwidth=0.7)
# nombre_mod='Mean Shift'

# Model=DBSCAN(eps=1.6, min_samples=5)
# nombre_mod='DBSCAN'

# Model=mixture.GaussianMixture(n_components=4)
# nombre_mod='Gaussian Mixture'

# Model=AgglomerativeClustering(n_clusters=2)
# nombre_mod='HAC'

Yp=Model.fit_predict(Xt)
silueta=metrics.silhouette_score(Xt,Yp)
calinski=metrics.calinski_harabasz_score(Xt,Yp)
```

Ejemplo 9

```
#Método de reducción
pca = PCA(n_components=10)
nombre_red='PCA'
Xt = pca.fit_transform(Xf)
varianza=pca.explained_variance_ratio_.cumsum()

#Metodo de agrupamiento
Model=KMeans(n_clusters=2)
nombre_mod='Kmeans'

# Model=MeanShift(bandwidth=0.7)
# nombre_mod='Mean Shift'

# Model=DBSCAN(eps=1.6, min_samples=5)
# nombre_mod='DBSCAN'

# Model=mixture.GaussianMixture(n_components=4)
# nombre_mod='Gaussian Mixture'

# Model=AgglomerativeClustering(n_clusters=2)
# nombre_mod='HAC'

Yp=Model.fit_predict(Xt)
silueta=metrics.silhouette_score(Xt,Yp)
calinski=metrics.calinski_harabasz_score(Xt,Yp)

print('Metodo: ', nombre_mod, ', Silueta: ', '{:.2f}'.format(silueta))
print('Metodo: ', nombre_mod, ', Calinski: ', '{:.2f}'.format(calinski))
```

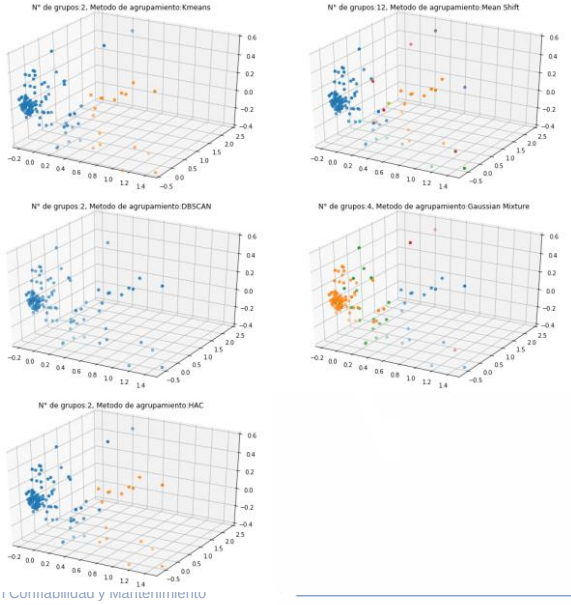

Ejemplo 9

	K-means	Mean Shift	DBSCAN	Gaussian Mixture	HAC
Silueta	0.51	0.48	0.69	0.32	0.51
Calinski	47.6	22.8	14.91	30.47	46.75

Ejemplo 9

```
labels_unique = np.unique(Yp)
n_clusters_ = len(labels_unique)

fig = plt.figure(figsize=(9,6))
ax = fig.add_subplot(111, projection='3d')
colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
for k, col in zip(range(n_clusters_), colors):
    my_members = Yp == k
    ax.scatter(Xt[my_members, 0], Xt[my_members, 1], Xt[my_members, 2], col + '.')
    ax.set_xlim(-0.25,1.5)
    ax.set_ylim(-0.8,2.5)
    ax.set_zlim(-0.4,0.6)

plt.title('N° de grupos:' + str(n_clusters_) + ', Metodo de agrupamiento:' + nombre_mod)
plt.show()
```


Ejemplo 9



Tarea 3

Consideramos nuevamente los datos de la tarea 1, pero esta vez se incorporó un tercer parámetro.

Los parámetros asociados a los cuatro casos se entregan en los siguientes archivos:

- Pa0_3.npy: parámetros del caso sin daño
- Pa1_3.npy: parámetros del caso Grieta 1
- Pa2_3.npy: parámetros del caso Grieta 4
- Pa3_3.npy: parámetros del caso Grieta 6

- Evalúe los distintos métodos de agrupamiento vistos en clases.