min-max los valores numéricos de un dataset.

Vamos a mostrar un ejemplo simple, pues en el módulo de programación de inteligencia artificial profundizarán en estos temas o pueden hacerlo buscando información de cómo usar estas técnicas en python.

```
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()


data_scaled = scaler.fit_transform(data)
```

Reducción de datos

Hasta aquí hemos visto las tareas necesarias para integrar los datos desde un conjunto de fuentes distintas y como tratar con muchos de los problemas que se presentan en ellos antes que se puedan usar para el entrenamiento de los algoritmos de aprendizaje automático. Sin embargo hay otro grupo de problemas que, aunque no impactan de manera directa los valores individuales del conjunto de datos, si lo hacen a nivel de las prestaciones y complejidad de los modelos.

En los últimos 20 años, la dimensionalidad de los conjuntos de datos usados en aprendizaje automático ha aumentado drásticamente. (Dimensionalidad es la cantidad de filas por columnas del conjunto) En los años 80 la dimensionalidad rondaba el número 100 y en los 2000 ya superaba los 1500.

Cada día es mucho más común encontrar conjuntos de datos con millones de casos y miles de rasgos. Mientras más grandes son estos conjuntos más lento y complejo se hace el entrenamiento de los modelos. Esto puede conducir a sobreajuste (overfitting) y los modelos tienden a ser menos interpretables.

Para ayudar a resolver estos problemas se usan técnicas de reducción de datos que se agrupan de la forma siguiente:

- Técnicas de reducción de la dimensionalidad.

- Técnicas de reducción del número de instancias o casos (numerosity reduction).
- Técnicas de reducción de la cardinalidad.

Las técnicas de reducción de dimensionalidad se pueden agrupar en:

- Selección de rasgos.
- Construcción/extracción de rasgos.

Las técnicas de reducción del número de instancias o casos se pueden agrupar en:

- Técnicas paramétricas. (no las veremos)
- Técnicas no paramétricas.

Las técnicas de reducción del número de instancias o casos no paramétricas se pueden agrupar en:

- Muestreo de datos.
- Selección de instancias. (casos)

Mientras que las técnicas de reducción de la cardinalidad se pueden clasificar en:

- Compartimentación. (binning)
- Discretización.

Es importante que dejemos claro el concepto de “dimensión” de un conjunto de datos. Cuando se habla de la dimensión de un conjunto, se hace referencia a la cantidad de rasgos de un conjunto de datos, es decir a la cantidad de columnas de esta. Es necesario recordar que un conjunto de datos está compuesto por filas y columnas y que a las columnas se les llaman atributos, rasgos, variables y ahora dimensiones, por otro lado a las filas se les llama casos, instancias, etc.

Tema 2. Análisis exploratorio de datos y preprocesamiento

¿Por qué se les llama dimensiones a los atributos o rasgos de un conjunto de datos? La respuesta es sencilla. Todos los algoritmos de aprendizaje automáticos usan conceptos matemáticos para modelar las cosas de la vida real. En esta situación específica, cada una de las instancias o casos de un conjunto son vistas como vectores embebidos en un espacio n-dimensional. Cada uno de los rasgos se hace coincidir con una de estas dimensiones.

Curso de la dimensionalidad

Uno de los principales problemas con los que nos encontramos en aprendizaje automático, cuando el número de dimensiones de un conjunto de datos es grande es el llamado “curso de la dimensionalidad”. Hay una relación lineal entre el número de ejemplos o casos de entrenamiento y la dimensionalidad para obtener modelos de alta calidad. Debido a esto, este problema se le llama “maldición de la dimensionalidad”.

Técnicas de reducción de la dimensionalidad

Uno de los problemas representativos de la reducción de dimensionalidad es el conocido como “Selección de Rasgos”. Por eso, iniciaremos esta sección analizando las técnicas más comunes usadas para atacar este problema y luego veremos las correspondientes a la construcción/extracción de rasgos.

Selección de rasgos

Se le denomina selección de rasgos al proceso por medio del cual se elige un subconjunto óptimo de rasgos, de entre todos los que componen el conjunto, basado en algún criterio.

La idea detrás de la solución de este problema es encontrar el subconjunto óptimo de rasgos, es decir los rasgos más importantes y descargar aquellos que son irrelevantes o redundantes.

Las razones para ejecutar un proceso de selección de rasgos son:

- Eliminar datos irrelevantes.
- Mejorar la calidad de la predicción en los modelos obtenidos.
- Reducir los costos computacionales del proceso del aprendizaje.
- Reducir la complejidad de la descripción de los modelos aprendidos.

Primero que todo, el problema de selección de rasgos, puede considerarse como un problema de búsqueda, donde cada estado del espacio de búsqueda es un subconjunto de rasgos. Para entender este problema, supongamos que tenemos 3 rasgos o dimensiones. Si sabemos que cada rasgo, puede formar parte o no de cada una de las combinaciones, entonces podemos formar 2 elevado a m subconjuntos, si m es el número de rasgos originales. Entonces probaríamos cada uno de los subconjuntos o combinaciones y determinamos el que genere el mejor modelo.

Rara vez los conjuntos de datos de la vida real tienen tan pocos rasgos y lo normal en problemas de ciencia o la salud es que tengan muchos. En tareas como la clasificación de textos donde se usan técnicas de procesamiento de lenguaje natural las dimensiones se cuentan en miles. Supongamos que tenemos solamente 32 rasgos, pues la cantidad de combinaciones que habría que explorar para encontrar el subconjunto optimo serían 4294967296. Esto demuestra la importancia de contar con algoritmos de selección sofisticados. (Ver la Figura 10).

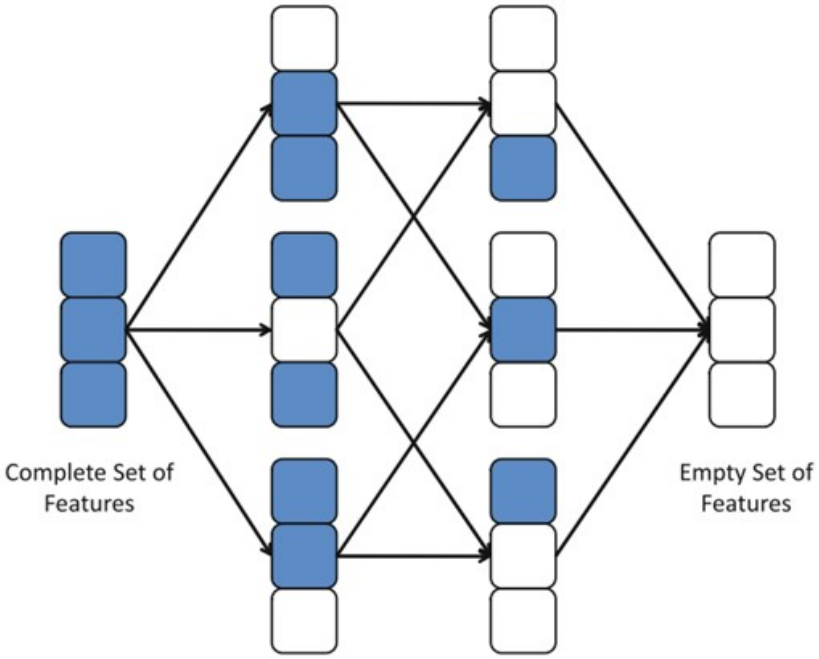


Figura 10. Espacio de búsqueda para selección de rasgos (m=3).

Los algoritmos de selección de rasgos, con el fin de encontrar el subconjunto óptimo, pueden comenzar a explorar todas las posibles combinaciones de rasgos de cuatro formas a la que en la literatura se les conoce como direcciones de búsqueda.

La primera, sería comenzar con un subconjunto vacío e ir añadiendo rasgos de manera secuencial e ir evaluándolo. A esta se le llama Generación Secuencial hacia adelante, el conjunto va creciendo hasta que alcanza el conjunto de datos originales. El criterio de parada puede ser un determinado umbral para rasgos relevantes o simplemente todos los posibles subconjuntos en una búsqueda por fuerza bruta.

La segunda, se le llama Generación Secuencial hacia atrás, y comienza con el subconjunto completo de rasgos e iterativamente va eliminando el peor cada vez. Al final quedará un solo rasgo, considerado el más importante o informativo. Como en el caso anterior, puede definirse un criterio de parada.

La tercera forma, es llamada Generación Bidireccional, esta comienza en ambas

direcciones simultáneamente, ejecutando las dos anteriores de forma concurrente. Este tipo de búsqueda terminará en cualquiera de dos casos: 1) Cuando una de las búsquedas encuentra el mejor subconjunto m antes de alcanzar el medio exacto. 2) cuando las búsquedas alcanzan el medio del espacio de búsqueda.

La cuarta y última forma de búsqueda es la llamada Generación aleatoria, esta comienza la búsqueda en una combinación aleatoria.

Seleccionar una dirección de búsqueda es solo una de las decisiones; pero no la única. Si necesitamos escoger un subconjunto óptimo de rasgos, además de escoger la dirección en la que recorreremos las combinaciones o espacio de búsqueda, también debemos elegir cómo las recorremos y a esto se le llama estrategia de búsqueda. A continuación esbozamos algunas:

- Búsqueda Exhaustiva: En esta estrategia se recorren todas y cada una de las combinaciones posibles para encontrar el subconjunto óptimo. Este tipo de estrategia no es práctica en dataset con un número de rasgos muy alto.
- Búsqueda Heurística: Este tipo de búsqueda hace uso de heurísticas con el fin de evitar recorrer de forma exhaustiva todas las combinaciones de rasgos. Este problema se ha atacado con muchas metaheurísticas tales como Optimización por colonias de hormigas (ACO). Este se considera un problema de Optimización combinatorial.
- Búsqueda no determinista: Este tipo de búsqueda aflora de la combinación complementaria de las dos anteriores.

Hasta aquí hemos visto la manera en que se recorren las posibles combinaciones de los m rasgos que componen nuestra conjunto de datos; pero aún no sabemos cómo escogemos el mejor subconjunto, a esto se le llama criterio de selección y veremos algunos a continuación.

Los criterios de selección se pueden clasificar de la siguiente manera:

- Criterios basados en medidas de información.
- Criterios basados en medidas de distancia.
- Criterios basados en medidas de dependencia.
- Criterios basados en medidas de consistencia.
- Criterios basados en medidas de calidad de la clasificación (Accuracy).

Criterios basados en medidas de información

Este tipo de criterio se basa en alguna medida de ganancia de información o medida de reducción de la incertidumbre, por ejemplo la entropía de Shannon.

$$-\sum_i P(c_i)\log_2 P(c_i).$$

En este tipo de criterio, un atributo A es escogido basado en una función de ganancia de información IG, si IG(A)>IG(B).

La forma exacta que toma esta medida depende del algoritmo.

Criterios basados en medidas de distancia

También conocidas como distancias de separabilidad, discriminación o divergencia. Este tipo de criterio es similar al anterior, aunque en este las funciones son medidas de distancia.

Las medidas más usadas en este tipo de criterio están sumariadas en la siguiente Tabla.

Nombre	Forma matemática
Distancia euclidiana	$D_e = \left[\sum_{i=1}^{m}(x_i - y_i)^2\right]^{\frac{1}{2}}$
Distancia City-Block	$D_{cb} = \sum_{i=1}^{m}\lvert x_i - y_i\rvert$
Distancia de Cebyshev	$D_{ch} = max_i\lvert x_i - y_i\rvert$
Distancia de Minkowski de orden m	$D_m = \left[\sum_{i=1}^{m}(x_i - y_i)^m\right]^{\frac{1}{m}}$
Distancia cuadrática Q	$D_q = \sum_{i=1}^{m}\sum_{j=1}^{m}(x_i - y_i)\, Q_{ij}(x_j - y_j)$
Distancia Canberra	$D_{ca} = \sum_{i=1}^{m}\frac{\lvert x_i - y_i\rvert}{x_i + y_i}$
Separación angular	$D_{as} = \frac{\sum_{i=1}^{m} x_i \cdot y_i}{[\sum_{i=1}^{m} x^2 \sum_{i=1}^{m} y^2]^{\frac{1}{2}}}$

Tabla 2. Criterios basados en medidas de distancia. Fuente: elaboración propia.

Criterios basados en medidas de dependencia

A este tipo de medida se les conoce también como medidas de correlación o asociación. Su objetivo es determinar cuan correlacionadas o asociadas están dos variables, de forma tal que conociendo el valor de una podamos derivar el valor de la otra.

El procedimiento común en selección de rasgos es establecer una correlación entre cada uno de los rasgos y la variable de decisión tal que si R(a) es la correlación entre el atributo a y la clase y R(b) es la correlación del atributo b y la clase, entonces escogemos al atributo a, si: R(a)>R(b).

Dicho en otras palabras, se escogen los rasgos con mayor correlación. Si cualquiera de los atributos del dataset es estadísticamente independiente de la clase, su eliminación no tendrá efecto alguno en la separabilidad entre las clases en un problema de clasificación.

Sin embargo, este tipo de correlación no puede manejar correlaciones no lineales entre atributos, por tanto hay que apelar a otras medidas de correlación, por ejemplo la medida de dependencia de Bhattacharyya.

Una de las medidas más usadas es el coeficiente de correlación de Pearson:

$$\rho(x,y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\left[\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2\right]^{\frac{1}{2}}}$$

Criterios basados en medidas de consistencia

A diferencia de las medidas anteriores, esta trata de encontrar el conjunto mínimo de rasgos que separaría las clases de la misma forma que lo hace el conjunto de rasgos completo. La idea de estas medidas es la de encontrar un subconjunto de rasgos que mantenga la misma consistencia en los datos que la consistencia arrojada teniendo en cuenta el conjunto de rasgos originales.

Aquí la idea de inconsistencia es la de dos casos con rasgos con los mismos valores de los rasgos; pero con clases diferentes asignadas.

Criterios basados en medidas de calidad de la clasificación

Este tipo de medida descansa en las formas de evaluación de la calidad de clasificación cuando se usa un clasificador para determinar la optimalidad de un subconjunto de rasgos. La idea detrás de este tipo de medida es someter a entrenamiento un clasificador con los subconjuntos de rasgos que pretendemos evaluar y escoger aquel que mejor calidad de clasificación nos arroje.

Tema 2. Análisis exploratorio de datos y preprocesamiento

Hasta ahora no hemos hablado de las métricas necesarias para la evaluación de los modelos de aprendizaje supervisado y por tanto solo sumariaremos estas medidas en una tabla y luego en la sección correspondiente a los algoritmos de clasificación daremos más detalles sobre ellas.

	Mathematical form
Accuracy	$\frac{tp+fp}{tp+tn+fp+fn}$
Error rate	1− Accuracy
Chi-squared	$\frac{n(fp \times fn - tp \times tn)^2}{(tp+fp)(tp+fn)(fp+tn)(tn+fn)}$
Information gain	$e(tp+fn, fp+tn) - \frac{(tp+fp)e(tp,fp)+(tn+fn)e(fn,tn)}{tp+fp+tn+fn}$ where $e(x,y) = -\frac{x}{x+y}\log_2\frac{x}{x+y} - \frac{y}{x+y}\log_2\frac{y}{x+y}$
Odds ratio	$\frac{tpr}{1-tpr} \Big/ \frac{fpr}{1-fpr} = \frac{tp \times tn}{fp \times fn}$
Probability ratio	$\frac{tpr}{fpr}$

Tabla 3. Ejemplo. Fuente: elaboración propia.

Donde, tp es true positivo, fp es falso positivo, tn es true negativo y fn significa falso negativo. En la sección de evaluación de los algoritmos de clasificación se verán los detalles.

Los métodos de selección de rasgos se pueden agrupar, básicamente, en las siguientes categorías:

- Filtros.
- Envoltorios o Wrappers.
- Embebidos.
- Híbridos.

Métodos de selección de rasgos del tipo filtros

Los métodos que se clasifican como “filtros” operan antes de someter los datos al entrenamiento de los algoritmos de aprendizaje automático. Este nombre lo reciben a partir de que filtran los rasgos no deseados antes del proceso de aprendizaje.

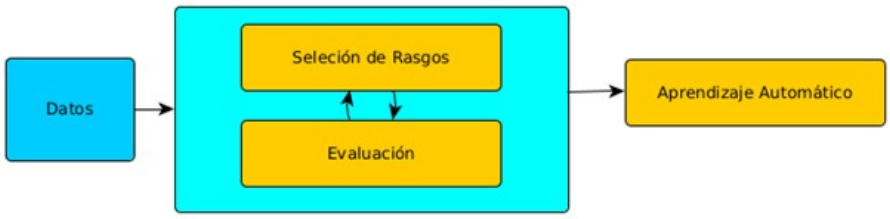


Figura 11. Operación de los métodos conocidos como filtros.

Es importante tener en cuenta que no todos los métodos de filtrado pueden usarse con todos los métodos de aprendizaje, por eso en la lista siguiente se clasifican según el método en los que se puede utilizar.

Hay muchos métodos de filtrado, en la tabla siguiente se contemplan algunos.

Nombre	Clase de Filtro	Tarea Aplicable
Information gain	Univariada, información	clasificación
Gain ratio	Univariada, información	clasificación
Correlation-based feature selection (CFS)	Multivariada, dependencia	Clasificación, regresión
Fast correlation-based filter (FCBF)	Multivariada,información	clasificación
Multi-Cluster Feature Selection (MCFS)	Multivariada, Similaridad	Clustering

Tabla 4. Métodos de filtrado. Fuente: elaboración propia.

Estos métodos pueden usar medidas univariadas para elegir cada uno de los rasgos o multivariadas.

Veamos un ejemplo práctico de selección de rasgos usando el módulo Scikit-Lear de phyton.

Este módulo contiene un grupo de clases que permite de forma muy sencilla e intuitiva implementar selección de rasgos en sus etapas de preprocesamiento de

datos.

Este módulo tiene un método muy simple de usar, que permite eliminar rasgos que tienen varianza cero o muy cerca de ella o por debajo de un determinado umbral. Veamos:

```
import pandas as pd
from sklearn import datasets
from sklearn.feature_selection import VarianceThreshold
from sklearn.preprocessing import StandardScaler
#*****************************************************************
iris = datasets.load_iris()
#*****************************************************************
v_thr = VarianceThreshold(threshold=0.6)
data= v_thr.fit_transform(iris.data)
print(v_thr.get_support())
data.shape
```

```
[ True False  True False]
(150, 2)
```

Figura 12. Ejemplo. Fuente: elaboración propia.

Como se puede ver en este ejemplo, la clase que permite este tipo de filtrado es VarianceThreshold, y toma como parámetro un valor que sirve como umbral para el filtrado. El valor por defecto es cero.

También incluye una clase llamada SelectKBest, que permite usar métodos de filtrado basados en medidas univariadas, como chi cuadrado, veamos un ejemplo y luego damos algunos detalles.

```
import pandas as pd
from sklearn import datasets
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
#*****************************************************************
iris = datasets.load_iris()
#*****************************************************************
iris_new = SelectKBest(chi2, k=2).fit_transform(iris.data, iris.target)
iris_new.shape

(150, 2)
```

Figura 13. Ejemplo. Fuente: elaboración propia.

En este ejemplo se escogen los dos mejores rasgos según la medida chi cuadrado.

Las medidas que se pueden usar con esta clase son:

- f_classif: basada en ANOVA F-value y usada en tareas de clasificación.
- mutual_info_classif: Usa Información mutua, variable objetivo discreta.
- Chi2: Chi-cuadrado, estadístico de rasgos no negativos, clasificación .
- f_regression: F-value entre label/feature para tareas de regresion.
- mutual_info_regression: Información Mutua para variable objetivo continua.
- SelectPercentile: Selecciona los rasgos basado en los percentiles de los scores más altos.
- SelectFpr: Selecciona los rasgos basado en la tasa de falsos positivos.
- SelectFdr: Selecciona los rasgos basado en false discovery rate.
- SelectFwe: Selecciona los rasgos basado en family-wise error rate.
- GenericUnivariateSelect: Selecciona los rasgos de modo configurable.
- Para el resto de los detalles del uso de los métodos de selección de rasgos en el

módulo Scikit-Learn, ver su documentación.

Métodos de selección de rasgos del tipo Wrappers

Este tipo de método integra dentro del proceso de selección de rasgos un algoritmo de aprendizaje automático como una caja negra que le permite evaluar la calidad de los subconjuntos de rasgos. Ahora bien, el problema con este tipo de método es que puede llevar a sobreajuste (overfitting) o sesgos y es más lento que los métodos de filtrado.

Con vistas a evitar el sobreajuste, dentro de estos métodos hay que implementar también algún método de validación estadística (como validación cruzada que aún no hemos tocado) lo que los hace más complejos que los anteriores.

La idea en el uso de este tipo de método es escoger el subconjunto de rasgos que mejor calidad de clasificación arroje. Estos métodos requieren de dos etapas de trabajo, una donde se selecciona el subconjunto de rasgos optimo o al menos uno suboptimo y la otra en la cual se entrenan los métodos de aprendizaje con el conjunto de rasgos obtenidos.

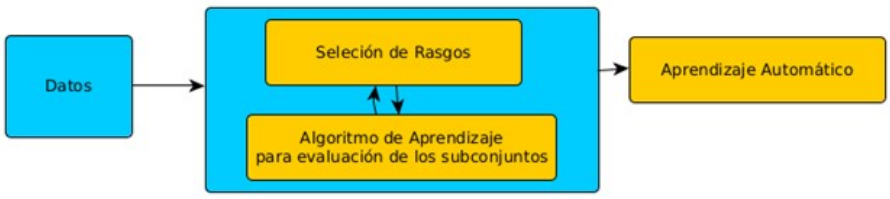


Figura 14. Operación de los métodos conocidos como Wrappers.

Métodos de selección de rasgos del tipo embebidos

Este enfoque es parecido a los métodos wrapper en que los rasgos son seleccionados específicamente para un algoritmo de aprendizaje. Sin embargo en este tipo de métodos los rasgos son seleccionados el mismo proceso de aprendizaje.

Estos métodos integran el proceso de selección de rasgos en el proceso de entrenamiento y eso los hace más eficientes en algunos aspectos al no tener que dividir los datos en entrenamiento y validación. Además, pueden obtener soluciones más rápido al no tener que reentrenar para cada subconjunto de rasgos, como los métodos anteriores.

Este tipo de método no es nuevo y algoritmos como C4.5 tienen métodos de selección de rasgos incorporados.

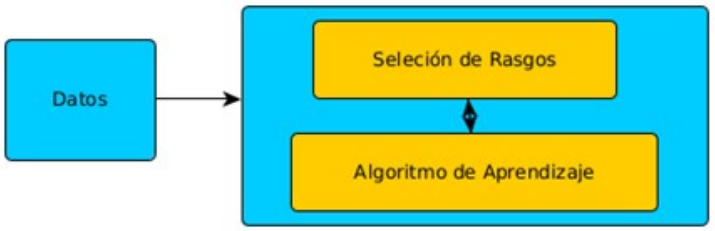


Figura 14. Operación de los métodos conocidos como embebidos.

Hasta aquí hemos hecho un breve recorrido por la clasificación más antigua y conocida del problema de Selección de rasgos; pero lo cierto es que la literatura es abrumadoramente grande y la cantidad de métodos y técnicas se extienden constantemente. Debido a esto es imposible que en este recorrido pudiéramos adentrarnos todo cuento quisiéramos y dejamos de su parte la tarea de explorar los métodos más actuales.

Análisis de componentes principales (PCA)

Quizás el método de reducción de la dimensionalidad más conocido es el de análisis de componentes principales o PCA como se le conoce por sus siglas en ingles. Esto se debe en gran medida a que es uno de los métodos más antiguos, introducido en los trabajos de Pearson en 1901 y Hotelling en 1933.

Si vemos nuestro conjunto de datos como un grupo de vectores representados en un

espacio n-dimencional, entonces la idea de PCA es encontrar un espacio p-dimensional donde p<n y que se puedan representar la mayoría de los vectores del conjunto de datos perdiendo la menor cantidad de ellos y preservando su estructura.

Desde el punto de vista práctico el método PCA, toma un conjunto de datos de x observaciones y n variables y nos retorna un conjunto de x observaciones y p variables donde p<n. A las nuevas variables se les conoce como componentes principales y son una combinación lineal de las variables originales.

Es importante tener en cuenta que para la utilización de PCA los valores del dataset deben ser numéricos y antes de aplicarlo hay que normalizarlos.

Para poder captar la perspectiva práctica del uso de este método, veamos cómo se puede hacer con el módulo de aprendizaje automático Scikit-Learn.

Para el ejemplo que presentamos usamos el conjunto llamado Iris dataset. Este conjunto de datos contiene 150 ejemplares de flores de 3 categorías distintas, descritas por 4 variables numéricas. Las clases o categorías son: setosa, versicolor y virginica.

```
In [28]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.decomposition import PCA
         from mpl_toolkits.mplot3d import Axes3D
         from sklearn import datasets
         #*********************************************************************
         # En esta sección se carga el dataset del conjunto de datasets internos
         # de Scikit_learn
         iris_data = datasets.load_iris(as_frame=True)
         iris_target = iris_data.target
         #*********************************************************************
         # En esta se crea un objeto a partir de la clase PCA y como argumento
         # toma la cantidad de componentes a los que queremos reducir nuestros
         # dataset. Si no sabemos o no queremos establecer un número de componentes
         # le pasamos como parámetro 0.95 que significa que queremos mantener el
         # el 95% de la varianza de los datos
         PCA_DR = PCA(n_components=3)
         # Aquí se calculan los componentes, es decir esta es la aplicación de PCA
         PCA_DR.fit(iris_data.data)
         # Aquí se transforma el dataset a partir de los calculos anteriores
         iris_Tranf= PCA_DR.transform(iris_data.data)
         #*********************************************************************
         fig = plt.figure(1, figsize=(8, 6))
         ax = Axes3D(fig, elev=-150, azim=110)
         ax.scatter(iris_Tranf[:, 0], iris_Tranf[:, 1], iris_Tranf[:, 2], c=iris_target,cmap=plt.cm.Set1, edgecolor='k', s=40
         ax.set_title("Tres componentes principales arrojados por PCA para el dataset iris")
         ax.set_xlabel("1er eigenvector")
         ax.w_xaxis.set_ticklabels([])
         ax.set_ylabel("2do eigenvector")
         ax.w_yaxis.set_ticklabels([])
         ax.set_zlabel("3er eigenvector")
         ax.w_zaxis.set_ticklabels([])
         plt.show()
```

Figura 14 Ejemplo de aplicación de un PCA al conjunto de datos Iris.

Al aplicar el PCA a este conjunto, como en el ejemplo dado en la fig 13. reducimos el dataset a 3 variables o componentes principales al pasar el argumento n_components=3 al constructor de la clase PCA. Si por el contrario pasáramos el argumento n_components=0.95 la implementación de este PCA nos retornaría los componentes principales que agrupan el 95% de la varianza de nuestros datos. Y en este caso, es decir con el dataset iris serían dos componentes. Esto significa que hemos reducido los datos originales con 4 rasgos a 3 en el caso que mostramos en el ejemplo con el fin de graficar los resultados. Ver Figura 15.

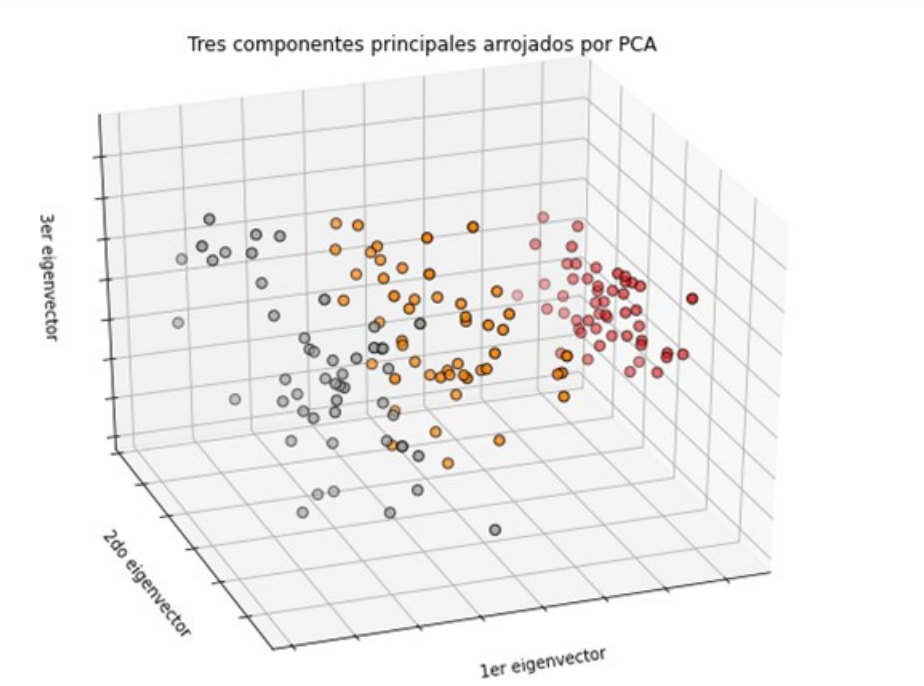


Figura 14. Gráfica de aplicación de un PCA al dataset Iris, para tres componentes.

Este módulo contiene varias implementaciones de PCA, para revisar más detalles de cómo aplicar este método, revisar: Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. De Aurélien Géron.

Análisis de factor

Al igual que el análisis de componentes principales o PCA, esta técnica de reducción de la dimensionalidad conduce a la deducción de un nuevo conjunto de variables que prácticamente describe la conducta de los datos originales. Aunque difiere en que este no trata de buscar transformaciones de los atributos dados, en vez, trata de descubrir factores ocultos en las variables originales.

La idea básica de esta técnica es intentar encontrar un grupo de factores ocultos, llamados factores latentes, de forma tal que los atributos originales puedan ser recuperados por medio de un conjunto de transformaciones lineales sobre estos factores.

La idea aquí es caracterizar la dependencia entre variables por medio de un pequeño número de factores.

¿Qué es un factor? Es una variable latente que describe la asociación entre el número de variables observadas. Cada factor explica una cierta cantidad de varianza en las variables observadas. Aquellos factores con la menor cantidad de varianza son eliminados.

¿Qué son los Loadings? Es una matriz que muestra las relaciones entre cada una de las variables y su factor subyacente. ¿Cómo se elige el número de factores? Generalmente un eigenvalue mayor que 1 es un buen criterio para seleccionar un factor. Otro método es a partir de la gráfica llamada scree, determinado donde la curva hace un codo.

No veremos aquí la forma matemática de llegar a los resultados; pero si veremos como en el caso de PCA, un ejemplo desarrollado usando scikit-lear.

Para iniciar con el ejemplo primero que todo debemos dejar claro las asunciones básicas que se requieren para la aplicación de esta técnica:

- Esta técnica asume que los datos son numéricos.

- Asume también que no tienen outliers.
- Que el tamaño de la muestra es mayor que la cantidad de factores.
- Y que no debe haber homocedasticidad entre las variables.
- Otro de los elementos a tener en cuenta antes de ejecutar el análisis de factores son las pruebas de adecuación de los datasets. Esto implica preguntarse si podemos encontrar factores en el dataset que tenemos a mano.

Básicamente hay dos test que se usan para esto:

- El test de Bartlett.
- El test de Kaiser-Meyer-Olkin.
- El test de Bartlett, es una prueba estadística que permite verificar si un conjunto de muestras tienen la misma varianza. Es una prueba paramétrica y la hipótesis nula h0 es que todas las muestras tienen la misma varianza.

Por tanto, si el valor de p-value retornado por la prueba es mayor que un valor de significación determinado, por ejemplo 0.05 o 5% entonces podemos decir que la hipótesis nula es válida. Sin embargo, a los efectos del uso de esta prueba en el análisis de factor, lo que nos dice es que si todas las varianzas son iguales entonces no podemos aplicar el procedimiento de análisis de factor a estos datos.

Por tanto, si el resultado de la aplicación del test de esfericidad de Bartlett arroja un p-value > 0.05, sus datos no son adecuados para aplicarle el análisis factorial.

El test de Kaiser-Meyer-Olkin, por su parte, considera que sus datos son adecuados para aplicar el análisis si el resultado es superior a 0.6. Los valores retornados por esta prueba estás entre 0 y 1.

Como ejemplo, mostraremos uno desarrollado con el paquete Scikit-learn y el otro con un paquete llamado factor_analyzer.

```python
import pandas as pd
import numpy as np
from sklearn import decomposition
from sklearn import datasets
from sklearn.decomposition import FactorAnalysis
from sklearn.preprocessing import StandardScaler
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
from factor_analyzer.factor_analyzer import calculate_kmo
#*********************************************
# Cargando el dataset
iris = datasets.load_iris(as_frame=True)
#*********************************************
#Estandarización de las variables
scaler = StandardScaler()
iris_scalado = scaler.fit_transform(iris.data)
#*********************************************
chi_square_value,p_value=calculate_bartlett_sphericity(iris_scalado)
kmo_all,kmo_model=calculate_kmo(iris_scalado)
if p_value<0.05 and kmo_model>0.5:
    fa=FactorAnalysis(n_components=2)
    fa.fit(iris_scalado)
    iris_Transformed = fa.transform(iris_scalado)
    fig = plt.figure(figsize=(20,10))
    fig.suptitle("Análisis de Factor con el módulo Scikit-Learn", fontsize=14)
    ax = fig.add_subplot(2,3,1)
    ax.scatter(iris_Transformed[:,0], iris_Transformed[:,1], c=iris.target)
else:
    print("Sus datos no son adecuados para este tipo de técnica")
    print("Pues p_value es igual a "+str(p_value))
    print("y el valor de kmo es "+str(kmo_model))
```

Figura 15. Análisis de factores con Scikit-Learn. Fuente: elaboración propia.

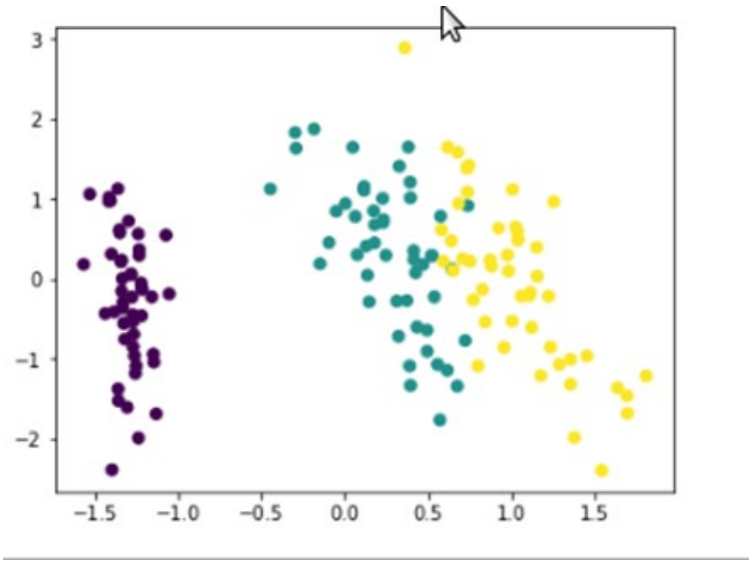


Figura 16. Resultado del Análisis de factores con Scikit-Learn. Fuente: elaboración propia.

En el ejemplo mostrado en la Fig. 15, hemos escogido representar un conjunto de datos con 4 rasgos por medio de uno que sola mente tiene 2 y contiene la mayoría

de la información del conjunto original.

No hablamos aquí de los tipos de rotación y su significado, pues a los efectos de la reducción de dimensionalidad es irrelevante.

Un método manual para escoger la cantidad de factores, es inspeccionar los eigenvalues y escoger los factores para los cuales estos son mayores que 1.

```
import pandas as pd
import numpy as np
from sklearn import datasets
from factor_analyzer import FactorAnalyzer
from sklearn.preprocessing import StandardScaler
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
from factor_analyzer.factor_analyzer import calculate_kmo
#*************************************************
# Cargando el dataset
iris = datasets.load_iris(as_frame=True)
#*************************************************
#Estandarización de las variables
scaler = StandardScaler()
iris_scalado = scaler.fit_transform(iris.data)
#*************************************************
chi_square_value,p_value=calculate_bartlett_sphericity(iris_scalado)
kmo_all,kmo_model=calculate_kmo(iris_scalado)
if p_value<0.05 and kmo_model>0.5:
    fa = FactorAnalyzer(rotation=None)
    fa.fit(iris_scalado)
    iris_Transformed = fa.transform(iris_scalado)
    fig = plt.figure(figsize=(20,10))
    fig.suptitle("Análisis de Factor con el módulo factor_analyzer", fontsize=14)
    ax = fig.add_subplot(2,3,1)
    ax.scatter(iris_Transformed[:,0], iris_Transformed[:,1], c=iris.target)
else:
    print("Sus datos no son adecuados para este tipo de técnica")
    print("Pues p_value es igual a "+str(p_value))
    print("y el valor de kmo es "+str(kmo_model))
```

Figura 17. Análisis de Factores con factor_analyzer. Fuente: elaboración propia.

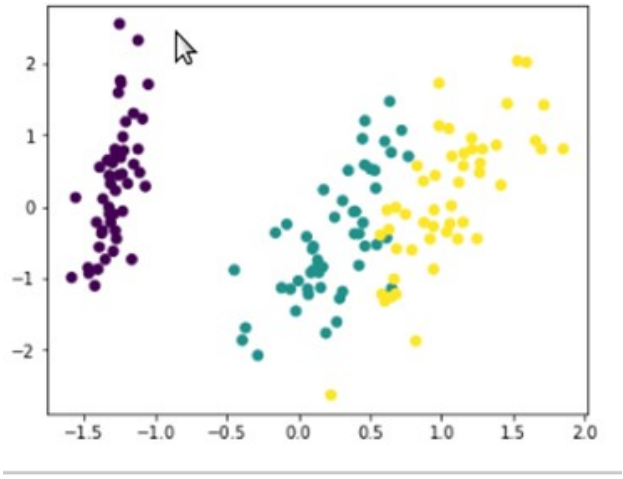


Figura 18. Resultado del Análisis de Factores con factor_analyzer. Fuente: elaboración propia.

Locally Linear Embedding. (LLE)

A falta de un buen término para traducir el nombre de este método de reducción de la dimensionalidad, hemos preferido no hacerlo y referirnos a él como LLE.

Este método, lo traemos como representante de un tipo de método de reducción de la dimensionalidad en los que, a diferencia de PCA, pueden lidiar con relaciones no lineales entre las variables de un conjunto de datos. Estos métodos se engloban bajo el nombre de Manifold Learning.

Un Manifold, y lo dejamos sin traducción, es un objeto o estructura matemática, tomado de la topología. No pretendemos dar definiciones matemáticamente correctas aquí, sino desarrollar la intuición detrás de estos métodos y especialmente LLE.

Intuitivamente podemos imaginar un manifold como un objeto o figura representada enR^nel cual es embebido en un espacio R^kdonde k>n.

La base del uso de estos métodos en reducción de la dimensionalidad es la llamada "Hipótesis de Manifold" la cual establece que los datos altamente dimensionales de la vida real radican en Manifolds de más baja dimensión embebidos en dichos espacios.

Esto nos lleva a plantear este problema, de manera muy informal, como la búsqueda de un Manifold de más baja dimensión que contiene la información de nuestros datos, en un espacio de dimensiones mayores en donde está representado.

Dejamos el estudio de la mecánica de trabajo del algoritmo para que sea estudiada a partir de otras fuentes y veremos de forma práctica como se usa a partir de un ejemplo escrito en Phyton.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import decomposition
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import LocallyLinearEmbedding
#*************************************************
# Cargando el dataset
n_points = 1000
X, color = datasets.make_swiss_roll(n_points, random_state=0)
#*************************************************
fig = plt.figure(figsize=(20,10))
fig.suptitle("LLE del swiss roll con 1000 puntos, 15 vecinos", fontsize=14)
ax = fig.add_subplot(231, projection='3d')
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=color, cmap=plt.cm.Spectral)
ax.view_init(4, -72)
#*************************************************************************************
lle= LocallyLinearEmbedding(n_neighbors=15, n_components=2,method='modified')
X_transformed=lle.fit_transform(X)
ax = fig.add_subplot(2, 3, 2)
ax.scatter(X_transformed[:, 0], X_transformed[:, 1],c=color, cmap=plt.cm.Spectral)
```

Figura 19. Ejemplo de python que aplica LLE al dataset llamado swiss roll. (Locally Linear Embedding). Fuente: elaboración propia.

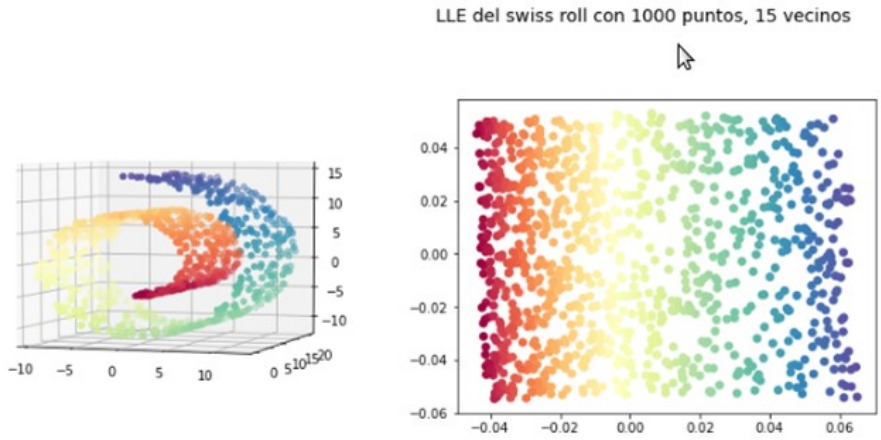


Figura 20. Resultado de la aplicación del método LLE al dataset llamado swiss roll. (Locally Linear Embedding). Fuente: elaboración propia.

¿Qué es lo que podemos ver del resultado de la aplicación de LLE al dataset swiss roll? Claramente, el dataset es un manifold no lineal de dos dimensiones embebido en un espacio de tres. Al aplicar el método, lo que estamos tratando de hacer es reducir la dimensionalidad del dataset sin perder la estructura de los datos. La gráfica de la izquierda muestra el dataset original dado en un espacio tridimensional y la gráfica de la derecha muestra el mismo dataset después de aplicar el método LLE en

dos dimensiones. Es importante que nos demos cuenta que la estructura de los datos es la misma aún representada en 2D.

Dentro de esta clase de métodos de reducción de la dimensionalidad y que se llama Manifold Learning también hay otros métodos, tales como: Isomap, MDS y t-SNE que están implementados en el módulo Scikit-Learn.

Hasta este punto hemos tratado lo concerniente a las técnicas de reducción de la dimensionalidad de un conjunto de datos y comenzaremos a revisar algunas técnicas de reducción de la numerosidad como Selección de casos o Instancias entre otras.

Selección de casos

En un conjunto de datos, puede haber casos que, o son redundantes o ruidosos o simplemente no aportan información para la tarea a mano. Debido a esto, como en el caso de selección de rasgos, la selección de casos permite disminuir la numerosidad del conjunto mejorando así el rendimiento de los algoritmos de aprendizaje automático.

Por tanto, la tarea de Selección de Instancias o casos se puede definir a groso modo como: Dado un conjunto de entrenamiento T, el objetivo de la selección de instancias, es obtener un subconjunto $S \subset T$tal que S no contenga instancias redundantes o superfluas y que $Acc(S) \approx Acc(T)$Donde Acc(X)denota la Accuracy obtenida, usando a X como conjunto de entrenamiento.

De menara general todos los métodos de selección de casos (Selección de Prototipos) tienen características comunes por los que se les puede ir clasificando. De manera generar se toman como criterios para hacerlo, los siguientes: Dirección de búsqueda, la estrategia de selección y la evaluación de la búsqueda.

Si tomamos en cuenta la forma en que comienza la evaluación de las instancias, entonces podemos clasificarlos en Incrementales o Decrementales, en Batch (por lotes), mezcladas o Prefijadas.

Clasificación de los métodos de Selección de casos, según la dirección de búsqueda:

- Incrementales.
- Decrementales.
- Batch.
- Mezcladas.
- Prefijadas.

Métodos Incrementales: Inician con el subconjunto S vacío (S=ϕ) y van incorporando instancias a medida que operan según un criterio predefinido.

Métodos decrementales: comienzan con el subconjunto S (S=T) con todas las instancias del conjunto T y las van eliminando en la medida que operan según el criterio preseleccionado por el método en particular.

Batch: La idea con este tipo de métodos es que marcan a cada instancia como elegible o no según algún criterio antes de eliminarlas y luego se eliminan de una vez.

Mezcladas: En este tipo de métodos el procesamiento se inicia con un subconjunto preseleccionado o aleatoriamente o por medio incremental o decremental e interativamente se le van añadiendo o eliminando instancias que satisfagan un determinado criterio.

Prefijadas: Este tipo de método constituye una subfamilia de los de direcciones mezcladas, pero con la diferencia que los números de operaciones de eliminación o adición son las mismas. Es decir, el número de instancias o prototipos al final es decidido al inicio de la fase y nunca es cambiado.

Muchos de los métodos de selección de casos se basan en la clasificación de las

instancias a partir de su ubicación dentro de las clases a las que pertenecen. Hay casos o instancias que se ubican en los bordes de las clases y otras más al centro de ellas. A las primeras se les llama instancias de frontera y las segundas instancias interiores.

Por tanto, se dice que una instancia $i_j \in T$ es de frontera de la clase C_i si $i_j \in C_i$ y es vecino más cercano de una instancia $i_k \notin C_i$, de otra forma se le conoce como instancia interior de la clase C_i.

Basados en este tipo de clasificación de las instancias los métodos de selección de casos se pueden dividir en (estrategia de selección):

- Condensación.
- Edición.
- Híbridos.

Condensación: La idea básica de este tipo de métodos es retener las instancias de frontera, pues estás afectan mucho más la calidad de la clasificación que las interiores. Sin embargo, se ha demostrado que, aunque mantienen la calidad de la clasificación en el conjunto de entrenamiento, la generalización se ve afectada. Estos métodos tienen una gran capacidad de reducción, pues la cantidad de instancias de frontera es mucho menor que las interiores.

Edición: En este tipo de métodos, al contrario de los de condensación, lo que se busca es mantener las interiores y descargar las de frontera. Eliminan también las que son ruidosas o no están de acuerdo a sus vecinos.

Híbridos: Estos métodos buscan encontrar el subconjunto S menor que maximice la generalización, es decir la calidad de la clasificación sobre un conjunto de prueba. Esto les permite eliminar un tipo u otra de instancia basado en dos estrategias determinadas.

Los métodos de selección de casos, también se pueden clasificar, basándonos en la manera en cómo se decide si se elimina o se mantiene la instancia, en:

- Métodos Filtros.
- Métodos envoltorios. (Wrappers)

Filtros: En este tipo de método la decisión de de mantener o eliminar una instancia se hace por medio de una Heurística o determinada regla.

Envoltorios (Wrapper): En este tipo de métodos, la decisión se toma basado en un clasificador, usualmente KNN.

Otro de los elementos a tener en cuenta para la selección de los métodos de selección de instancias es la complejidad computacional, que, aunque no la tocaremos, sí que es uno de los elementos importantes.

Aunque se han propuesto cientos de algoritmos en la literatura, todos los días se proponen nuevos y es un área de investigación de constante actividad. Con esa gran cantidad de métodos es imposible que podamos abarcar muchos de ellos. En la siguiente figura se muestran muchos de ellos clasificados por los criterios anteriores.

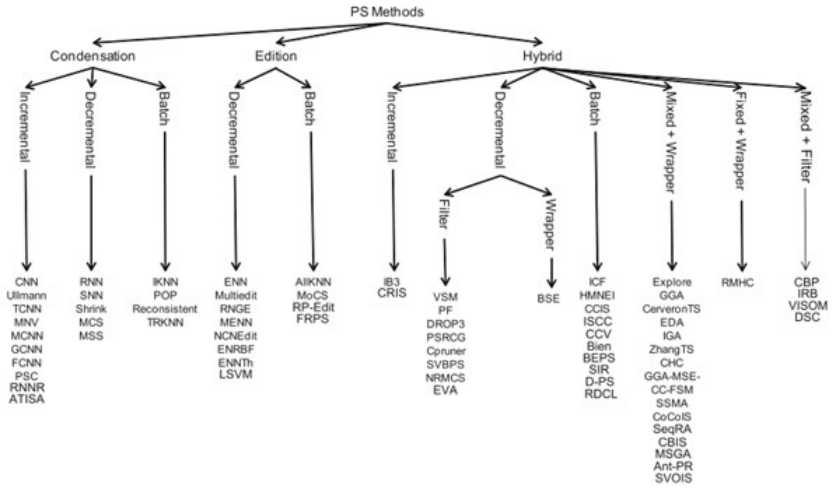


Figura 21. Clasificación de los métodos de selección de casos o instancias. (Data Preprocessing in Data

Mining). Fuente: elaboración propia.

Con la imposibilidad de describir la mayoría de estos métodos, elegimos algunos que son representativos, tales como:

- CNN. (Condensed Nearest Neighbours)
- ENN. (Edited Nearest Neighbour)
- DROP. (Decremental Reduction Optimization)
- ICF. (Iterative Case Filtering)

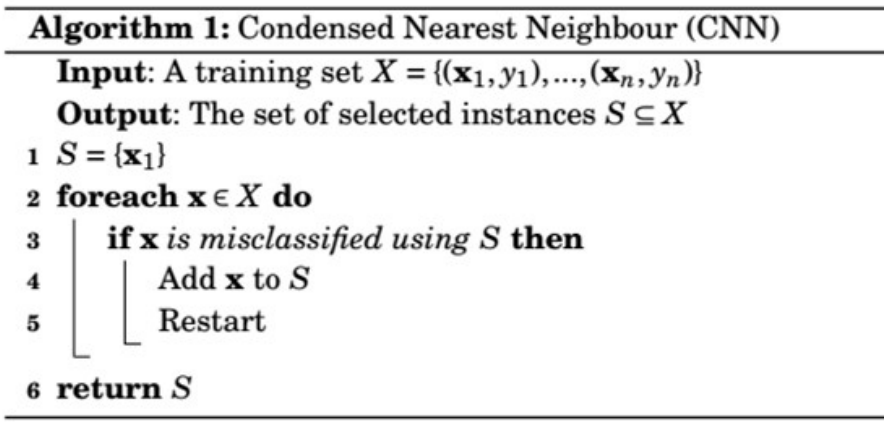
Condensed Nearest Neighbours (CNN): Este método se considera la primera propuesta formal del uso de KNN para el problema de selección de casos. Este método se basa en el concepto de consistencia de un conjunto.

Definición de consistencia:

Se dice que un conjunto S es consistente con otro X, si:

Siendo X un conjunto de entrenamiento y S un subconjunto de X, entonces un clasificador KNN, usando S como conjunto de entrenamiento, puede clasificar correctamente todas las instancias en X.

Siguiendo esta definición entonces S, debe ser consistente con X y por tanto, idealmente, más pequeño.



Algorithm 1: Condensed Nearest Neighbour (CNN)

Input: A training set $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$

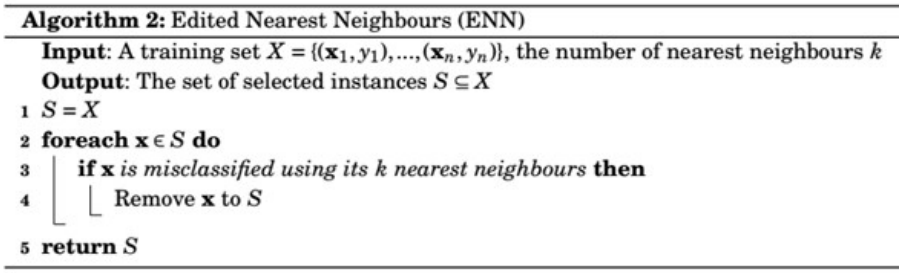
Output: The set of selected instances $S \subseteq X$

```
1 S = {x_1}
2 foreach x ∈ X do
3     if x is misclassified using S then
4         Add x to S
5         Restart
6 return S
```

Figura 20. Método Condensed Nearest Neighbours. (CNN). Fuente: elaboración propia.

Edited Nearest Neighbour. (ENN): Este es un método decremental, es decir, comienza con el conjunto S igual al conjunto de entrenamiento X y cada instancia es eliminada si no es bien clasificada por la regla KNN. El número de vecinos más cercanos es un parámetro del algoritmo y el artículo original se estable en 3.

ENN, elimina los casos ruidosos y las instancias de frontera. Las instancias interiores no son afectadas por el proceso de edición del método.



Algorithm 2: Edited Nearest Neighbours (ENN)

Input: A training set $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, the number of nearest neighbours k

Output: The set of selected instances $S \subseteq X$

```
1 S = X
2 foreach x ∈ S do
3     if x is misclassified using its k nearest neighbours then
4         Remove x to S
5 return S
```

Figura 23. Edited Nearest Neighbour. (ENN). Fuente: elaboración propia.

Decremental Reduction Optimization Procedure. (DROP): Esta familia de algoritmos se propusieron en el 2000 y están considerados entre los mejores métodos para selección de instancias para la clasificación. El criterio de eliminación

de las instancias se basa en dos conceptos “asociados” y “vecinos cercanos”. La relación de “asociado” es lo opuesto a vecino cercano. Por ejemplo, si una instancia p, tiene a q como vecino cercano, entonces, q tiene como asociado a p. Ver la fig. 20.

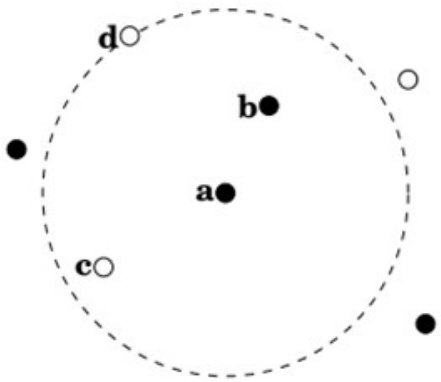


Figura 23. En knn con k=3. La instancia a tiene como vecinos más cercanos a b,c,d y ellos tienen como asociado a. Fuente: elaboración propia.

Algorithm 3: Decremental Reduction Optimization Procedure 3 (DROP3)

Input: A training set $X = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, the number of nearest neighbours k
Output: The set of selected instances $S \subseteq X$

```
Noise filtering: remove any instance in X misclassified by its k neighbours
S = X
Sort instances in S by the distance to their nearest enemy
foreach instance x ∈ S do
    Find x.N_{1...k+1}, the k+1 nearest neighbours of x in S
    Add x to each of its neighbour's list of associates
foreach instance x ∈ S do
    Let with = # of associates of x correctly classified by x as a neighbour
    Let without = # of associates of x correctly classified without x
    if without ≥ with then
        Remove x from S
        foreach associate a of x do
            Remove x from a's list of nearest neighbour
            Find a new nearest neighbour for a
            Add a to its new neighbour's list of associates
return S
```

Figura 24. Decremental Reduction Optimization Procedure. (DROP). Fuente: elaboración propia.

Iterative Case Filtering. (ICF): Este método, usa dos conceptos muy relacionados con conceptos de vecindad y asociado de DROP y son: coverage y reachable. La