

2. Algoritmos de IA

Introducción

- Los algoritmos de aprendizaje automático se pueden dividir en 3 grandes categorías:
 - Aprendizaje supervisado
 - Aprendizaje no supervisado
 - Aprendizaje de refuerzo
- **El aprendizaje supervisado** es útil en los casos en que una propiedad (etiqueta) está disponible para un determinado conjunto de datos (conjunto de entrenamiento), pero falta y necesita ser predicha para otras instancias.
- El aprendizaje no supervisado es útil en casos donde el desafío es descubrir relaciones implícitas en un determinado conjunto de datos no etiquetados (los elementos no están preasignados)
- El aprendizaje de refuerzo se encuentra entre estos dos extremos: hay alguna forma de retroalimentación disponible para cada paso o acción predictiva, pero no hay una etiqueta precisa o mensaje de error.
- La IA necesita de la utilización de algoritmos. Un algoritmo es un conjunto de instrucciones inequívocas que una computadora mecánica puede ejecutar. Un algoritmo complejo a menudo se construye sobre otros algoritmos más simples.
- Muchos algoritmos de IA son capaces de aprender de los datos, pueden mejorarse aprendiendo nuevas heurísticas (estrategias o "reglas generales" que han funcionado bien en el pasado), o pueden escribir otros algoritmos por sí mismos.



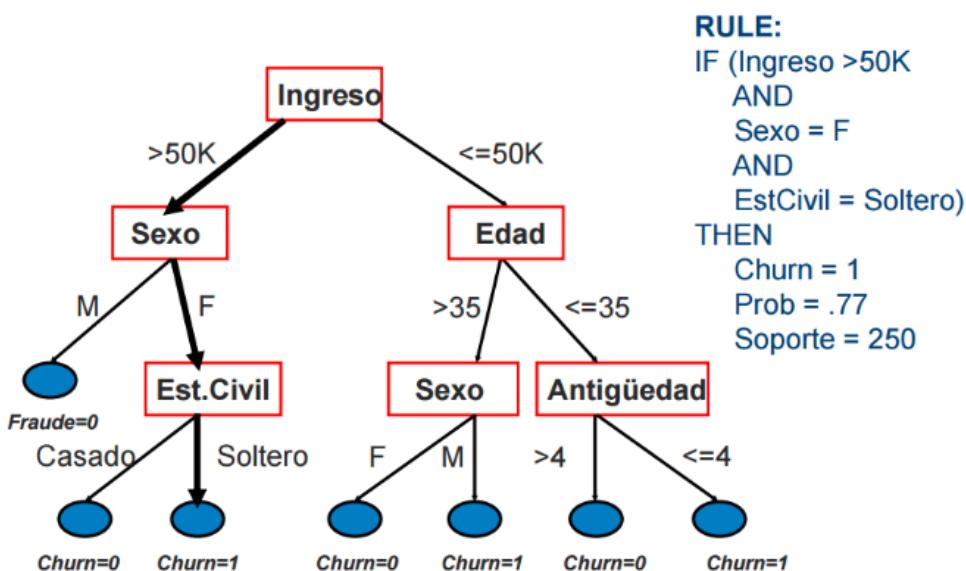
Machine Learning: modelos supervisados

ALGORITMOS SUPERVISADOS:

- Estos algoritmos emplean un conjunto de datos de entrenamiento etiquetados (preclasiificados), los cuales procesan para realizar predicciones sobre los mismos, corrigiéndolos cuando son incorrectas. El proceso de entrenamiento finaliza una vez que ha alcanzado un nivel adecuado de precisión.
- El árbol de decisión es el más utilizado en el aprendizaje supervisado.
- Estos algoritmos suelen denominarse CART o Classification and Regression Trees. Por tanto, podemos distinguir entre los Classification Trees, que son aquellos cuyo objetivo es predecir valores discretos, como un clasificador de emails en spam o no spam; y Regression Trees, orientados a predecir valores continuos, como el precio de una vivienda.

ÁRBOLES DE DECISIÓN:

- Dados los datos de atributos junto con sus clases, un árbol de decisión produce una secuencia de reglas que pueden usarse para clasificar los datos.
- El árbol de decisión, como su nombre lo dice, toma una decisión con el modelo en forma de árbol.
- Divide la muestra en dos o más significativos en sus variables de entrada.
- Para elegir un diferenciador (predictor), el algoritmo considera todas las características y realiza una división binaria en ellas (para datos categóricos, divididos por cat; para continuo, elija un umbral de corte). Luego elegirá el que tenga el menor costo (es decir, la mayor precisión) y se repite recursivamente, hasta que divide con éxito los datos en todas las hojas (o alcance la profundidad máxima).
- Un árbol de decisión es una herramienta de apoyo a la decisión que utiliza un gráfico o modelo de decisiones en forma de árbol y sus posibles consecuencias, incluidos los resultados de eventos fortuitos, los costes de recursos y la utilidad.



- Desde el punto de vista de la decisión comercial, un árbol de decisión es el número mínimo de preguntas de sí/no que uno tiene que hacer, para evaluar la probabilidad de tomar una decisión correcta (en la mayoría de los casos).
- Como método, nos permite abordar el problema de una manera estructurada y sistemática para llegar a una conclusión lógica.
- **Ventajas:** el árbol de decisión es fácil de entender y visualizar, requiere poca preparación de datos y puede manejar datos numéricos y categóricos.
- **Desventajas:** el árbol de decisión puede crear árboles complejos que no se generalizan bien, y los árboles de decisión pueden ser inestables porque pequeñas variaciones en los datos pueden generar un árbol completamente diferente.

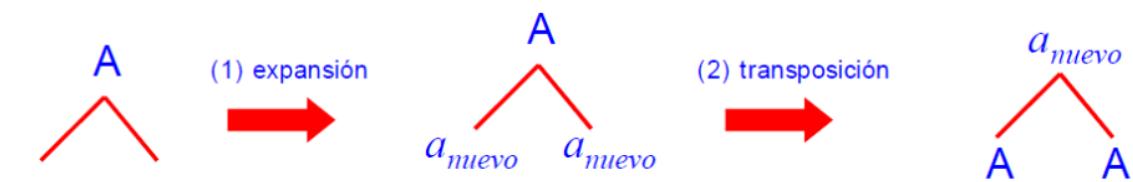
CUÁNDO ES ADECUADO UTILIZAR UN ÁRBOL DE DECISIÓN?

- Las instancias del concepto son representadas por pares atributo-valor.
- La función objetivo tiene valores de salida discretos.
- Las descripciones del objeto son disyuntivas.
- El conjunto de aprendizaje tiene errores.
- El conjunto de aprendizaje es incompleto.

REESTRUCTURANDO ÁRBOLES (PULL-UP)

- La reestructuración de un árbol consiste en poner en los nodos superiores aquellos atributos que mejor expliquen la clasificación de instancias de acuerdo con el **indicador E**.
- A este proceso se le denomina “pull-up” y corresponde al siguiente esquema algorítmico:
 - Si el atributo a subir nuevo está en la raíz
 - Entonces fin_del_método sino:
 1. Recursivamente subir el atributo nuevo a la raíz del subárbol inmediato. Convertir cualquier árbol no expandido en uno expandido eligiendo nuevo el atributo a testear.
 2. Transponer el árbol de forma que anevo se sitúe en la raíz y la antigua raíz como raíz de cada subárbol dependiente de anevo.

Ejemplo



CLASIFICACIÓN NAIVE (INGENUA) DE BAYES

- Naive Bayes es un clasificador probabilístico inspirado en el teorema de Bayes. Bajo un supuesto simple, que es, los atributos son condicionalmente independientes.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood Class Prior Probability
 ↓ ↑
 Posterior Probability Predictor Prior Probability

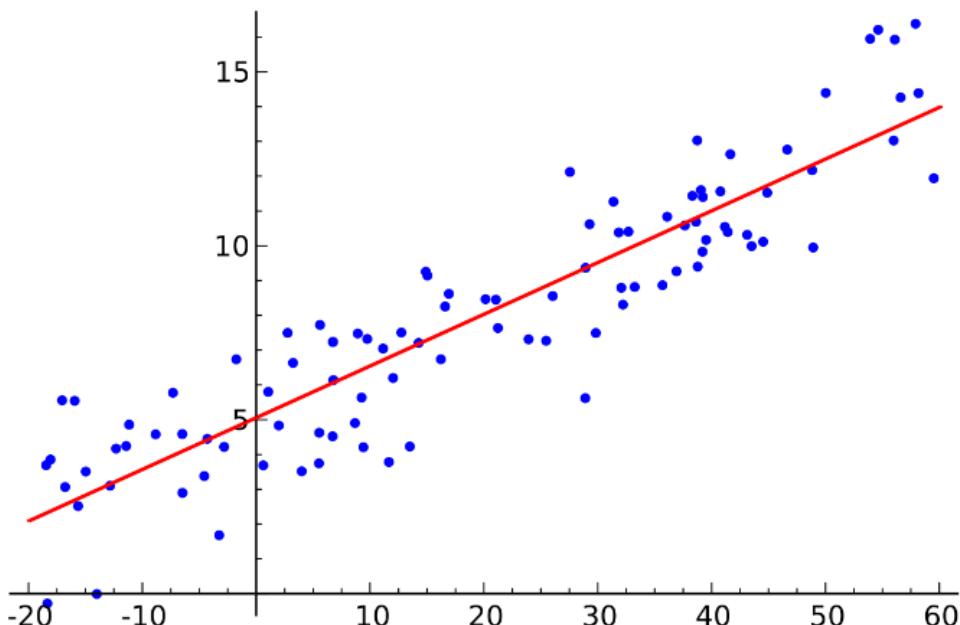
$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

- La clasificación se realiza derivando el máximo posterior, que es el P máximo ($C_i | X$) con la suposición anterior aplicada al teorema de Bayes.
- Esta suspensión reduce en gran medida el costo computacional al contar solo la distribución de la clase. A pesar de que la suposición no es válida en la mayoría de los casos, ya que los atributos son dependientes, sorprendentemente Naive Bayes ha podido desempeñarse de manera impresionante.
- Naive Bayes es un algoritmo muy simple de implementar y se han obtenido buenos resultados en la mayoría de los casos.
- Puede ser fácilmente escalable a conjuntos de datos más grandes, ya que lleva tiempo lineal, en lugar de una aproximación iterativa costosa como se usa para muchos otros tipos de clasificadores.
- Naive Bayes puede sufrir un problema llamado problema de probabilidad cero. Cuando la probabilidad condicional es cero para un atributo particular, no puede dar una predicción válida. Esto necesita ser arreglado explícitamente usando un estimador laplaciano.
- **Ventajas:** este algoritmo requiere una pequeña cantidad de datos de entrenamiento para estimar los parámetros necesarios. Los clasificadores ingenuos de Bayes son extremadamente rápido en comparación con los métodos más sofisticados.
- **Desventajas:** se sabe que Naive Bayes es un mal estimador.
- **Pasos para la implementación:**
 - Inicialice el clasificador que se utilizará
 - Entrene al clasificador: todos los clasificadores en scikit-learn usan un método de ajuste (X, y) para ajustar el modelo (entrenamiento) para los datos de tren dados X y la etiqueta de tren y .
 - Predecir el objetivo: dada una observación X sin etiqueta, la predicción X devuelve la etiqueta predicha y
 - Evaluar el modelo clasificador

- Algunos ejemplos en mundo real son:
 - Marcar un correo electrónico como spam
 - Clasificar un artículo de noticias sobre tecnología, política o deportes
 - Verificar si un texto expresa emociones positivas o negativas
 - Utilizado para el software de reconocimiento facial

REGRESIÓN DE MÍNIMOS CUADRADOS ORDINARIOS

- Mínimos cuadrados es un método para realizar regresión lineal. Podemos pensar en la regresión lineal como la tarea de ajustar una línea recta a través de un conjunto de puntos. Hay varias estrategias posibles para hacer esto.
- La estrategia de “mínimo cuadrados ordinarios” es la siguiente: dibujamos una línea y luego, para cada uno de los puntos de datos, medimos la distancia vertical entre el punto y la línea, y procedemos a sumarlos. La línea ajustada sería aquella donde esta suma de distancia es lo más pequeña posible.

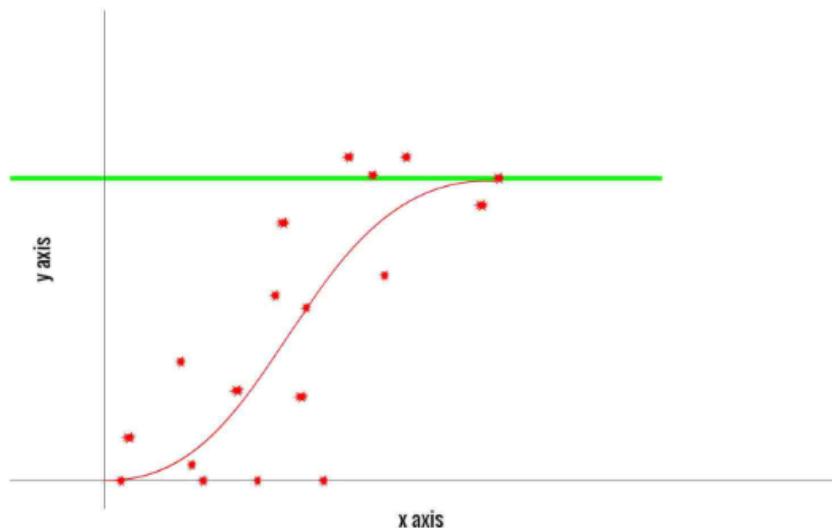


- **LINEAL** se refiere al tipo de modelo que se está utilizando para ajustar los datos, mientras que los mínimos cuadrados se refieren al tipo de métrica de error que está minimizando.
- La idea de la regresión lineal es establecer la relación entre las variables independientes y dependientes por medio de ajustar una mejor línea recta con respecto a los puntos.
- Esta línea de mejor ajuste se conoce como línea de regresión y está representada por la siguiente ecuación lineal:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

REGRESIÓN LOGÍSTICA

- Es una forma estadística muy potente para modelar un resultado binomial con una o más variables explicativas.
- Mide la relación entre la variable dependiente categórica y una o más variables independientes mediante la estimación de probabilidades utilizando una función logística, que es la distribución logística acumulativa.



- Los modelos lineales, también pueden ser utilizados para clasificaciones; es decir, que primero ajustamos el modelo lineal a la probabilidad de que una cierta clase o categoría ocurra y, a luego, utilizamos una función para crear un umbral en el cual especificamos el resultado de una de estas clases o categorías. La función que utiliza este modelo, no es ni más ni menos que la función logística:

$$f(x) = \frac{1}{1+e^{-x}}$$

Veamos, aquí también un pequeño ejemplo en Python:

```
In [7]: # Creando un dataset de ejemplo
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4)

In [8]: # Importando el modelo
from sklearn.linear_model import LogisticRegression

rlog = LogisticRegression() # creando el modelo

# Dividiendo el dataset en entrenamiento y evaluacion
X_entrenamiento = X[:-200]
X_evaluacion = X[-200:]
y_entrenamiento = y[:-200]
y_evaluacion = y[-200:]

rlog.fit(X_entrenamiento, y_entrenamiento) #ajustando el modelo

# Realizando las predicciones
y_predic_entrenamiento = rlog.predict(X_entrenamiento)
y_predic_evaluacion = rlog.predict(X_evaluacion)

In [9]: # Verificando la exactitud del modelo
entrenamiento = (y_predic_entrenamiento == y_entrenamiento).sum().astype(float) / y_entrenamiento.shape[0]
print("sobre datos de entrenamiento: {:.2f}".format(entrenamiento))
evaluacion = (y_predic_evaluacion == y_evaluacion).sum().astype(float) / y_evaluacion.shape[0]
print("sobre datos de evaluación: {:.2f}".format(evaluacion))

sobre datos de entrenamiento: 0.92
sobre datos de evaluación: 0.91
```

Ejemplo regresión logística

- En general, las regresiones se pueden usar en aplicaciones del mundo real como:
 - Puntuación de crédito
 - Medir las tasas de éxito de las campañas de marketing
 - Predecir los ingresos de un determinado producto
 - Evaluar si se producirá un terremoto en un día en particular

MÁQUINAS DE VECTORES DE SOPORTE (SUPPORT VECTOR MACHINE)

- Es un algoritmo de clasificación binario. Dado un conjunto de puntos de 2 tipos en lugar de N dimensiones, SVM genera un hiperplano dimensional ($N - 1$) para separar esos puntos en 2 grupos.
- Digamos que tiene algunos puntos de 2 tipos en un documento que son linealmente separables, SVM encontrará una línea recta que separa esos puntos en 2 tipos y está situada lo más lejos posible de todos esos puntos.
- Mientras la mayoría de los métodos de aprendizaje tienen como objetivo minimizar los errores cometidos por el modelo generado a partir de los ejemplos de entrenamiento (error empírico), el sesgo inductivo asociado a la SVMs radica en la minimización del denominado **riesgo estructural**. La idea es seleccionar un

hiperplano de separación equidistante de los ejemplos más cercanos de cada clase para lograr un **margen máximo** a cada lado del hiperplano.

Ejemplo SVM:

```
In [14]: # importando SVM
from sklearn import svm

# importando el dataset iris
iris = datasets.load_iris()
X = iris.data[:, :2] # solo tomamos las primeras 2 características
y = iris.target

h = .02 # tamaño de la malla del grafico

# Creando el SVM con sus diferentes métodos
C = 1.0 # parametro de regulacion SVM
svc = svm.SVC(kernel='linear', C=C).fit(X, y)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)
lin_svc = svm.LinearSVC(C=C).fit(X, y)

# crear el area para graficar
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

# titulos de los graficos
titles = ['SVC con el motor lineal',
          'LinearSVC',
          'SVC con el motor RBF',
          'SVC con el motor polinomial']

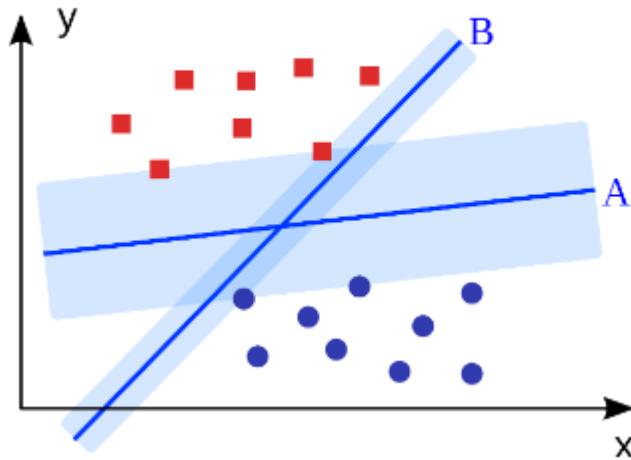
for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):
    # Realizando el gráfico, se le asigna un color a cada punto
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)

    # Graficando tambien los puntos de datos
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
    plt.xlabel('largo del petalo')
    plt.ylabel('ancho del petalo')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])
```

- El término de escala, algunos de los mayores problemas que se han resuelto utilizando SVM con la publicidad gráfica, el reconocimiento de “splice site” humano en genética, detección de género basada en imágenes, clasificación de imágenes a gran escala, etc.



- En la actualidad existe un número importante de repositorios web y de paquetes software de libre distribución dedicados a la implementación de SVMs y muchas de sus variantes.
- Algunos de estos paquetes software son:
 - 1. LIBSVM
La librería LIBSVM es un paquete software pensado para resolver problemas de clasificación y regresión mediante máquinas de vectores soporte. Entre sus principales características, cabe citar que es un software de código abierto (disponible en C++ y Java); implementa diferentes formulaciones SVM con la posibilidad de usar diferentes tipos de kernels; permite la clasificación multiclase y la posibilidad de usar técnicas de validación cruzada para la selección de modelos. También ofrece interfaces para una gran cantidad de lenguajes de programación (Python, R, MATLAB, Perl, Ruby, Weka, Common, LISP, CLISP, Haskell, OCaml, LabVIEW, y PHP). Además, en su [página web](#) dispone de un applet para implementar sencillos problemas de clasificación y de regresión en dos dimensiones.
 - 2. SVMLight
Es una implementación en C de máquinas de vectores de soporte. Entre sus principales características el permitir resolver no sólo problemas de clasificación y de regresión, sino también problemas de ranking; permite manejar varios cientos de miles de ejemplos de entrenamiento, junto con muchos miles de vectores soporte; soporta funciones kernel estándar y, además, permite al usuario definir sus propias funciones kernel. Como novedad presenta una implementación SVM, llamada SVMStruct, para predecir salidas estructuradas o multivariadas, como pueden ser conjuntos, secuencias y árboles. [Enlace](#)

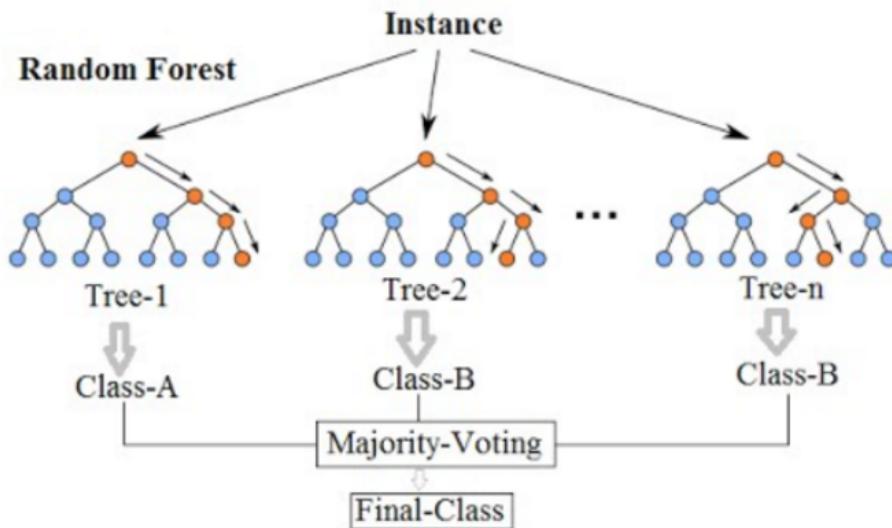
MÉTODOS DE ENSAMBLAJE

- Son algoritmos de aprendizaje que construyen un conjunto de clasificaciones y luego clasifican nuevos datos con puntos mediante un voto ponderado de sus predicciones.
- El método de conjunto original es el promedio Bayesiano, pero los algoritmos más recientes incluyen la codificación de salida de corrección de errores, el embolsado y el refuerzo.
- Los métodos de ensamblaje aportan una serie de ventajas frente a los modelos individuales:
 - Promueven riesgos.
 - Reducen la varianza: la opinión agregada de un grupo de modelos es menos ruidosa que la opinión individual de uno de los modelos. En finanzas, esto se llama diversificación: una cartera mixta de muchas acciones será mucho menos variable que solo uno de los valores. Es por eso que sus modelos serán mejores con más puntos de datos en lugar de menos.
 - Es poco probable que se ajusten demasiado.

RANDOM FOREST

- El bosque aleatorio es un modelo de conjunto que hace crecer múltiples árboles y clasifica objetos según los “votos” de todos los árboles, es decir, un objeto se asigna a una clase que tiene más votos de todos los árboles. Al hacerlo, el problema con un alto sesgo(sobreajuste) podría aliviarse.

Random Forest Simplified



- Random forest es un metaestimulador que se ajusta a varios árboles de decisión en varias submuestras de conjuntos de datos y utiliza el promedio para mejorar la precisión predictiva del modelo y controla el sobreajuste. El tamaño de la

submuestra siempre es el mismo que el tamaño de la muestra de entrada original, pero las muestras se extraen con reemplazo.

- La técnica random forest, que introduce mejoras sobre árboles creados usando el método bagging.
- El principal problema de Random Forest es que es más difícil de interpretar.
- Ya no se dispone de un único árbol en el que se puede observar el efecto de cada variable, sino de un **gran número** de ellos cuyo efecto conjunto no puede visualizarse.
- Por otro lado, random forest permite obtener medidas con relación a la importancia que los diferentes predictores han tenido en el modelo, por tanto, permite en parte interpretar este.
- La importancia de los predictores se evalúa como el número de veces que han sido utilizados por los diversos árboles y su capacidad para reducir el índice de Gini en ellos.

VENTAJAS DE RANDOM FOREST

- Podría manejar un gran conjunto de datos con alta dimensionalidad, salida en función de la importancia de la variable, útil para explorar los datos.
- Podría manejar falta de datos mientras se mantiene la precisión.
- En la mayoría de los casos , la reducción y ajuste del clasificador Random Forest son más precisos que los árboles de decisión.

DESVENTAJAS DE RANDOM FOREST

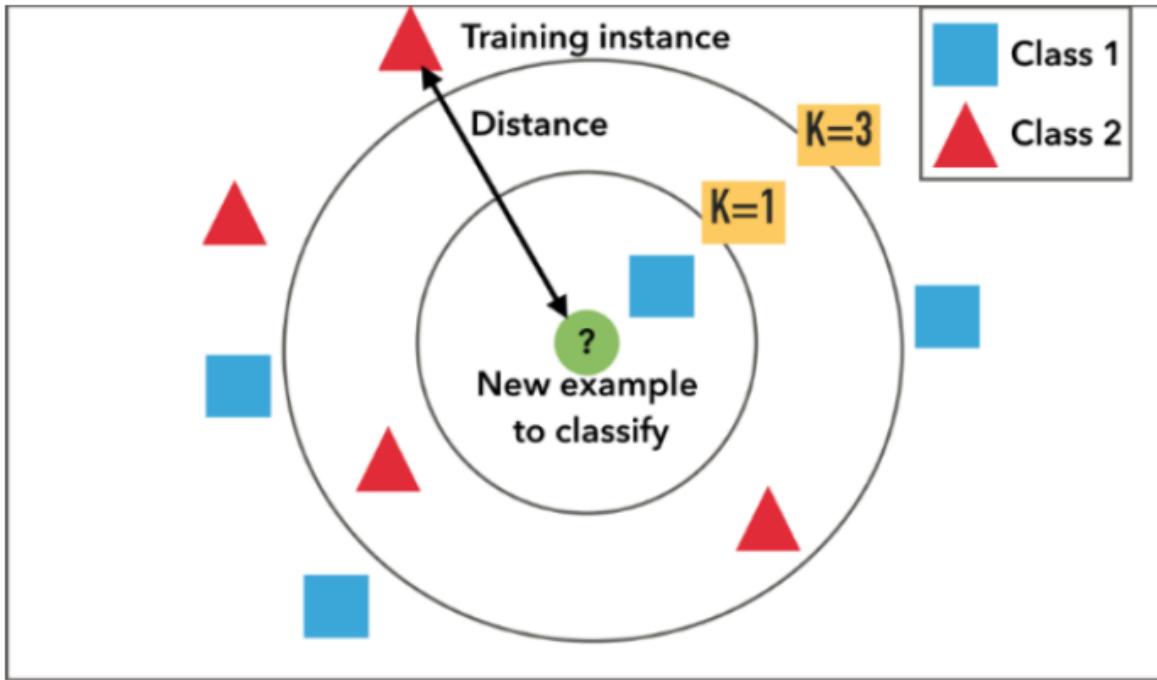
- Podría ser una caja negra, los usuarios tienen poco control sobre lo que hace modelo.
- La predicción es lenta en tiempo real, difícil de implementar y el algoritmo es complejo.

KNN clasifica un objeto por mayoría de votos de los vecinos del objeto, en espacio del parámetro de entrada. El objeto se asigna a la clase que es más común entre sus **k (un entero especificado por humanos) vecino más cercano**.

Es un **algoritmo no paramétrico, perezoso (lazy)**. No es paramétrico, ya que no supone nada sobre la distribución de datos (los datos no tienen que distribuirse normalmente).

Es perezoso ya que realmente no aprende ningún modelo y hace generalización de los datos (no entrena algunos parámetros de alguna función donde la entrada X de salida y).

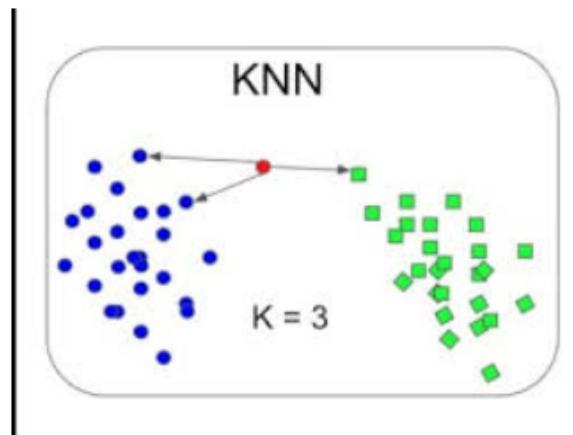
Hablando estrictamente, esto no es realmente un algoritmo de aprendizaje. Simplemente clasifica objetos según la similitud de características (característica = variable de entrada)



La clasificación se calcula a partir de un voto de mayoría simple de los k vecinos más cercanos de cada punto.

Ventajas: este algoritmo es simple de implementar, robusto frente al ruido, y efectivo si los datos de entrenamiento son grandes.

Desventajas: es necesario determinar el valor de K y el coste de cálculo es alto, ya que necesita calcular la distancia de cada instancia a todas las muestras de entrenamiento.



Algoritmo KNN

Debemos especificar una **métrica** para poder medir la proximidad. Suele utilizarse por razones computacionales la distancia Euclídea, para este fin.

Denominaremos conjunto de referencia (y lo notaremos por R) al conjunto de prototipos sobre el que se buscará el(s) vecino(s) más cercano(s).

Si $K_i(X)$ es el número de muestras de la clase presentes en los k vecinos más próximos a X, esta regla puede expresarse como:

$$d(x) = \text{wci}_K(X) = \max_i = 1 \rightarrow J_{K_i}(K)$$

Machine Learning: modelos no supervisados

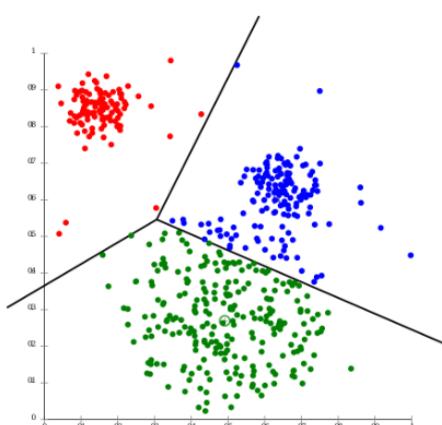
- El modelo no supervisado encuentra **patrones** donde no tenemos una base de datos que contenga respuestas correctas. Esto puede ser debido a diversos motivos:
 - Que estas respuestas correctas sean inobservables.
 - Que no sea posible obtenerlas.
 - Que no exista una respuesta correcta per se.
- Este sistema se emplea sin usar **ningún tipo de datos etiquetado**. No recibe outputs ni correlaciones entre outputs y respuestas “correctas”, sino que el algoritmo debe **explorar** los datos y encontrar algún tipo de patrón o estructura

Importante!

Este tipo de modelo tiene sentido cuando los datos son de carácter transaccional.

ALGORITMOS DE CLUSTERING(AGRUPACIÓN)

- **Clustering** es la tarea de agrupar un conjunto de objetos de modo que los objetos en el mismo grupo (cluster) sean más similares entre sí que con los de otros grupos.
- Los objetos de un **cluster** deben presentar distintas características a aquellos agrupados en otro. Distinguimos entre **hard** y **soft** clustering.
- En el primero, un objeto será miembro de un cluster o no, mientras que en el segundo podría pertenecer a varios.



- Los análisis de clústeres se pueden realizar en dos fases, **jerárquico** o de **K-medias**.
- Cada uno de estos procedimientos emplea un algoritmo diferente en la creación de clústeres y tiene opciones que no están disponibles en los otros.

ANÁLISIS DE CLÚSTERES EN DOS FASES

- En algunas aplicaciones, se puede seleccionar como método el procedimiento. Análisis de clústeres en dos fases. Ofrece una serie de características exclusivas que se detallan a continuación
- Selección automática del número más apropiado de clústeres y medidas para la selección de los distintos modelos de clúster. Posibilidad de crear modelos de clúster basados al mismo tiempo en variables categóricas y continuas. Posibilidad de guardar el modelo de clúster en un archivo XML externo y, a continuación, leer el archivo y actualizar el modelo de clúster con datos más recientes.
- Además, el procedimiento Análisis de clústeres en dos fases permite analizar **archivos de datos grandes**.

ANÁLISIS DE CLÚSTERES JERÁRQUICO

- El uso del procedimiento Análisis de clústeres jerárquico se dirige a **archivos de datos más pequeños**(cientos de objetos por agrupar en clústeres) y ofrece las siguientes características exclusivas:
 - Posibilidad de agrupar en clústeres casos o variables
 - Posibilidad de calcular un rango de soluciones posibles y guardar los clústeres de pertenencia para cada una de dichas soluciones.
 - Distintos métodos de formación de clústeres, transformación de variables y medida de disimilitud entre clústeres.
- Cuando todas las variables sean del mismo tipo, el procedimiento Análisis de clústeres jerárquico podrá analizar variables de intervalo (continuas), de recuento o binarias.

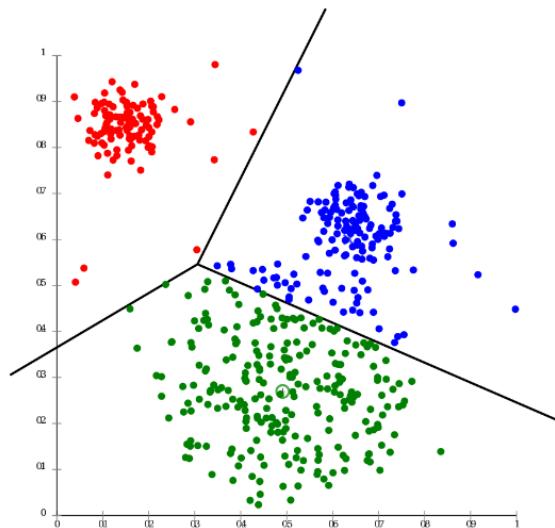
ANÁLISIS DE CLÚSTERES DE K-MEDIAS

- El uso de procedimiento Análisis de clústeres de K-medias se dirige a **datos continuos** y requiere que el usuario indique previamente el número de clústeres, a continuación explicamos las características que ofrece:
 - Posibilidad de guardar las distancias desde los centros de los clústeres hasta los distintos objetos.
 - Posibilidad de leer los centros de los clústeres iniciales y guardar los centros de los clusters finales un archivo IBM SPSS statistics externo.
- También, el procedimiento Análisis de clústeres de K-medias puede analizar **archivos de datos grandes**.
- El clustering **no es un algoritmo** en sí mismo, sino que existen distintos métodos de llevar a cabo esta técnica, entre los que podemos destacar hierarchical (jerárquico), k-means o DBSCAN. Estos métodos difieren en propiedades, y por eso son aplicados en función del data-set en cuestión, así como el objetivo que se pretende.

- Cada algoritmo de clustering es diferente, veamos algunos ejemplos:
 - Algoritmos basados en centroides
 - Algoritmos basados en conectividad
 - Algoritmos basados en densidad
 - Probabilístico
 - Reducción de dimensionalidad
 - Redes neuronales/aprendizaje profundo

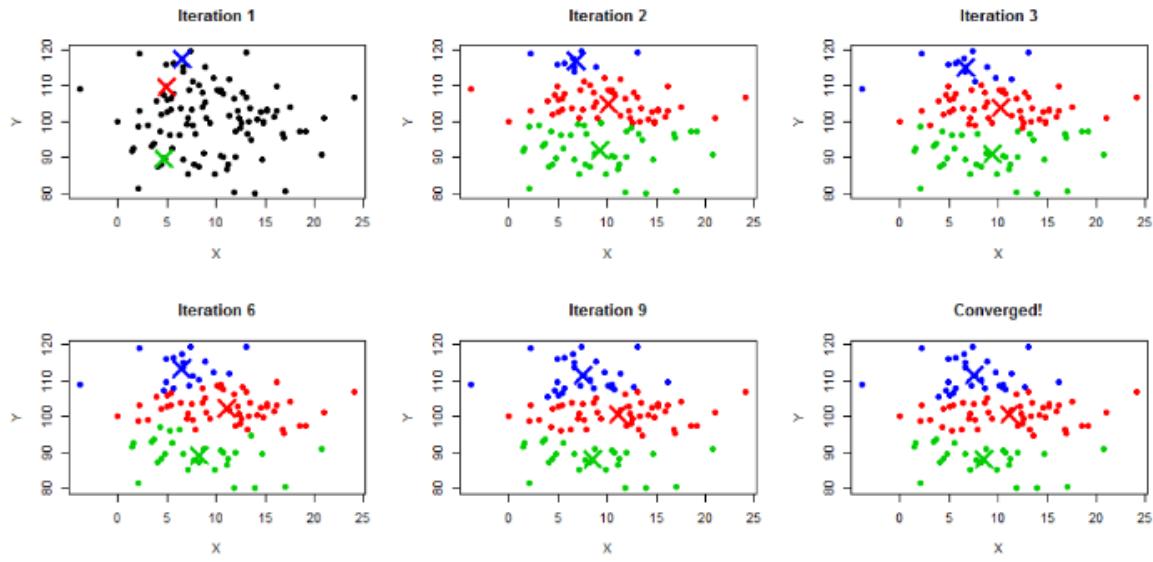
DIFERENTES TIPOS DE ALGORITMOS DE CLUSTERING

- K-means es probablemente el algoritmo de agrupación más conocido. Se enseña en muchas clases introductorias de ciencia de datos y aprendizaje automático. Es fácil de entender e implementar.



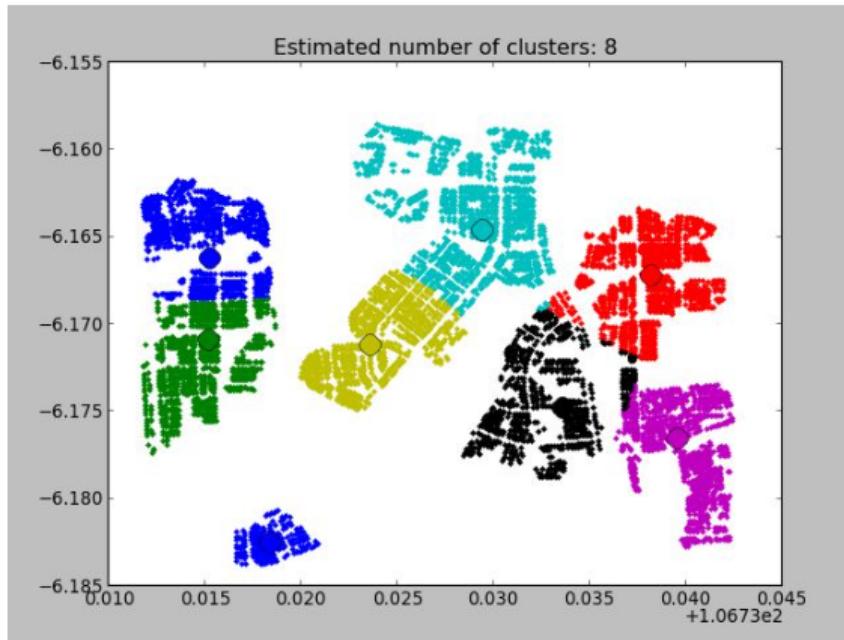
1. Para comenzar, primero seleccionamos un número de clases/ grupos para usar e inicializamos aleatoriamente sus respectivos puntos centrales (centroides). Para determinar la cantidad de clases a utilizar, es bueno echar un vistazo rápido a los datos e intentar identificar cualquier agrupación distinta. Los puntos centrales son vectores de la misma longitud que cada vector de punto de datos y son los "X" en el gráfico anterior.
2. Cada punto de datos se clasifica calculando las distancias entre ese punto y cada centro de grupo, y luego clasificando el punto para estar en el grupo cuyo centro está más cerca de él.
3. En base a estos puntos clasificados, calculamos el centro de grupo tomando la media de todos los vectores en el grupo.
4. Repetimos estos pasos para un número determinado de iteraciones o hasta que los centros de grupo no cambian mucho entre iteraciones. También se puede optar por inicializar aleatoriamente los centros de grupo varias veces y luego seleccionar la ejecución que parece que proporcionó los mejores resultados.

- K-means tiene la ventaja de que es bastante **rápido**, ya que todo lo que realmente estamos haciendo es calcular las distancias entre puntos y centros grupales. Esto supone hacer muy pocos cálculos. Por lo tanto, tiene una complejidad lineal $O(n)$.



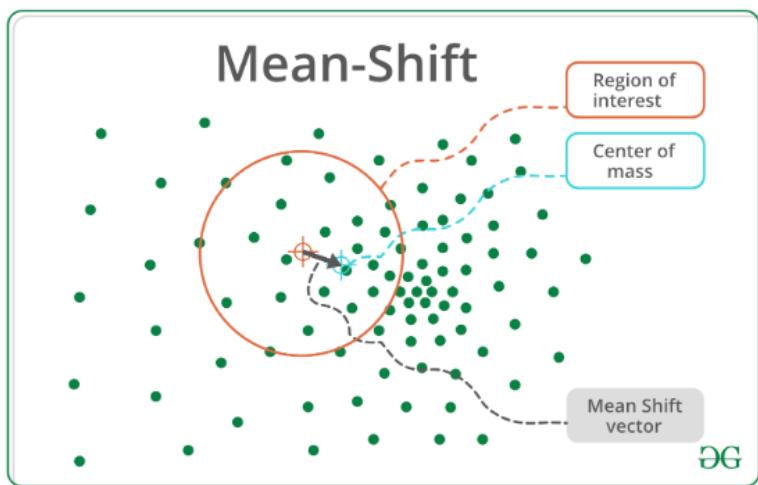
Ejemplo de un modelo K-MEANS

- Por otro lado, K-means tiene un par de desventajas:
 - Debe seleccionar cuántos grupos / clases hay. Esto no siempre es trivial e idealmente con un algoritmo de agrupamiento nos gustaría que los descubriera porque el objetivo es obtener una idea de los datos.
 - K-means también comienza con una elección aleatoria de centros de agrupación y, por lo tanto, puede producir diferentes resultados de agrupación en diferentes ejecuciones y carecer de consistencia. Otros métodos de clúster son más consistentes.
- **K-medians** es otro algoritmo de agrupamiento relacionado con K-means, excepto que en lugar de volver a calcular los puntos centrales del grupo usando la media, usamos el vector mediano del grupo. Este método es menos sensible a los valores atípicos (debido al uso de la Mediana) pero es mucho más lento para conjuntos de datos más grandes, ya que se requiere la clasificación en cada iteración al calcular el vector Mediana.
- **Mean-shift clustering** es un algoritmo basado en ventanas deslizantes que intenta encontrar áreas densas de puntos de datos.
- Es un algoritmo basado en centroides, lo que significa que el objetivo es ubicar los puntos centrales de cada grupo/ clase, que funciona actualizando los candidatos para que los puntos centrales sean la media de los puntos dentro de la ventana deslizante.
- Estas ventanas candidatas se filtran en una etapa de postprocesamiento para eliminar los duplicados por cercanía, formando el conjunto final de puntos centrales y sus grupos correspondientes.



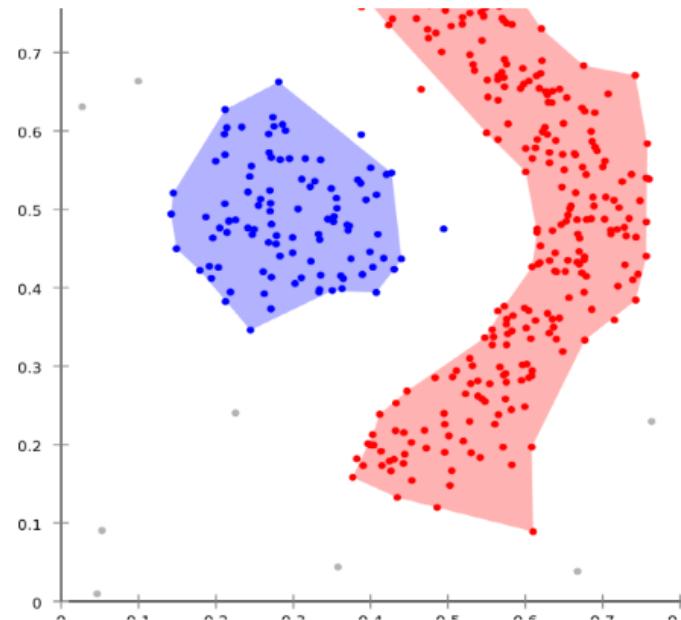
Para explicar el desplazamiento medio, consideramos un conjunto de puntos en un espacio bidimensional como la ilustración anterior.

1. Comenzamos con una ventana deslizante circular centrada en un punto C (seleccionado al azar) y que tiene un radio r como núcleo. El cambio medio es un algoritmo de escalada que implica cambiar este núcleo iterativamente a una región de mayor densidad en cada paso hasta la convergencia.
 2. En cada iteración, la ventana deslizante se desplaza hacia regiones de mayor densidad desplazando el punto central a la media de los puntos dentro de la ventana (de ahí el nombre). La densidad dentro de la ventana deslizante es proporcional al número de puntos dentro de ella. Naturalmente, al cambiar a la media de los puntos en la ventana, se moverá gradualmente hacia áreas de mayor densidad de puntos.
 3. Continuamos desplazando la ventana deslizante de acuerdo con la media hasta que no haya una dirección en la que un desplazamiento pueda acomodar más puntos dentro del núcleo. Mira el gráfico de arriba; seguimos moviendo la ventana circular hasta que ya no aumentamos la densidad (es decir, el número de puntos en la ventana).
 4. Este proceso de los pasos 1 a 3 se realiza con muchas ventanas deslizantes hasta que todos los puntos se encuentran dentro de una ventana. Cuando se superponen varias ventanas deslizantes, se conserva la ventana que contiene la mayoría de los puntos. Los puntos de datos se agrupan de acuerdo con la ventana deslizante en la que residen.
- A diferencia de la agrupación de K-means, no es necesario seleccionar el número de agrupaciones, ya que el cambio de medias los descubre automáticamente. Esa es una gran ventaja.
 - El hecho de que los centros de clúster converjan hacia los puntos de máxima densidad también es bastante deseable, ya que es bastante intuitivo de entender y encaja bien en un sentido natural basado en datos. El inconveniente es que la selección del tamaño / radio de la ventana “ r ” puede no ser trivial.



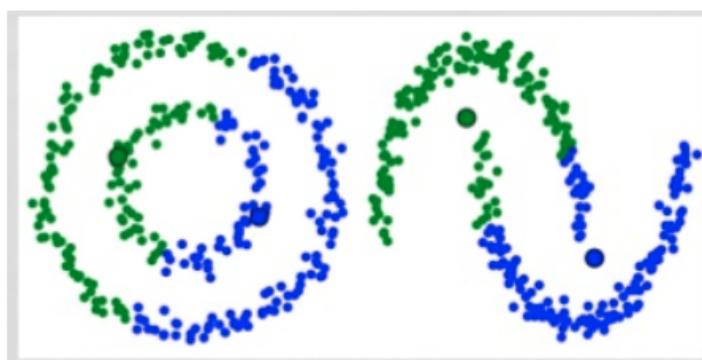
Mean-Shift

- La **Agrupación espacial** basada en densidad de aplicaciones con ruido (DBSCAN) es un algoritmo agrupado basado en densidad similar al Mean Shift, pero con un par de ventajas notables



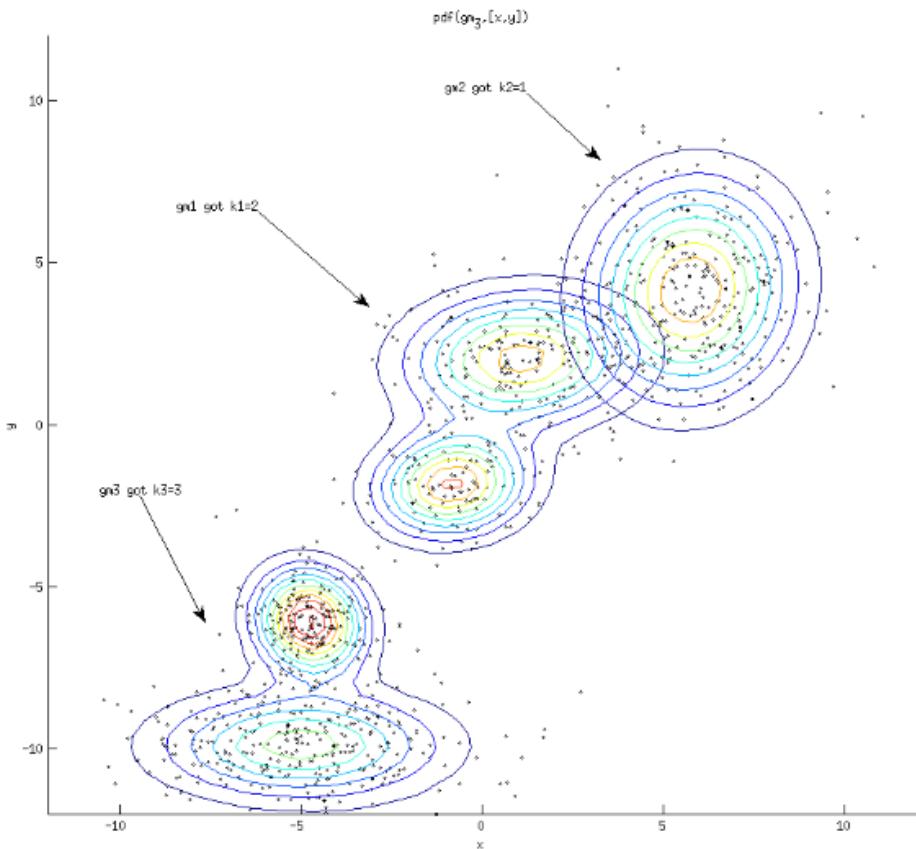
1. DBSCAN comienza con un punto de datos de inicio arbitrario que no ha sido visitado. La vecindad de este punto se extrae usando una distancia ϵ (todos los puntos que están dentro de la distancia ϵ son puntos de vecindad)
2. Si hay un número suficiente de puntos (de acuerdo con el parámetro determinado `minPoints`) dentro de esta vecindad, se inicia el proceso de agrupación y el punto de datos actual se convierte en el primer punto en el nuevo grupo. De lo contrario, el punto se etiquetará como ruido (más tarde este punto ruidoso podría convertirse en parte del clúster). En ambos casos, ese punto está marcado como "visitado".

3. Para este primer punto en el nuevo grupo, los puntos dentro de su vecindario de distancia E también se vuelven parte del mismo grupo. Este procedimiento de hacer que todos los puntos de la vecindad E pertenezcan al mismo grupo se repite para todos los puntos nuevos que se acaban de agregar al grupo de grupos.
 4. Este proceso de los pasos 2 y 3 se repite hasta que se determinan todos los puntos en el grupo, es decir, todos los puntos dentro de la vecindad E el grupo han sido visitados y etiquetados.
 5. Una vez que hayamos terminado con el clúster actual, se recupera y procesa un nuevo punto no visitado, lo que conduce al descubrimiento de un clúster o ruido adicional. Este proceso se repite hasta que todos los puntos estén marcados como visitados. Como al final de este se han visitado todos los puntos, cada punto se ha marcado como perteneciente a un grupo o como ruido.
- DBSCAN presenta algunas grandes ventajas sobre otros algoritmos de agrupamiento:
 - No requiere un número fijo de clústeres en absoluto
 - También identifica valores atípicos como ruidos a diferencia del cambio medio que simplemente los arroja a un clúster incluso si el punto de datos es muy diferente.
 - Es capaz de encontrar grupos de tamaño y formas arbitrarias bastante bien.
 - El principal inconveniente de DBSCAN es que no funciona tan bien como otros algoritmos cuando los grupos son de densidad variable. Esto se debe a que la configuración del umbral de distancia E y minPoints para identificar los puntos vecinos variará de un grupo a otro cuando la densidad varíe. Este inconveniente también ocurre con datos de muy alta dimensión (muchas dimensiones), ya que nuevamente el umbral de distancia E se vuelve difícil de estimar.
 - Uno de los principales inconvenientes de K-means es su ingenuo (Naive) del valor medio para el centro de clúster. Podemos ver por qué esta no es la mejor manera de hacer las cosas con la siguiente imagen



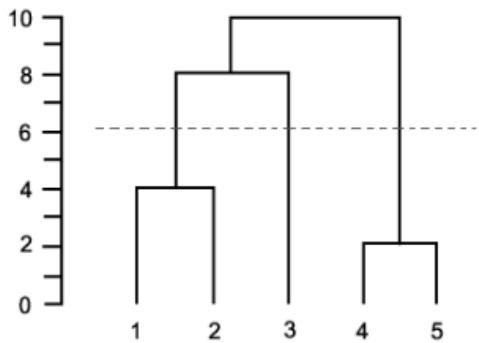
En el lado izquierdo parece bastante obvio para el ojo humano que hay dos grupos circulares con diferentes radios centrados en la misma media. K-means no puede manejar esto porque los valores medios de los grupos están muy juntos. K-means también falla en los casos en que los grupos no son circulares, nuevamente como resultado del uso de la media como centro de grupo.

- Los **modelos de mezcla gaussiana**(GMM) nos brindan más flexibilidad que K-means. Con los GMM asumimos que los puntos de datos están distribuidos en Gauss. Esta es una suposición menos restrictiva que decir que son circulares usando la media. De esa manera, tenemos dos parámetros para describir la forma de los grupos: la media y la desviación estándar.
- Tomando un ejemplo en dos dimensiones, esto significa que los grupos pueden tomar cualquier tipo de forma elíptica (ya que tenemos un desviación estándar en las direcciones x e y). Por lo tanto, cada distribución gaussiana se asigna a un solo grupo.
- Para encontrar los parámetros de Gauss para cada grupo (por ejemplo, la media y la desviación estándar) usaremos un algoritmo de optimización llamado Expectation - Maximization (EM). En la siguiente imagen tenemos una ilustración de los gaussianos que se están ajustando a los grupos. Luego podemos continuar con el proceso de Agrupación de Expectación - Maximización usando GMM.



1. Comenzamos seleccionando el número de grupos (como lo hace K-means) e inicializando aleatoriamente los parámetros de distribución gaussianos para cada grupo. Es posible tratar de proporcionar una buena estimación de los parámetros iniciales echando un vistazo rápido a los datos también. Sin embargo, esto no es siempre necesario ya que los gaussianos comienzan siendo muy pobres, pero se optimizan rápidamente.

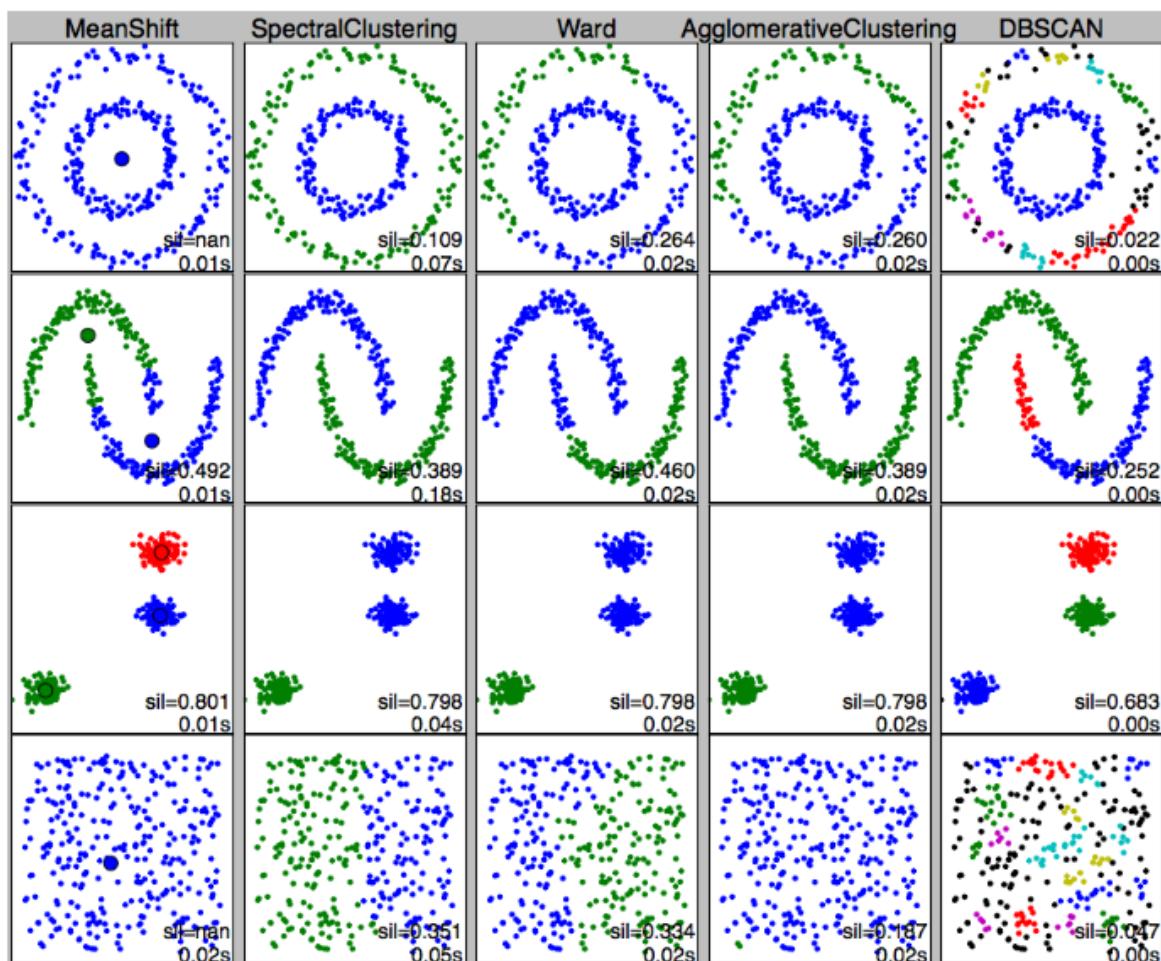
2. Dadas estas distribuciones gaussianas para cada grupo, se calcula la probabilidad de que cada punto de datos pertenezca a un grupo particular. Cuanto más cerca esté un punto del centro de Gauss, más probable es que pertenezca a ese grupo. Esto debería tener sentido intuitivo ya que con una distribución gaussiana estamos asumiendo que la mayoría de los datos se encuentran más cerca del centro del clúster.
 3. En base a estas probabilidades, calculamos un nuevo conjunto de parámetros para las distribuciones gaussianas de modo que minimicemos las probabilidades de los puntos de datos dentro de los grupos. Calculamos estos nuevos parámetros usando una suma ponderada de las posiciones de los puntos de datos que pertenecen a ese grupo particular.
 4. Los pasos 2 y 3 se repiten iterativamente hasta la convergencia, donde las distribuciones no cambian mucho de iteración a iteración.
- Hay dos ventajas clave para usar GMM:
 - Los GMM son mucho más flexibles en términos de covarianza de conglomerados que K-means; debido al parámetro de desviación estándar, los grupos pueden adoptar cualquier forma de elipse, en lugar de estar restringidos a círculos. K-means es en realidad un caso especial de GMM en el que la covarianza de cada grupo a lo largo de todas las dimensiones se aproxima a 0.
 - Dado que los GMM usan probabilidades, pueden tener múltiples grupos por punto de datos. Entonces, si un punto de datos se encuentra en el medio de dos grupos superpuestos, simplemente podemos definir su clase diciendo que pertenece al X-percent a la clase 1 y al Y-percent a la clase 2. Es decir, los GMM admiten **asignaciones de grupos mixtas**.
 - Los algoritmos de agrupamiento jerárquico en realidad se dividen en 2 categorías: de arriba hacia abajo o de abajo hacia arriba. Los algoritmos ascendentes tratan cada punto de datos como un solo grupo al principio y luego combinan (o aglomeran) sucesivamente pares de grupos hasta que todos los grupos se hayan fusionado en un solo grupo que contiene todos los puntos de datos. Por lo tanto, el agrupamiento jerárquico ascendente se denomina agrupamiento jerárquico aglomerativo o HAC.
 - Esta jerarquía de grupos se representa como un árbol (o dendrograma). La raíz del árbol es el grupo único que reúne todas las muestras, siendo las hojas los grupos con una sola muestra. Consulte el gráfico a continuación para ver una ilustración antes de pasar los pasos del algoritmo.



1. Comenzamos tratando cada punto de datos como un solo grupo, es decir, si hay X puntos de datos en nuestro conjunto de datos, entonces tenemos X grupos. Luego seleccionamos una métrica de distancia que mide la distancia entre grupos. Como ejemplo, utilizaremos un enlace promedio que define la distancia entre dos grupos para que sea la distancia promedio entre los puntos de datos en el primer grupo y los puntos de datos en el segundo grupo.
 2. En cada iteración combinamos dos grupos en uno. Los dos grupos que se combinarán se seleccionan como aquellos con el enlace promedio más pequeño. Es decir, según nuestra métrica de distancia seleccionada, estos dos grupos tienen la menor distancia entre sí y, por lo tanto, son los más similares y deben combinarse.
 3. El paso 2 se repite hasta llegar a la raíz del árbol, es decir, solo tenemos un grupo que contiene todos los puntos de datos. De esta manera, podemos seleccionar cuántos grupos queremos al final, simplemente eligiendo cuándo dejar de combinar los grupos, es decir, cuando dejamos de construir el árbol.
- La agrupación jerárquica no requiere que especifiquemos la cantidad de agrupaciones e incluso podemos seleccionar qué cantidad de agrupaciones se ve mejor ya que estamos construyendo un árbol.
 - Además, el algoritmo no es sensible a la elección de la métrica de distancia; todos ellos tienden a funcionar igualmente bien, mientras que con otros algoritmos de agrupamiento, la elección de la métrica de distancia es crítica.
 - Un caso de uso particularmente bueno de los métodos de agrupamiento jerárquico es cuando los datos subyacentes tienen una estructura jerárquica y desea recuperar la jerarquía; otros algoritmos de agrupamiento no pueden hacer esto.
 - Estas ventajas del agrupamiento jerárquico tienen el coste de una menor eficiencia, ya que tiene una complejidad temporal de $O(n^3)$, a diferencia de la complejidad lineal de K-means y GMM.

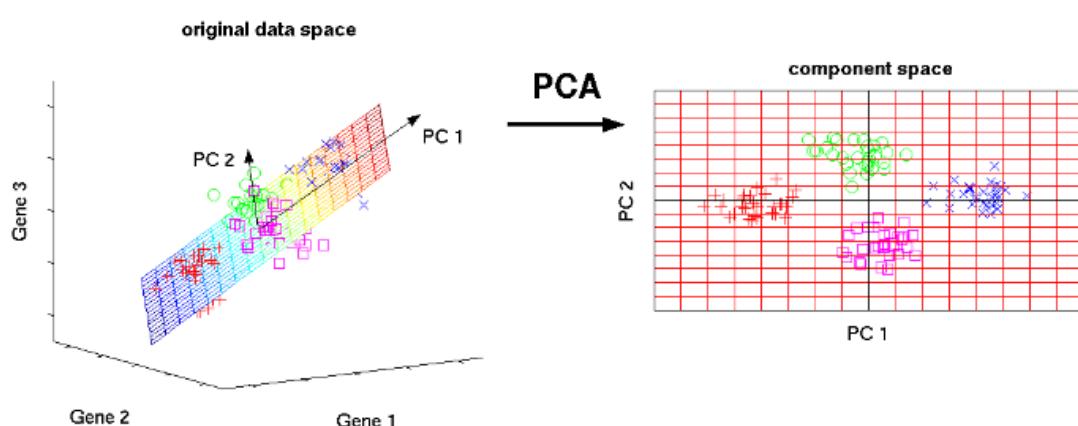
COMPARATIVA FINAL

- A continuación vemos cómo los diferentes algoritmos se comparan y contrastan con diferentes datos.



ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

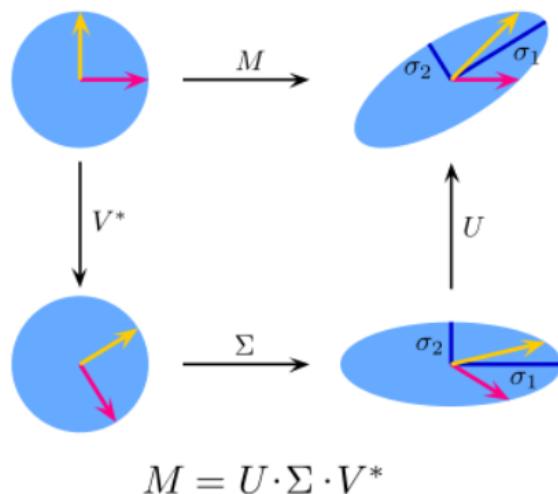
- Es un procedimiento estadístico que utiliza una transformación ortogonal para convertir un conjunto de observaciones de variables posiblemente correlacionadas en un conjunto de valores de variables no correlacionadas linealmente llamadas componentes principales.



- Algunas de las aplicaciones de PCA incluyen compresión y simplificación de datos para facilitar el aprendizaje y la visualización. Hay que tener en cuenta que el conocimiento del dominio es muy importante al elegir si nos interesa aplicar PCA o no. No es adecuado en casos donde los datos son ruidosos, porque todos los componentes de PCA tienen una variación bastante alta.

DESCOMPOSICIÓN DE VALOR SINGULAR (SVD)

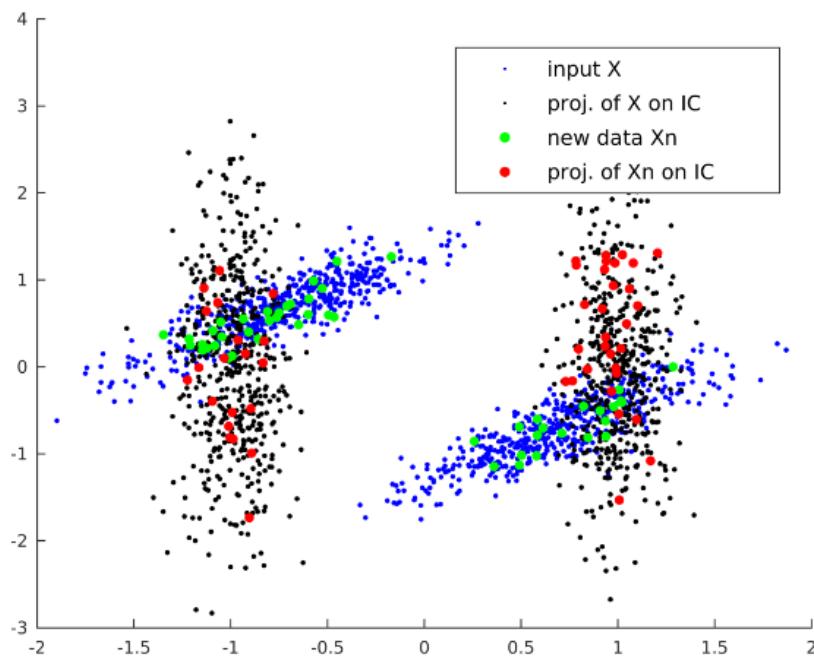
- En álgebra lineal, SVD es una factorización de una matriz compleja real. Para una matriz $m \times n$ dada, existe una descomposición tal que $M = U\Sigma V^*$, donde U y V son matrices unitarias y Σ es una matriz diagonal.



- PCA es en realidad una aplicación simple de SVD. en la visión por computadora, los algoritmos de reconocimiento de la primera cara utilizaban PCA y SVD para representar caras como una combinación lineal de “caras propias”
- Después se llevaba a cabo una reducción de dimensiones y se hacían coincidir las caras con las identidades mediante métodos simples. Aunque los métodos modernos son mucho más sofisticados, muchos aún dependen de técnicas similares.

ANÁLISIS DE COMPONENTES INDEPENDIENTES (ICA)

- Es una técnica estadística para revelar factores ocultos que subyacen en conjunto de variables aleatorias, medidas o señales. ICA define un modelo generativo para los datos multivariados observados, que generalmente se proporciona como una gran base de datos de muestras.
- En el modelo, se supone que las variables de datos son mezclas lineales de algunas variables latentes desconocidas, el sistema de mezcla también es desconocido. Se supone que las variables latentes no son gaussianas y son mutuamente independientes, y se denominan componentes independientes de los datos observados.



- ICA está relacionado con PCA, pero es una técnica mucho más poderosa que es capaz de encontrar los factores subyacentes de las fuentes cuando los métodos clásicos fallan.
- Sus aplicaciones incluyen imágenes digitales, bases de datos de documentos, indicadores económicos y mediciones psicométricas.
- A continuación vamos a ver la aplicación práctica de estos algoritmos y otros, para implementar una serie de técnicas muy utilizadas en IA.

INDETERMINACIONES DE ICA

1. No podemos determinar las **varianzas** (energías) de las componentes independientes.

Esto se debe a que al ser A y s desconocidas, cualquier escalar ui que multiplique a una de las fuentes si podría cancelarse dividiendo la columna correspondiente, ai , de A por el mismo escalar ui :

$$x = \sum_{i=1}^n \left(\frac{1}{\mu_i} a_i \right) (s_i \mu_i)$$

Fórmula ICA

2. No se puede determinar el orden de los componentes independientes debido a que **A** y **s** son desconocidas y se puede modificar fácilmente el orden de los términos del sumatorio. Los elementos de **Ps** son las variables independientes, **si**, originales, pero en un orden diferente. La matriz **AP-1** sería una nueva matriz de mezcla que se obtiene aplicando la técnica ICA.

$$x = AP^{-1}Ps$$

Componentes independientes

1. Independencia estadística

Es la idea principal para la construcción del método, ya que queremos encontrar dentro de nuestras señales mezcladas aquellas que son más independientes (estadísticamente) respecto a las otras.

2. La matriz A debe ser cuadrada

Es necesario que la matriz A sea cuadrada y de rango completo, es decir, $si = xi$.

3. Se consigue que el experimento esté libre de ruido

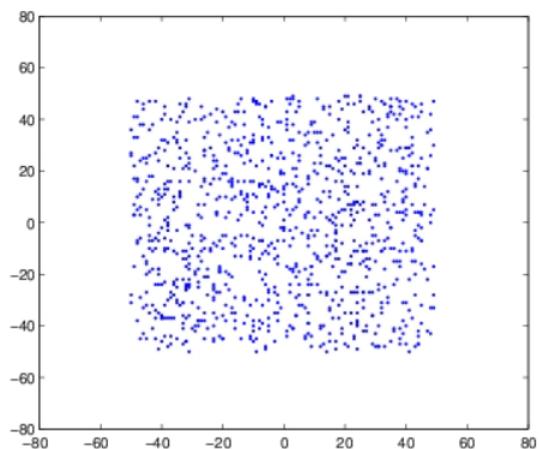
Todas las señales que se muestran de las **señales mezcladas x** deben tener únicamente combinaciones lineales de aquellas señales independientes s, no debe haber información derivada de ruido externo, debido a este ruido podría ser interpretado como otra señal independiente.

4. Los datos deben estar centrados

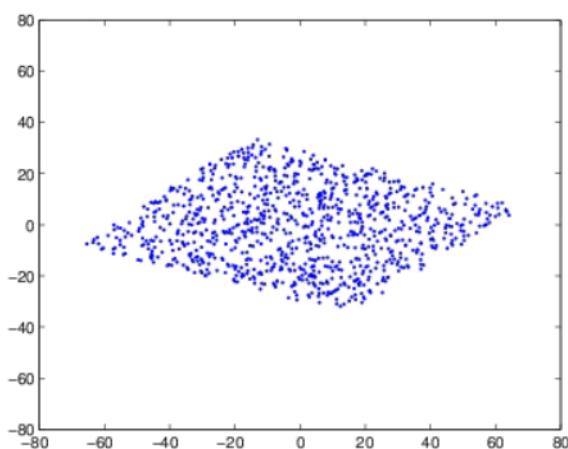
Los datos ingresados al método ICA deben ser centrados, es decir, **su media** debe ser **cero**.

5. Función de densidad de probabilidad no Gaussiana

Las señales fuente independiente deben tener **función de densidad de probabilidad no Gaussiana**, ya que así se garantiza que los componentes independientes se pueden separar efectivamente.



Señales independientes con fdp no gaussiana.



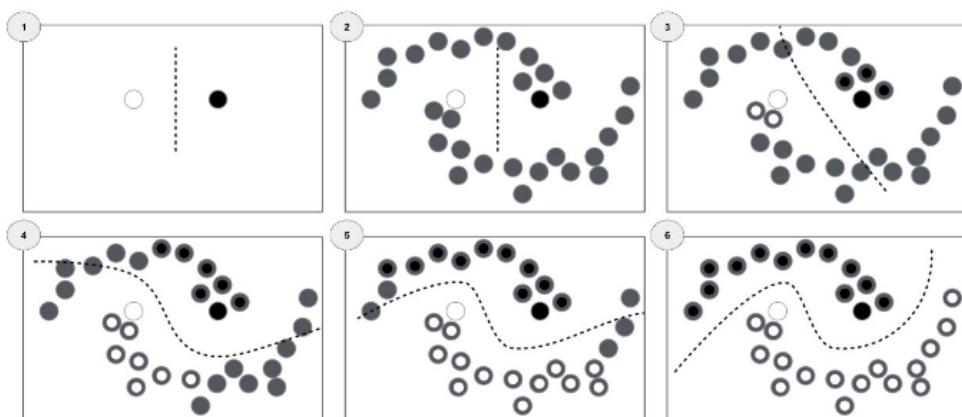
Señales mezcladas

MODELO SEMI-SUPERVISADO

- A diferencia del aprendizaje supervisado y el no supervisado, los algoritmos semi-supervisados utilizan pocos datos etiquetados y muchos datos **no** etiquetados como parte del conjunto de entrenamiento.
- Estos algoritmos semi-supervisados intentan buscar la información estructural que contienen los datos no etiquetados con el objetivo de crear modelos predictivos que funcionen mejor que aquellos que solo utilizan datos etiquetados.cada vez son más utilizados los modelos de aprendizaje semi-supervisados en nuestro día a día.

Un ejemplo de dichos modelos es el análisis de las conversaciones grabadas en un call center. Con el objetivo de inferir automáticamente características de los interlocutores (género, edad, geografía,...), sus estados de ánimo (contentos, enfadados, sorprendidos,...), los motivos de la llamada (error en la factura, nivel de servicio, problemas de calidad,...), entre otros, es necesario disponer de un volumen elevado de casos ya **etiquetados** sobre las cuales aprender los patrones de cada tipología de llamadas.

- En el enfoque semi-supervisado se asume cierta estructura en la distribución subyacente de los datos, es decir, los datos más próximos entre sí se supone que tienen la misma etiqueta.
- Por lo tanto, en el aprendizaje semi-supervisado lo que tenemos son tanto datos etiquetados como datos no etiquetados, es decir, además de tener tuplas (X, Y) , tenemos datos sólo de X de los que no sabemos su respuesta Y .
- El objetivo está en combinar datos etiquetas y no etiquetados para construir un modelo supervisado que sea mayor ya que: la cantidad de datos etiquetados se puede aumentar, lo que produce una mejora de los resultados de los modelos. Esto supone un alto coste de etiquetar los datos X sin etiquetas.
- En el enfoque semi-supervisado se asume cierta estructura en la distribución subyacente de los datos, es decir, los datos más próximos entre sí se supone que tienen la misma etiqueta.



Aprendizaje semisupervisado

PROCESO DEL APRENDIZAJE SEMI-SUPERVISADO

- Recopilar los pocos datos históricos que tengan resultados disponibles.
 - Recopilar los datos históricos sin resultados. Evaluar la posibilidad de etiquetar manualmente más datos históricos.
 - Instruir a la IA a que aprenda utilizando aprendizaje supervisado los datos históricos para los que tenemos resultados.
 - Usar el modelo de machine learning aprendido para etiquetar automáticamente el resto de los datos.
 - Usar otro modelo de machine learning supervisado con los datos etiquetados inicialmente y los datos etiquetados automáticamente.
-
- Entre los **métodos de** aprendizaje semisupervisado se encuentran:
 - Self-training
 - Co-training
 - Assemble
 - Re-weighting

SELF-TRAINING

- Es el método más simple y más utilizado
- Este algoritmo consta de los siguientes **pasos**:
 1. Se construye un clasificador con los ejemplos etiquetados.
 2. Mediante dicho clasificador, se clasifican todos los ejemplos no etiquetados.
 3. Se añaden al conjunto de ejemplos etiquetados los nuevos ejemplos clasificados que tengan un mayor índice de confianza.
 4. Se repite el procedimiento hasta que solo quede un cierto porcentaje de ejemplos no etiquetados, o se llegue al número máximo de iteraciones.

PROCESO DE SELF-TRAINING

L = (x_i , y_i); ejemplos etiquetados

U = (x_i , ?); ejemplos no etiquetados

T; umbral de confianza

While **U** \neq ; or **U`** \neq ; **do**

Entrena un clasificador **C** con **L**

Clasifica **U** con **C**

Encuentra un subconjunto **U`** de **U** con alto nivel de confianza (confianza > **T**)

L + U` = L

U - U` = U

end while

CO-TRAINING

- El co-training supone que existen dos conjuntos de **atributos independientes y compatibles** para los datos (dos vistas de los datos). Con cada conjunto (o vista) de atributo es más que suficiente para aprender un clasificador adecuado.
- Se aprende un clasificador con cada subconjunto de atributos (redundantes) y se usa para clasificar datos para el otro clasificador y aumentar, así, su conjunto de entrenamiento. Este método es parecido al self-training, con la diferencia de que en este caso hay **dos clasificadores** a tener en cuenta, y el umbral de confianza se suele ser mayor (en torno al 95%)

Algoritmo *Democratic Co-learning*:

Data: $Z_l = \{(X_l, Y_l)\}$; ejemplos etiquetados.

X_u ; ejemplos no etiquetados.

$Z_u = \{((0, \dots, 0), 0)\}$; conjunto auxiliar.

begin

while $X_u \neq \emptyset$ and $Z_u \neq \emptyset$ **do**

 Construir k clasificadores distintos C_1, \dots, C_k con Z_l ;

 Clasificar X_u con C_1, \dots, C_k ; $\Rightarrow Y_u^1, \dots, Y_u^k$;

 Separar los elementos de Y_u^1, \dots, Y_u^k que se hayan clasificado con mayoría absoluta en la misma clase, y que además la suma de los índices de confianza de dicha clase sea mayor que la suma de los de las restantes clases, que llamaremos Y'_u (que toman dicha clase como valor); con sus respectivos valores de X_u , que llamaremos X'_u ; $\Rightarrow Z_u = \{(X'_u, Y'_u)\}$;

 Modificar $Z_l = Z_l \cup Z_u$; $X_u = X_u \setminus X'_u$;

end

end

Result: $Z_l = Z_l \cup (X_u, Y_u^*)$, donde Y_u^* son los elementos resultantes de Y_u^1, \dots, Y_u^k que no tienen una mayoría absoluta de una única clase. Estos elementos son pronosticados con un regla de selección, como por ejemplo *weighted majority voting*

.....
Proceso de Co-learning

ASSEMBLE

- El objetivo de ASSEMBLE(Adaptive semi-supervised enSEMBLE) es crear un ensamble de clasificadores que trabaje en forma consistente tanto con ejemplos etiquetados como no etiquetados. La finalidad es maximizar el margen (margin) tanto con ejemplos etiquetados como no etiquetados. Para esto, se introduce el concepto de **pseudo-clase** para los ejemplos no etiquetados.
- La pseudo-clase de un ejemplo no etiquetado es la **clase predicha** por el ensamble hasta ese momento: $y_{xi} = F(x_i)$ para $x_i \in U$.
- Las pseudo-clases para los datos iniciales se pueden obtener usando vecinos más cercanos o clase mayoritaria en los ejemplos etiquetados, al inicio se les da más peso a los ejemplos etiquetados.

Sea $I = |L|$ y $u = |U|$ y

$$D_1(i) = \begin{cases} \beta/I & \text{si } i \in L \\ (1 - \beta)/u & \text{si } i \in U \end{cases}$$

Sea $y_i = c$ { c es la clase del vecino más cercano en L a $i \in U$ }

Sea $f_1 = \mathcal{L}(L + U, Y, D_1)$ { \mathcal{L} es el algoritmo de aprendizaje}

for $t = 1$ to T **do**

 Sea $\hat{y}_i = f_t(x_i), i = 1, \dots, I + u$ {clasificación con el ensamble actual}

$\epsilon = \sum_i D_t[y_i \neq \hat{y}_i], i = 1, \dots, I + u$ {errores pesados}

if $\epsilon > 0.5$ **then**

 Stop

end if

$w_t = 0.5 * \log(\frac{1-\epsilon}{\epsilon})$ {peso para el clasificador}

Algoritmo Assemble

Sea $F_t = F_{t-1} + w_t f_t$ {nuevo ensamble}

Sea $y_i = F_t(x_i)$ si $i \in U$ {posiblemente nueva clasificación para los ejemplos no etiquetados con el nuevo ensamble} {cálculo de nuevos pesos a los ejemplos}

$$D_{t+1} = \frac{\alpha e^{-y_i F_{t+1}(x_i)}}{\sum_j \alpha_j e^{-y_j F_{t+1}(x_j)}} \text{ para toda } i$$

$S = \text{Muestrea}(L + U, I, D_{t+1})$ {muestreo nuevo de datos}

Algoritmo Assemble

RE-WEIGHTING

- Se aprende un modelo con los ejemplos etiquetados y se aplica a los ejemplos no etiquetados.
- Los ejemplos etiquetados y no etiquetados se agrupan de acuerdo al valor de probabilidad de que se tenga esa clase.
- Los ejemplos no etiquetados en cada grupo se etiquetan de acuerdo a la distribución de las clases de los ejemplos etiquetados en ese grupo.
- Por ejemplo, supongamos que tenemos la siguiente tabla:

Grupo	No Etiq.	Etiq.	Clase 0	Clase 1
0.6 - 0.7	20	100	10	90

Lo que nos dice es que en el grupo tienen probabilidad entre 0.6 y 0.7 se encuentran 20 ejemplos no etiquetados y 100 etiquetados, de los cuales 10 son de clase 0 y 90 de clase1

El peso del grupo se define como:

$$|L + U| / |L| = 120 / 100 = 1.2$$

Se aplica este peso a cada clase: NClase0

$$= 1.2 * 10 = 12 \text{ y } NClase1 = 1.2 * 90 = 108$$

Aprendizaje por refuerzo

- El Reinforcement Learning intentará hacer aprender a la máquina basándose en un esquema de “**premios y castigos**” en un entorno en donde hay que tomar acciones y que está afectado por múltiples variables que varían con el tiempo.



CONCEPTO

- En los últimos años, los avances que se han producido han permitido la aparición de nuevos modelos a través de los cuales se han conseguido importantes avances y evoluciones en el campo del Machine Learning. Sin lugar a dudas, uno de los avances más importantes lo representa el conocido como **aprendizaje por refuerzo** (Deep Reinforcement Learning)
- Puede considerarse que el aprendizaje por refuerzo aparece como una nueva generación dentro de las técnicas que son propias del Machine Learning.
- No en vano, a través del Deep Reinforcement Learning la máquina acaba aprendiendo nuevas tareas de una forma más evolucionada. No obstante, lo cierto es que los primeros trabajos en relación con este ámbito pueden remontarse a la década de 1960.

- En los últimos tiempos los avances propios del aprendizaje profundo han motivado un notable auge en este proceso de aprendizaje profundo por refuerzo. Y es que, estos avances, han propiciado que los sistemas basados en el Deep Reinforcement Learning sean capaces de resolver problemas con una mayor complejidad.
- Resulta, en ese sentido y para terminar de hacerse una idea de cómo ha sido la evolución seguida hasta llegar a este punto, repasar los avances históricos en los diferentes campos:

1. Sistemas programables

Son aquellas que se encargan de resolver problemas técnicos con un elevado grado de concreción. Debido a ello, tienen que ser programados de forma explícita.

2. Sistemas expertos

Son aquellos que se encuentran basados en reglas fijas que permiten trasladar los conocimientos de un experto en un aspecto concreto y determinado.

3. Sistemas de aprendizaje automático

Son aquellos que, de un modo automático, acaban aprendiendo reglas de carácter decisivo bastante complejas, a través del análisis estadístico de los datos.

4. Sistemas de aprendizaje por refuerzo

Son aquellos que se caracterizan por explorar y por adquirir los datos fundamentales referentes a un problema por iniciativa propia, a través del diseño automático de diversas estrategias que permitan darle solución.

- Así las cosas, puede considerarse que un sistema de aprendizaje profundo por refuerzo se encuentra basado en una máquina inteligente que es capaz de optimizar los principales aspectos relacionados con los procesos de decisión que se le presentan.
- Todo ello, además, teniendo en cuenta el resultado final de dicha toma de decisiones:
 - Si el proceso de toma de decisiones ha acabado resultando **beneficioso** para sus intereses, de forma automática la máquina aprenderá a tomar dicha decisión en futuros casos.
 - Si el proceso de toma de decisiones ha acabado resultando **negativo** para sus intereses, la máquina tratará de no volver a tomar dicha decisión en el futuro más cercano.
- Puede considerarse, además, que el aprendizaje profundo por refuerzo presenta procesos que guardan muchísimas similitudes con los modos de aprendizaje condicionado de los seres vivos. No en vano, a través de este tipo de procedimientos, la máquina es capaz de tomar aquellas decisiones que resultan más idóneas y adecuadas a su situación concreta, pudiendo desarrollar estrategias a largo plazo que permitan la plena optimización de los beneficios que pueda acabar obteniendo.
- Finalmente, es necesario indicar que las capacidades de aprendizaje de la máquina se encuentran fundamentadas en modelos de aprendizaje profundo o de redes neuronales profundas. Una circunstancia que, además, permite conseguir

importantes avances en aspectos referentes al análisis de datos fruto de imágenes, de sonidos o del propio lenguaje natural.

Los videojuegos suelen ser ejemplos del uso de RL porque son un entorno YA programado en el que se está simulando un ambiente y en el que ocurren eventos a la vez. Por lo general el jugador es el agente que debe decidir qué movimientos hacer.

- En el aprendizaje por refuerzo no tenemos una “etiqueta de salida”, por lo que no es de tipo supervisado y si bien estos algoritmos aprenden por sí mismos, tampoco son de tipo no supervisado, en donde se intenta clasificar grupos teniendo en cuenta alguna distancia entre muestras.
- En el mundo real contamos con múltiples variables que por lo general se interrelacionan y que dependen de otros casos de negocio y dan lugar a escenarios más grandes en donde tomar decisiones. Para conducir un coche no basta una inteligencia que pueda detectar un semáforo en rojo, verde ó amarillo; tendremos muchísimos factores - todos a la vez - a los que prestar atención: a qué velocidad vamos, estamos ante una curva?, hay peatones?, es de noche y debemos encender las luces?.
- Para realizar estas tareas con un algoritmo de aprendizaje supervisado deberíamos tener varias máquinas interactuando entre ellas. Con un enfoque de aprendizaje por refuerzo tenemos una alternativa más eficiente y fácil de integrar. Se intenta hacer aprender al sistema basándose en una interacción de premios y castigos. El objetivo ahora es maximizar la recompensa que se obtiene, asumiendo algunos errores o soluciones que no sean óptimas del todo.

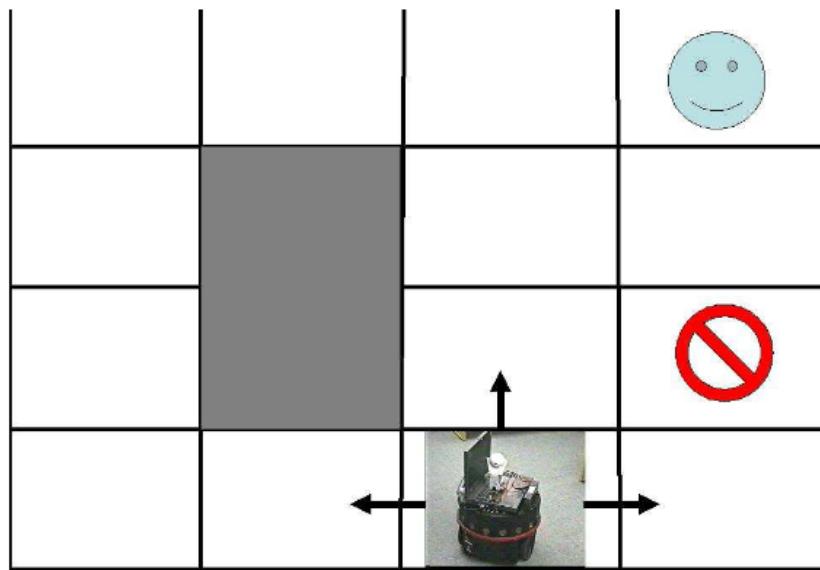
ELEMENTOS QUE COMPONEN UN MODELO DE APRENDIZAJE POR REFUERZO

- El aprendizaje por refuerzo es un modelo que funciona a partir de la interacción de unos componentes concretos, a través de la realización de diversas acciones. Estas acciones que acaban desarrollándose tienen un marcado carácter de retroalimentación, viéndose conectado por una serie de elementos determinados.
- En relación con los principales **componentes** que dan forma y que se relacionan dentro de los sistemas de aprendizaje profundo por refuerzo, es posible llegar a hablar tanto del agente como del ambiente.
- Por un lado, se encontraría el denominado como **agente**. Este no es más que el modelo o la máquina con la que se pretende trabajar. Se deberá trabajar y entrenar al agente, para que pueda aprender a tomar decisiones de la forma más correcta posible.
- Por otro lado, es necesario mencionar al **ambiente**. Este factor puede llegar a ser definido, de forma sencilla y rápida, como el entorno en el cual debe llegar a desempeñar el agente. Se trata de un elemento de gran trascendencia en los procesos comportamentales y de aprendizaje de este, ya que en el ambiente se encuentran inscritas las limitaciones y las reglas a las que deberá adaptarse.

- Asimismo, es necesario hacer hincapié en un hecho claro y de gran trascendencia: tanto el agente como el ambiente acaban interactuando entre sí. Se produce una retroalimentación entre ambos, a través de una serie de aspectos
- Los más destacados son:
 - **La acción**
Son las posibles decisiones que, en un momento concreto, debe tomar el agente ante un tipo de situación
 - **El estado**
Se trata de un factor que se encuentra estrechamente relacionado con el ambiente. De este modo, este concepto hace alusión a aquellos indicadores que sirven para hacerse una idea del estado del ambiente ante cada situación.
 - **Las recompensas o los castigos**
Son las consecuencias finales ante las decisiones del agente en sus interacciones con el ambiente. De este modo, a través de la utilización de recompensas y de castigos, se podrá orientar al agente sobre si su decisión ha sido la idónea o no.
- De este modo, es posible sintetizar el modelo básico de funcionamiento de aprendizaje por refuerzo a través del conocido como **Proceso de Decisión de Markov**.
- Explicando esto de un modo conciso, podría secuenciar del siguiente modo:
 1. El agente recibe un estado inicial, llevando a cabo una acción con la que acaba interviniendo en el ambiente.
 2. La acción tomada por el agente acabará generando, consecuentemente, cambios en el ambiente.
 3. Estos cambios acabarán propiciando, en las siguientes interacciones entre el agente y el ambiente, la presencia de recompensas tanto de tipo positivo como de tipo negativo.
 4. En el caso en el que la recompensa sea positiva, el agente verá reforzado ese comportamiento o esa toma de decisiones de cara al futuro.
 5. En el caso en el que la recompensa sea negativa, el agente sufrirá una penalización, que lo acabará forzando a tomar decisiones distintas ante esa misma situación.

EJEMPLO DE UN MDP

- Supongamos que en un mundo virtual en forma de rejilla hay un agente que se desplaza de un cuadro a otro, pudiendo moverse a los cuadros continuos: **arriba, abajo, derecha, izquierda**.
- El agente desea desplazarse para llegar a cierta meta (recompensa positiva), evitando los obstáculos en el camino (recompensa negativa). Dado que cada movimiento implica cierto gasto de energía para el agente, cada movimiento tiene asociado cierto costo.
- Como existe incertidumbre en las acciones del agente, asumimos que con un 80% de probabilidad llega a la celda deseada, y con un 20% de probabilidad cae en otra celda contigua.



Ejemplo de un MDP

- Por ejemplo en un mundo virtual en que el agente debe navegar, la celda de arriba a la derecha(S4) representa la meta, con una recompensa positiva; mientras que las celdas grises y la celda con el signo de prohibido (peligro), tienen recompensas negativas
- El agente (en dibujo en el estado S15) puede desplazarse a las 4 celdas vecinas, excepto si está en la orilla del mundo virtual.
- Consideramos un mundo de 4x4 celdas, podemos definir un MDP que representa este problema:

Estados:

$$S = \left\{ \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \\ s_5 & s_6 & s_7 & s_8 \\ s_9 & s_{10} & s_{11} & s_{12} \\ s_{13} & s_{14} & s_{15} & s_{16} \end{array} \right\}$$

representación del problema

Acciones: $A = \{\text{izquierda, arriba, derecha, abajo}\}$

Recompensas:

$$R = \left\{ \begin{array}{cccc} -1 & -1 & -1 & 10 \\ -1 & -100 & -1 & -1 \\ -1 & -100 & -1 & -10 \\ -1 & -1 & -1 & -1 \end{array} \right\}$$

Recompensas del problema

- La meta tiene una recompensa positiva (+10), los obstáculos (-100) y la zona de peligro (-10) negativas. Las demás celdas tienen una pequeña recompensa negativa (-1) que indican el costo de moverse del agente, de forma que lo hacen buscar una trayectoria “corta” a la meta.

FUNCIONAMIENTO

- Para conseguir el correcto funcionamiento de un sistema de aprendizaje profundo por refuerzo, será necesario tener en consideración diversos aspectos que son fundamentales para que el agente intervenga con el ambiente, generando nuevos estados.
- Entre los principales aspectos que inciden en el funcionamiento de este tipo de modelos, será necesario tener en consideración aspectos como:
 - Recompensas y penalizaciones
 - Algoritmos a utilizar
 - La ecuación de Bellman

RECOMPENSAS VS. PENALIZACIONES

- Uno de los primeros aspectos que deben tenerse en cuenta es que, en sus primeros pasos, el agente estará completamente en blanco. Esto es, no tendrá conocimiento alguno sobre qué es lo que debe hacer ante cada tipo de situación, no sobre cómo tendrá que comportarse. Este tipo de conocimientos los acabará adquiriendo a través del entrenamiento al que sea sometido a lo largo de sus diferentes interacciones con el ambiente.
- Fruto de todo ello, muchos expertos en la materia han concluido que, en esos primeros pasos, la toma de decisiones por parte del agente ante situaciones concretas será aleatoria. En base a las indicaciones que reciba sobre si dicha decisión ha sido correcta o incorrecta, irá puliendo su forma de actuar: una optimización de recursos. Es, en estos casos, donde entran en juego de forma decisiva tanto las recompensas como las penalizaciones.
- De forma genérica, pueden considerarse como recompensas todos aquellos estímulos positivos que sirven como refuerzo ante un comportamiento concreto y

determinado del individuo. En el caso concreto de los agentes, las recompensas son simples scores o puntuaciones numéricas.

- Así las cosas, deberá prefijar con una puntuación determinada cualquier posible decisión que tome el agente. Todo ello, teniendo además siempre en consideración el conocido como **dilema de exploración y explotación**: esto es, habrá que evitar que el algoritmo acabe estancándose, buscando siempre la decisión que sabe que le va a reportar la gratificación y la recompensa inmediata ante cualquier tipo de situación.
- Se trata, en definitiva, de conseguir que el agente encuentre el equilibrio entre la explotación de los recursos de los que dispone y que conoce que son seguros y entre la exploración del ambiente.
- La consecuencia más directa de la correcta resolución de este importante dilema por parte del agente no será otra que conseguir que este explore el ambiente, aprendiendo de forma sucesiva cuáles son los modos comportamentales más aptos para conseguir las recompensas y para evitar las posibles penalizaciones.
- Todo ese conocimiento adquirido por parte del agente, a través de sus diferentes interacciones con el ambiente, se almacenarán en las denominadas como **políticas**. Se trata, este último, de un aspecto de gran importancia: no en vano, de la creación de unas buenas políticas dependerá, directamente, la posibilidad de que se acabe disponiendo de un buen agente.
- Este camino, sin embargo, no es nada sencillo: requerirá de horas y horas de entrenamiento por parte del agente, de modo que acumulen tantos aciertos y errores como sean necesarios para conseguir unas políticas fuertes.

ALGORITMOS A UTILIZAR

- Otro de los aspectos determinantes para conseguir el correcto funcionamiento de los modelos de aprendizaje profundo por refuerzo son los **algoritmos** a utilizar en el proceso.
- Son diversos los algoritmos que pueden llegar a utilizarse para la correcta implementación de los modelos basados en el aprendizaje profundo por refuerzo. Sin embargo, uno de los más utilizados y recomendados para la realización de este tipo de sistemas es el denominado como **Q-Learning**.
- Este tipo de algoritmo se encuentra íntimamente relacionado con las políticas, ya mencionadas en el apartado anterior. No en vano, el término "Q" pasará a identificarse directamente con las políticas: de este modo, este valor acabará generando la puntuación de una política concreta ante una acción y un estado concreto.
- El algoritmo Q-Learning intenta aprender cuánta **recompensa** obtendrá a largo plazo para cada pareja de **estado** y **acciones** (s,a). A la función mencionada se le denomina como la función de acción - valor representado como $Q(s,a)$.
- Podemos formalizar el **cálculo de los valores Q** por medio de la siguiente ecuación:

$$Q(st, at) = r(st, at) + \gamma \max_{at+1} Q(st + 1, at + 1)$$

- El **valor de Q óptimo** para un par (estado, acción) es la suma de la recompensa recibida cuando se aplica la acción junto al valor descontado del mejor valor Q que se puede conseguir desde el estado alcanzando al aplicar esa acción. Para

aproximar este cálculo, al principio del aprendizaje los valores Q se establecen a un valor fijo (puede ser aleatorio), a continuación el agente va tomando pares de (estado, acción) y anota cuánta recompensa recibe en ellos, entonces, actualiza el valor almacenado del valor Q de cada par considerando como ciertas las anotaciones tomadas de los otros pares.

- Uno de los problemas que presenta Q-Learning es que el agente requiere un gran número de episodios de entrenamiento para aprender una función de **valor aceptable**.
- Actualmente hay dos enfoques principales para la **aceleración del proceso de aprendizaje**:
 1. Permitir la incorporación de información provista por un observador externo.
 2. Integrar learning con planning
- El primero consiste en facilitar que un experto u observador externo pueda incorporar “consejos” que le sirvan al agente para aprender ciertos aspectos complejos del ambiente en forma más eficaz.
- En el segundo caso, el agente aprende un modelo del ambiente en forma simultánea al aprendizaje de la política, lo que favorece hacer un uso más intensivo de una cantidad limitada de experiencia.
- Existe un enfoque alternativo para acelerar el proceso de aprendizaje, basado en la implementación **paralela** del algoritmo de Q-Learning.
- Conviene realizar un repaso a los elementos que inciden directamente en el proceso de entrenamiento de los modelos basados en el aprendizaje profundo por refuerzo:
 - **Acciones:** pueden ser entendidas como las diferentes decisiones que acaba llevando a cabo el agente ante estados concretos del ambiente.
 - **Recompensas:** pueden ser tanto positivas (se suma puntuación) o negativas (se resta puntuación)
 - **Comportamiento:** se trata de un aspecto muy interesante, relacionado con el dilema de exploración y explotación. En ese caso, se busca evitar el denominado como comportamiento avaricioso (o greedy) por parte del agente: es decir, que no se rija por la consecución de recompensas a corto plazo y con comportamientos que considera seguros, atreviéndose a explorar y a interactuar con nuevas posibilidades que le ofrece el ambiente.
 - **Políticas:** aparecerán reflejadas en una tabla (puede llegar a tener una dimensión no definida) en la que el agente verá reflejado el modelo comportamental que tendrá que adoptar ante cada tipo de situación.
- Teniendo en cuenta todo ello, a través del empleo del algoritmo Q-Learning lo que se busca es conseguir que, a través del entrenamiento y de las diferentes simulaciones a las que sea sometido, el agente consiga almacenar la mayor cantidad de información en sus propias políticas.
- Todo ello, consiguiendo tanto que el agente obtenga la mayor cantidad de puntuaciones posibles en las recompensas como que se aventure y eluda el estancamiento.

LA ECUACIÓN DE BELLMAN

- Para conseguir que el agente vaya rellenando y cumplimentando de forma óptima e idónea las políticas, un aspecto de gran importancia que tendrá que ser considerado en todo momento será la conocida como **ecuación de Bellman**.
- Se trata de una ecuación (que también puede ser conocida como ecuación de programación dinámica) que debe su denominación a Richard Bellman, descubridor de la misma.
- Se trata de un aspecto básico y fundamental para conseguir que se cumpla la optimalidad que se encuentra asociada, de manera inherente, a los procesos de optimización matemática.
- En líneas generales, a través de la actuación de Bellman es posible describir el valor de un problema de decisión en determinados puntos en el tiempo, valiéndose de recompensas que son fruto tanto de las opciones iniciales como de la resolución final del problema en base a esas opciones inicialmente presentadas.
- Como resultado directo de todo ello, resulta posible establecer, de un problema de optimización general, partes más pequeñas de resolución simplificada, denominadas como subproblemas.
- Se trata de una ecuación que ha tenido importantes y notables aplicaciones. En un primer momento, fue utilizada en el ámbito de la ingeniería y la matemática aplicada. Sin embargo, con el paso del tiempo, la ecuación de Bellman ha pasado a erigirse en un interesantísimo mecanismo en las teorías económicas.
- En el caso concreto del aprendizaje profundo por refuerzo, la ecuación que suele emplearse viene definida de la siguiente forma:

$$Q^{s,a} = q(s,a) + \alpha[R + (\lambda \max Q(s'))] - Q(s,a)]$$

Donde:

Q(s-a): valor actual

a: ratio de aprendizaje

R: recompensa

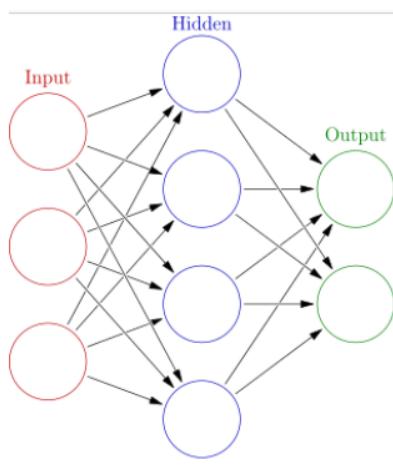
λ: tasa de descuento

maxQ(s'): valor óptimo esperado

Modelos profundos (Deep Learning)

COMPONENTES

- Las redes neuronales son un modelo computacional basado en un gran conjunto de neuronas simples que intentan simular el comportamiento de las neuronas de cerebros biológicos.



- Cada unidad neuronal se encuentra conectada con muchas otras y los enlaces entre ellas podrán ir incrementando o inhibiendo el estado de activación de neuronas adyacentes.
- Las redes neuronales son un **conjunto de algoritmos**, modelados libremente a partir del cerebro humano, que están diseñados para reconocer patrones. Interpretan los datos sensoriales a través de una especie de percepción de máquina, etiquetado o agrupación de datos sin procesar. Los patrones que reconocen son numéricos, contenidos en vectores, a los que se deben traducir todos los datos del mundo real, ya sean imágenes, sonido, texto o series de tiempo.
- Las redes neuronales nos ayudan a agrupar y clasificar. Podemos pensar en ellas como una capa de agrupamiento y clasificación sobre los datos que almacena y administra. Ayudan a agrupar los datos sin etiquetar de acuerdo con las similitudes entre las entradas de ejemplo, y clasifican los datos cuando tienen un conjunto de datos etiquetados para entrenar.
- Las redes neuronales también pueden extraer características que se alimentan a otros algoritmos para la agrupación y clasificación; por lo tanto, se puede pensar en redes neuronales profundas como componentes de aplicaciones de aprendizaje automático más grandes que involucran algoritmos para aprendizaje de refuerzo, clasificación y regresión.

Origen biológico

- Las redes neuronales surgen como un método de aprendizaje automático que simula el **comportamiento del cerebro**.
- Al igual que ocurre con nuestro cerebro, el aprendizaje de estas neuronas **no es supervisado** de forma que nosotros le daremos una entrada de datos, estas

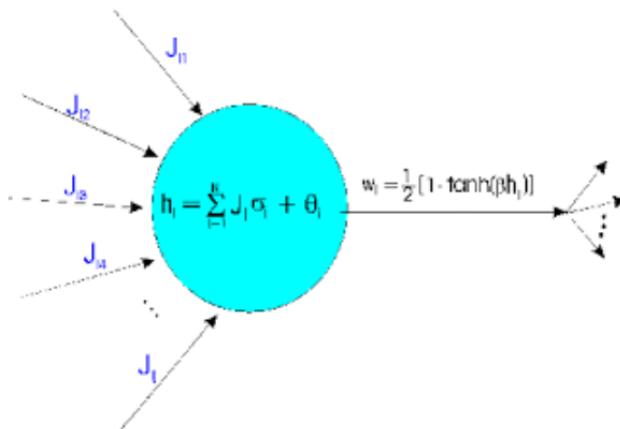
neuronas realizarán las operaciones y comunicaciones necesarias y nos darán una salida, pero nosotros no podemos ver qué pasos intermedios ha realizado para llegar a los resultados que nos ha dado como salida.

- El objetivo de una red neuronal es resolver los problemas de la misma forma que lo hace nuestro cerebro, aunque estas redes neuronales llegan a ser incluso más abstractas.
- Los proyectos actuales de redes neuronales suelen tener varias capas o bien un diseño de cubo, y la señal se va propagando de adelante hacia atrás.

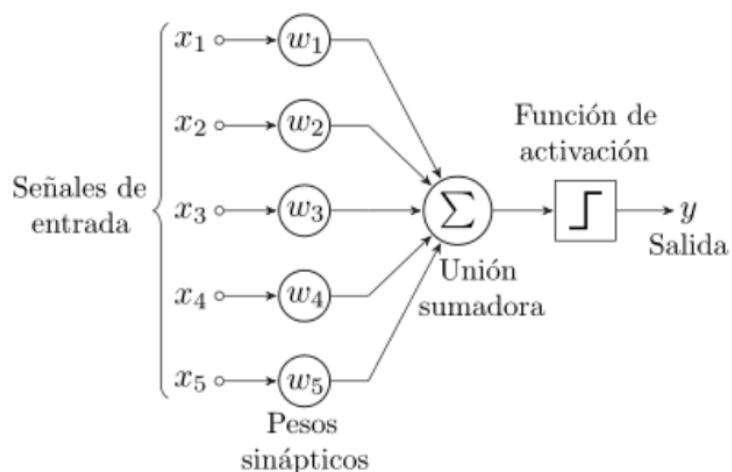
La neurona formal

- Una neurona formal es un modelo matemático que se utiliza dentro de la inteligencia artificial y se asocia con otras neuronas formales para construir lo que hemos llamado red neuronal.
- Una neurona formal contendrá varias fórmulas en su interior que intentarán reproducir lo más fielmente posible el funcionamiento de una neurona con las diferentes entradas, las importaciones y las salidas.

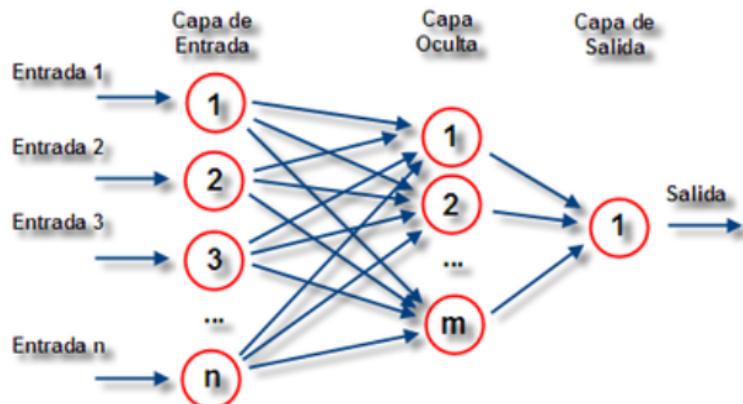
NEURONA FORMAL



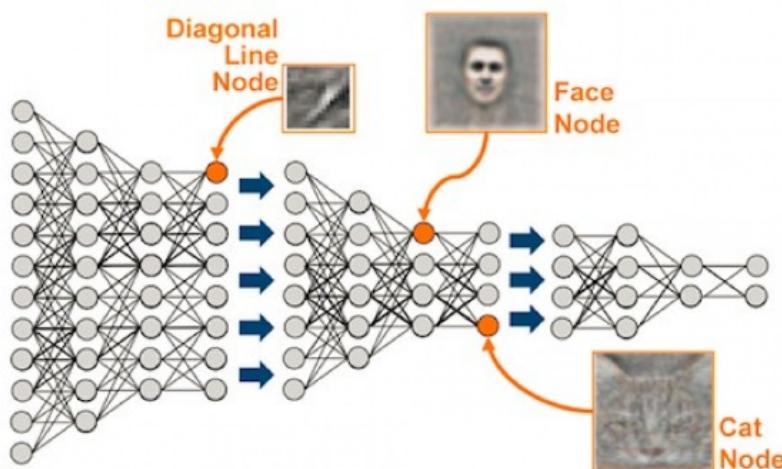
- El perceptrón dentro del campo de las redes neuronales puede tener dos conceptualizaciones.
- El perceptrón se puede referir a un tipo de red neuronal artificial que desarrolló Fran Rosenblatt, o bien, se puede entender como la neurona artificial o la unidad básica de inferencia en forma de discriminador lineal, a partir del que se desarrollará un algoritmo capaz de generar un criterio de selección.



- El modelo más simple de perceptrón es el de una única neurona, que será una célula especializada y caracterizada por tener una cantidad de canales de entrada que llamaremos **dendritas** y un canal de salida llamado **axón**.
- Una neurona sola no tiene razón de ser, y su labor especializada se irá haciendo más valiosa conforme se vaya uniendo con otras neuronas, y formando una red.
- El término **Feed-forward** se utiliza para describir un tipo de sistema que reacciona con los cambios en su entorno para mantener algún estado concreto del sistema.
- Para implementar un sistema de este tipo se necesitan muchos prerequisitos, las alteraciones que ocurran en el entorno deben poder medirse, y el tiempo que las alteraciones afectan al sistema debe ser mayor que el del sistema en sí.
- Este tipo de sistemas, hacen referencia a las redes del tipo perceptrón multicapa, en las que las salidas de las neuronas se conectan en las siguientes capas de forma que no existen bucles de retroalimentación.



- Las redes se forman a partir de **tres tipos de capas** según su localización:
 1. **Capa de entrada:** recibe los datos, tan solo es una capa y tendrá tantos nodos como datos de entrada se introduzcan.
 2. **Capa oculta:** capas que se localizan entre la capa de entrada y la de salida que se encargan de procesar los datos para alcanzar los resultados.
 3. **Capa de salida:** su función es sacar los datos que forman parte de la respuesta y también está formada solamente por una capa como la capa de entrada.
- Las redes neuronales profundas, las redes de creencias profundas y las redes neuronales recurrentes se han aplicado a campos como la visión por computadora, el reconocimiento de voz, el procesamiento del lenguaje natural, el reconocimiento de audio, el filtrado de redes sociales, la traducción automática y la bioinformática donde produjeron resultados comparables a los obtenidos por expertos humanos, y, en algunos casos, incluso mejores.



- Los algoritmos y redes de aprendizaje profundo pueden basarse en dos técnicas:
 - El aprendizaje no supervisado de múltiples niveles de características o representaciones de los datos. Las entidades de nivel superior se derivan de entidades de nivel inferior para formar una representación jerárquica.
 - Alguna forma de descenso en gradiente para el entrenamiento.

DNQ (DEEP Q NETWORK)

- Este concepto es una combinación de **redes neuronales profundas** con el algoritmo de aprendizaje por refuerzo **Q-learning**. Este algoritmo funciona bien cuando el entorno que se va a tratar es simple y la función $Q(s,a)$ queda representada como una matriz de valores.
- El problema aparece cuando se establecen una gran cantidad de estados y acciones diferentes, ya que la matriz del algoritmo se expande hasta lograr un tamaño muy grande y pasa a ser no viable la utilización de esta.
- Para la resolución de este problema, se creó el **algoritmo DQN**, del que han surgido otros algoritmos que son una evolución de este.
- En el caso de DQN, se utilizan dos redes neuronales con el objetivo de estabilizar el proceso de aprendizaje. Por un lado, la primera **red neuronal principal** está representada por los **parámetros 0**. Esta red se utiliza para estimar los valores Q del estado s y la acción a del presente: $Q(s,a;0)$. La segunda, la **red neuronal objetivo**, parametrizada por $0'$, tendrá la misma arquitectura que la red principal, pero se utilizará para aproximar los valores-Q del siguiente estado s' y la siguiente acción a' .
- El **aprendizaje** se produce en la red principal y no en la red objetivo. La **red objetivo** se bloquea, es decir, su parámetros no se cambian durante varias iteraciones y después los parámetros de la red principal se copian a la red objetivo, transmitiendo así el aprendizaje de una a otra, haciendo que las estimaciones calculadas por la red objetivo sean más precisas.

$$Q(s, a; \theta) = r + \gamma \max_{a'} Q(s', a'; \theta')$$

Ecuación de Bellman en DQN

¿QUÉ TIPO DE REDES NEURONALES HAY EN DEEP LEARNING?

CONVOLUTIONAL NEURAL NETWORKS (CNN, REDES NEURONALES CONVOLUCIONALES)

- Son redes neuronales artificiales que han sido creadas para procesar **matrices estructuradas**, como imágenes. Es decir, se encargan de clasificar imágenes basándose en los patrones y objetos que aparecen en ellas.
- Las CNN suelen usarse en visión artificial, ya que pueden trabajar con imágenes brutas (raw images) y no necesitan hacer un procesamiento previo.
- Esta cualidad hace que sean muy útiles para aplicaciones visuales de clasificación de imágenes, pero también procesamiento del lenguaje natural (natural language processing, nlp), ayudando con la clasificación de textos.

RECURRENT NEURAL NETWORKS (RNN)

- Las redes neuronales recurrentes son redes neuronales que usan **datos secuenciales o datos de series de tiempo**. Este tipo de redes solucionan problemas ordinales o temporales, como la traducción de idiomas, reconocimiento de voz, procesamiento de lenguaje natural y captura de imágenes.
- Las redes neuronales recurrentes se diferencian de otras redes neuronales artificiales en que tienen “**memoria**”. Es decir, las RNN recogen información de inputs anteriores para influenciar los inputs y outputs actuales.

Al escribir con el móvil, el teclado te muestra una serie de palabras como sugerencias a partir de lo que está escrito.

GENERATIVE ADVERSARIAL NETWORKS (GAN)

- Las redes generativas antológicas utilizan dos redes neuronales artificiales y las opone la una a la otra (por eso se les conoce como antológicas) para generar **nuevo contenido o datos sintéticos** que pueden hacerse pasar por reales. Una de las redes genera y la otra funciona como “discriminadora”.
- La red discriminatoria (también conocida como red antagonista) ha sido entrenada para **reconocer contenido real** y hace de censor para que la red que genera contenido haga contenido que parezca real. Por eso, este tipo de redes son muy usadas para generar imágenes, videos y voces.
- Como ya hemos visto, la IA es cualquier código, algoritmo o técnica que permite que una computadora imite el comportamiento cognitivo humano a la inteligencia.
- **Machine learning** es un subconjunto de IA que utiliza métodos estadísticos para permitir que las máquinas aprendan y mejoren con la experiencia.
- **Deep Learning** es un subconjunto de Machine Learning, que hace factible el cálculo de redes neuronales multicapas.
- **El aprendizaje automático se ve como un aprendizaje superficial, mientras que el aprendizaje profundo se ve como un aprendizaje jerárquico con abstracción.**

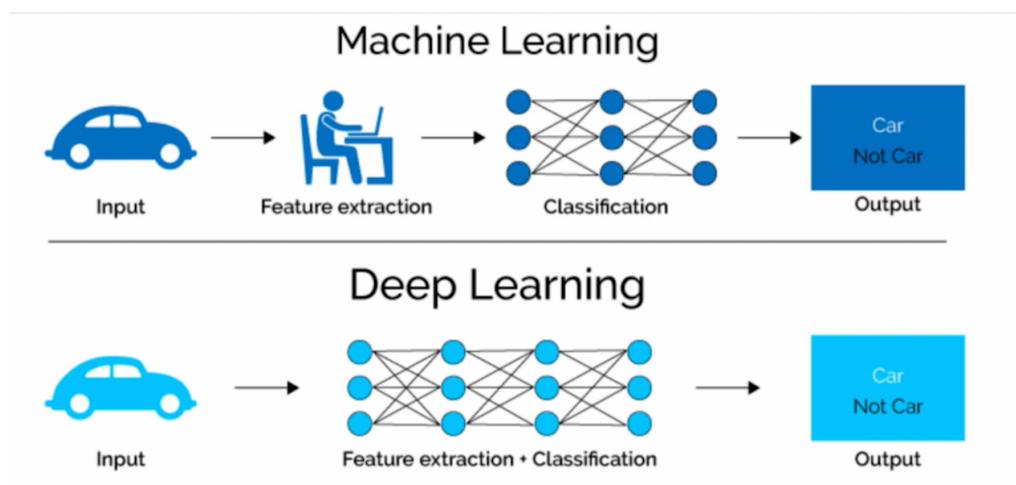
- El aprendizaje automático trata con una amplia gama de conceptos:
 - Supervisado
 - No supervisado
 - Aprendizaje reforzado
 - Regresión lineal
 - Funciones de coste
 - Sobreajuste (overfitting)
 - Infra ajuste (under-fitting)
 - Hiperparámetro
 - Etc.
- En el aprendizaje **supervisado**, aprendemos a predecir valores a partir de datos etiquetados. Una técnica de ML que ayuda aquí es la **clasificación** donde los valores objetivo son valores discretos; por ejemplo, gatos y perros.
- Otra técnica en el aprendizaje automático que podría ser de ayuda es la **regresión**. La regresión funciona en los valores objetivos. Los valores objetivo son valores continuos; por ejemplo, los datos del mercado de valores se pueden analizar mediante regresión.
- En el aprendizaje **no supervisado**, hacemos inferencias a partir de los datos de entrada que no están etiquetados o estructurados. Si tenemos un millón de registros médicos y tenemos que darle sentido, encontrar la estructura subyacente, valores atípicos o detectar anomalías, utilizamos la técnica de agrupamiento para dividir los datos en grupos amplios.
- Los conjuntos de datos se dividen en conjuntos de entrenamiento, conjuntos de prueba, conjuntos de validación, etc.
- En 2012, un algoritmo clasificó 1 millón de imágenes en 1000 categorías con éxito utilizando 2 GPU (Graphics Processing Unit) y las últimas tecnologías como Big Data. Esto puso de moda el concepto de Deep Learning.

Relación entre el aprendizaje profundo y el aprendizaje automático tradicional

- Uno de los principales desafíos encontrados en los modelos tradicionales de aprendizaje automático es un proceso llamado **extracción de características**. El desarrollador debe ser específico y decirle a la computadora las características a tener en cuenta. Estas características ayudarán a tomar decisiones.
- La introducción de datos sin procesar en el algoritmo rara vez funciona, por lo que la extracción de características es una parte crítica del flujo de trabajo tradicional de aprendizaje automático.
- Es una gran responsabilidad para el programador, ya que la eficiencia del algoritmo depende en gran medida de lo ingenioso que sea el programador. Pensemos en ejemplos de problemas complejos como el reconocimiento de objetos o el reconocimiento de escritura a mano.
- El aprendizaje profundo, con la capacidad de aprender múltiples capas de presentación, es uno de los pocos métodos que nos ha ayudado con la extracción automática de características. Se puede suponer que las capas inferiores realizan una extracción automática de características, lo que requiere poca o ninguna guía del desarrollador.

Diferencias entre aprendizaje automático y profundo

- El aprendizaje automático es el arte de la ciencia de hacer que las computadoras actúen según los algoritmos diseñados y programados.
- Muchos investigadores piensan que el aprendizaje automático es la mejor manera de avanzar hacia la IA a nivel humano.
- El aprendizaje automático incluye los siguientes tipos de patrones:
 - Patrón de aprendizaje supervisado
 - Patrón de aprendizaje no supervisado
- El aprendizaje profundo, en cambio, es un subcampo del aprendizaje automático donde los algoritmos en cuestión se inspiran en la estructura y función del cerebro llamadas redes neuronales artificiales.
- Todo el valor actual de aprendizaje profundo es a través del aprendizaje supervisado o el aprendizaje a partir de datos y algoritmos etiquetados.
- Cada algoritmo en aprendizaje profundo pasa por el mismo proceso. Incluye una jerarquía de transformación no lineal de entrada que se puede utilizar para generar un modelo estadístico como salida.



- Los siguientes pasos definen el proceso de **Machine Learning**:
 - Identifica conjuntos de datos relevantes y los prepara para su análisis
 - Elige el tipo de algoritmos a usar
 - Crea un modelo analítico basado en el algoritmo utilizado
 - Entrena el modelo en conjuntos de datos de prueba y lo revisa según sea necesario
 - Ejecuta el modelo para generar puntajes de prueba
- Las diferencias entre ambos aprendizajes son las siguientes:

La cantidad de datos

El aprendizaje automático funciona con grandes cantidades de datos. También es útil para pequeñas cantidades de datos. El aprendizaje profundo, por otro lado, funciona de manera eficiente si la cantidad de datos aumenta rápidamente.

Proceso de Machine Learning:

- Identifica conjuntos de datos relevantes y los prepara para su análisis
- Elige el tipo de algoritmo a usar
- Crea un modelo analítico basado en el algoritmo utilizado
- Entrena el modelo en conjuntos de datos de prueba y lo revisa según sea necesario
- Ejecuta el modelo para generar puntajes de prueba

DEPENDENCIAS DE HARDWARE

- Los algoritmos de aprendizaje profundo están diseñados para depender en gran medida de máquinas de alta gama a diferencia de los algoritmos tradicionales de aprendizaje automático. Los algoritmos de aprendizaje profundo realizan varias operaciones de multiplicación de matrices, que requieren una gran cantidad de soporte de hardware.

INGENIERÍA DE CARACTERÍSTICAS

- La ingeniería de características es el proceso de poner el conocimiento del dominio en características específicas para reducir la complejidad de los datos y crear patrones que sean visibles para los algoritmos de aprendizaje que funcionan.
- Ejemplo: los patrones tradicionales de aprendizaje automático se centran en píxeles y otros atributos necesarios para el proceso de ingeniería de características. Los algoritmos de aprendizaje profundo se centran en características de alto nivel a partir de datos. Reduce la tarea de desarrollar un nuevo extractor de características de cada nuevo problema.

ENFOQUE DE RESOLUCIÓN DE PROBLEMAS

- Los algoritmos tradicionales de aprendizaje automático siguen un procedimiento estándar para resolver el problema. Rompe el problema en partes, resuelve cada una de ellas y las combina para obtener el resultado requerido. El aprendizaje profundo se enfoca en resolver el problema de punta a punta en lugar de dividirlos en divisiones.

TIEMPO DE EJECUCIÓN

- El tiempo de ejecución es la cantidad de tiempo requerida para entrenar un algoritmo. El aprendizaje profundo requiere mucho tiempo para entrenar, ya que incluye muchos parámetros, lo que lleva más tiempo de lo habitual. El algoritmo de aprendizaje automático en comparación requiere menos tiempo de ejecución.

INTERPRETABILIDAD

- La interpretabilidad es el factor principal para la comparación del aprendizaje automático y los algoritmos de aprendizaje profundo. La razón principal es que el aprendizaje profundo todavía se da un segundo pensamiento antes de su uso en la industria.

Ejemplos con Weka / Orange

MINERÍA DE DATOS O DATA MINING

- Es un conjunto de técnicas y tecnologías que permiten explorar **grandes bases de datos**, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos que expliquen el comportamiento de estos datos.
- Este concepto surgió con el objetivo de ayudar a comprender una enorme cantidad de datos y que estos pudieran ser utilizados para extraer conclusiones para contribuir en la mejora y el crecimiento de las empresas.
- Su principal **finalidad** es explorar, mediante la utilización de distintas técnicas y tecnologías, bases de datos enormes de manera automática.
- El objetivo es encontrar patrones repetitivos, tendencias o reglas que expliquen el comportamiento de los datos que se han ido recopilando con el tiempo.

HERRAMIENTAS PARA MINAR DATOS

RAPID MINER

- Es uno de los sistemas de análisis predictivo más populares creado por la empresa con el mismo nombre. Está escrito en lenguaje de programación JAVA. Ofrece un entorno integrado para minería de texto, aprendizaje profundo, aprendizaje automático y análisis predictivo.
- El instrumento se puede utilizar para una amplia gama de aplicaciones, incluidas aplicaciones empresariales, aplicaciones comerciales, investigación, educación, formación, desarrollo de aplicaciones y aprendizaje automático.

ORANGE

- Orange es un paquete de software de minería de datos y aprendizaje automático perfecto. Es compatible con la visualización y es un software basado en componentes escritos en lenguaje informático Python y desarrollado en el laboratorio de bioinformática de la facultad de informática y ciencias de la información de la Universidad de Ljubljana, Eslovenia.
- Como es un software basado en componentes, los componentes de Orange se denominan “**widgets**”. Estos widgets van desde el procesamiento y la visualización de datos hasta la evaluación de algoritmos y el modelado predictivo.
- Los widgets ofrecen funcionalidades importantes como:
 - Mostrar tabla de datos y permitir seleccionar características
 - Lectura de datos Entrenamiento de predictores y comparación de algoritmos de aprendizaje
 - Visualización de elementos de datos, etc.

KNIME

- KNIME es la mejor plataforma de integración para análisis de datos e informes desarrollada por KNIME.com AG. Opera sobre el concepto de **canalización de datos modular** y consta de varios componentes de aprendizaje automático y minería de datos integrados juntos
- KNIME se ha utilizado ampliamente para la investigación farmacéutica. Además, funciona de manera excelente para el análisis de datos de clientes, el análisis de datos financieros y la inteligencia empresarial.
- Tiene algunas características brillantes como implementación rápida y eficiencia de escala. Los usuarios se familiarizan con KNIME en bastante menos tiempo y ha hecho que el análisis predictivo sea accesible incluso para usuarios ingenuos. Utiliza el ensamblaje de nodos para preprocesar los datos para análisis y visualización.

Weka, Waikato Environments for Knowledge analysis es un conjunto de librerías JAVA para extraer conocimientos desde bases de datos. Es un software que ha sido desarrollado en la universidad de Waikato (Nueva Zelanda) bajo licencia GPL, es decir es software libre lo cual ha impulsado que sea una de los frameworks más utilizados en esta área. Weka contiene las herramientas necesarias para realizar tareas de minería de datos especialmente preparación de datos, clasificación, regresión, clustering, características de selección y visualización además de transformaciones sobre los datos. Una de sus hipótesis se basa en que los datos se encuentran todos en un único fichero plano tipo arff, que veremos más adelante. Una de las propiedades más interesantes de este software es su facilidad para modificar métodos, añadir extensiones, etc. Weka funciona en cualquier plataforma en la que esté instalada una máquina virtual Java ya que está implementado en Java.

Existen otras herramientas similares como Oracle Data Miner, Clementine, otra desarrollada por IBM y otra por SAS pero el hecho de que Weka sea desarrollado bajo GPL es decir software libre lo ha hecho una alternativa muy interesante y accesible.

- En cuanto a las **ventajas** que presenta Weka frente a otras herramientas podemos destacar:
 - Es software libre
 - Es muy portable ya que al estar desarrollado en Java se ejecuta en cualquier plataforma que ejecute la máquina virtual de Java
 - Su interfaz de usuario es muy intuitiva y no es necesario tener conocimientos de programación
 - Contiene una extensa colección de técnicas para el procesamiento de datos y el modelado
 - Weka soporta las tareas propias de la Minería de Datos, especialmente el procesamiento de datos, clustering, clasificación, regresión, visualización y selección
 - Weka carga datos desde ficheros planos tipo arff que tienen una estructura interna determinada.
 - Cabecera: @RELATIONS iris donde **iris** es el nombre de la relación. Es muy útil para grandes volúmenes de información ya que permite por ejemplo si contiene ventas por día, @RELATION dia_1, @RELATION dia_2, @RELATION dia_3 etc. y así podríamos hacer minería de datos solo de días concretos o de todos los días.

- Atributos: @ATTRIBUTE son los atributos del fichero y el tipo de datos y/o valores que pueda tomar. En el caso que estamos tratando:
 - @ATTRIBUTE sepalength REAL
 - @ATTRIBUTE sepalwidth REAL
 - @ATTRIBUTE petallength REAL
 - @ATTRIBUTE petalwidth REAL
 - @ATTRIBUTE class {iris-setosa,Iris-versicolor,Iris-virginica}
- El último declarado es la variable muestra con las etiquetas, en este caso class con lo que tenemos 3 posibles clasificaciones.

- **Datos:** @DATA son propiamente los datos de la relación declarada al inicio del fichero

@DATA

5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa

En nuestro caso para la primera fila teniendo en cuenta los atributos

Sepallength = 5.1

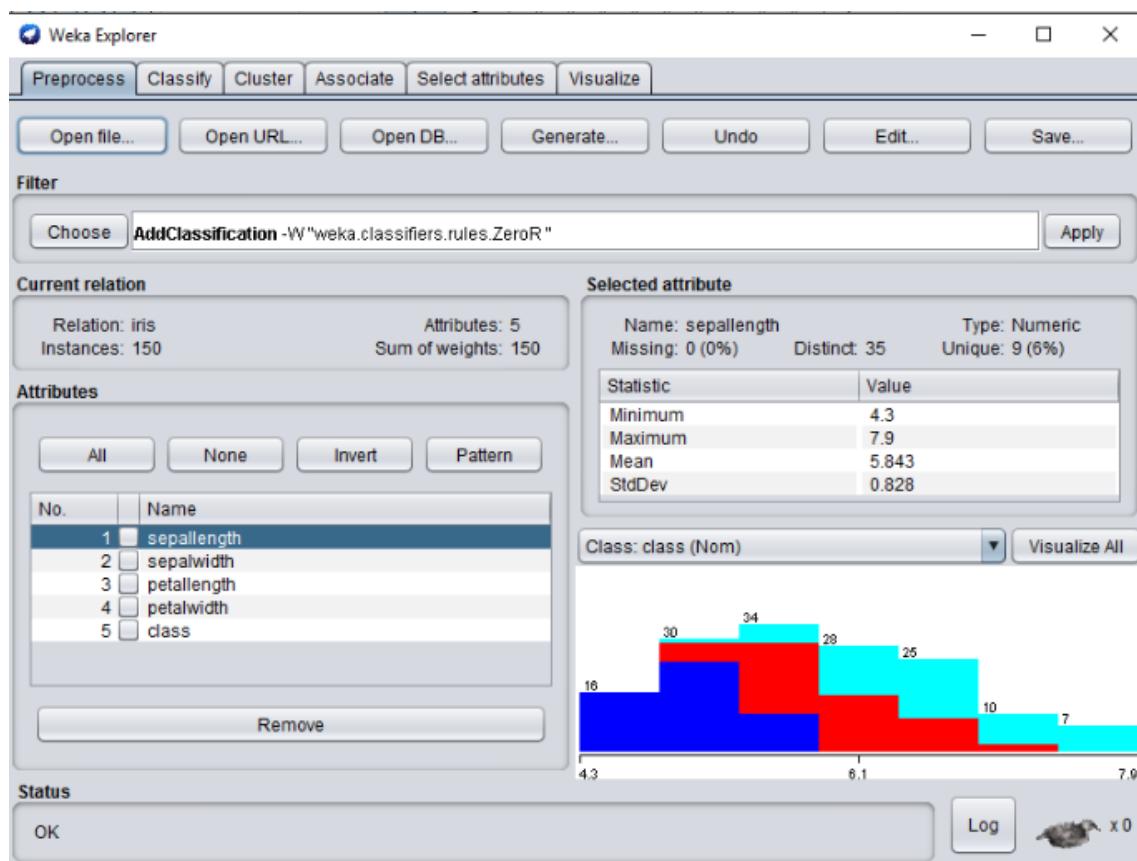
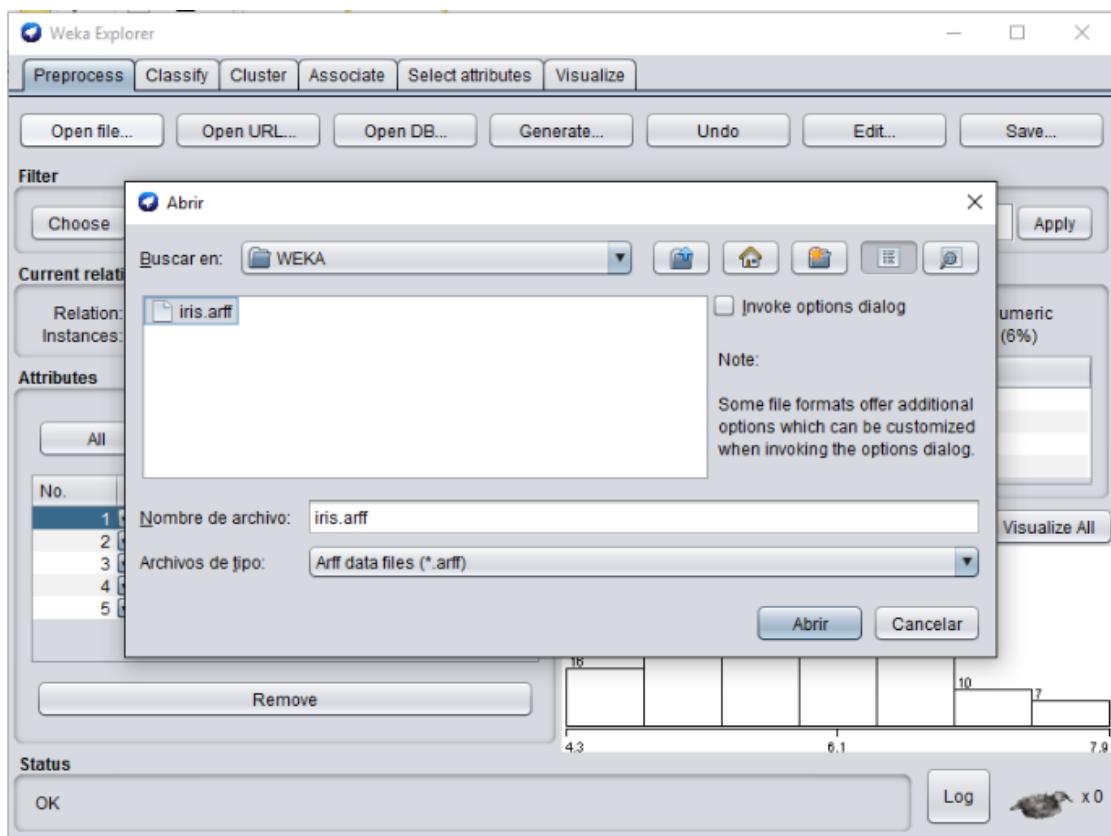
Sepalwidth = 3.5

Petallength = 1.4

Petalwidth = 0.2

Class = Iris-setosa

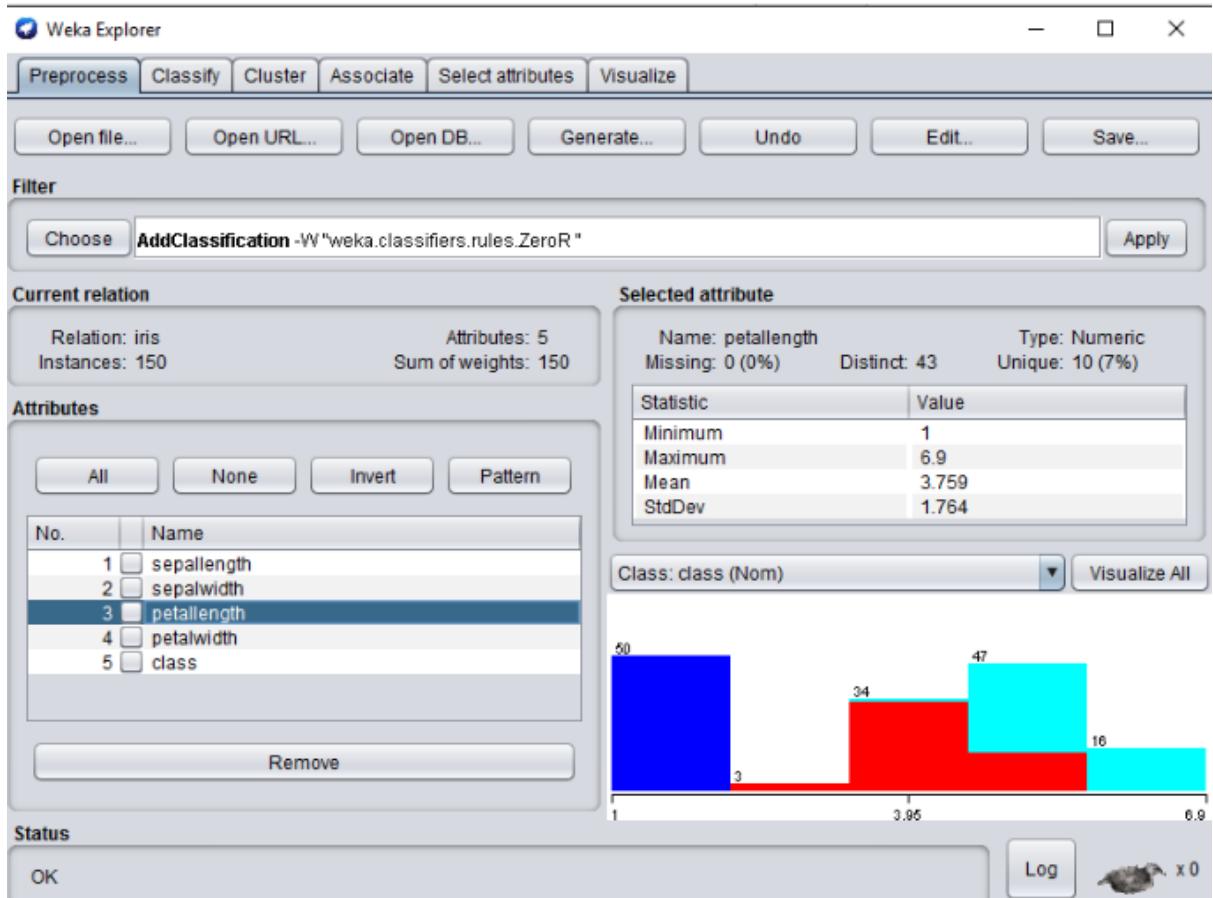
- Weka también proporciona acceso a bases de datos vía SQL debido a la conexión JDBC (Java Database Connectivity) y puede procesar los datos devueltos por una consulta hecha a la base de datos.
- Ofrece también la opción de poder usar Weka por terceros a través de su API.
- Weka es capaz de aplicar los siguientes algoritmos de minería de datos:
 - De clasificación: dada una serie de elementos etiquetados es capaz de predecir la etiqueta de nuevos elementos
 - De regresión: predicen variables continuas basándose en otras características de conjunto de datos
 - De selección de atributos: encuentra atributos nuevos
 - De clusterización o agrupamiento: identifica grupos con características similares
- Para ver estas técnicas de Data Mining con Weka lo mejor es hacerlo con un ejemplo. Tomaremos el fichero **iris.arff** que tenemos y lo cargaremos en el explorer de Weka



- Si por alguna razón no encontramos el archivo iris.arff en nuestro sistema, podemos descargarlo en el siguiente proyecto de github:

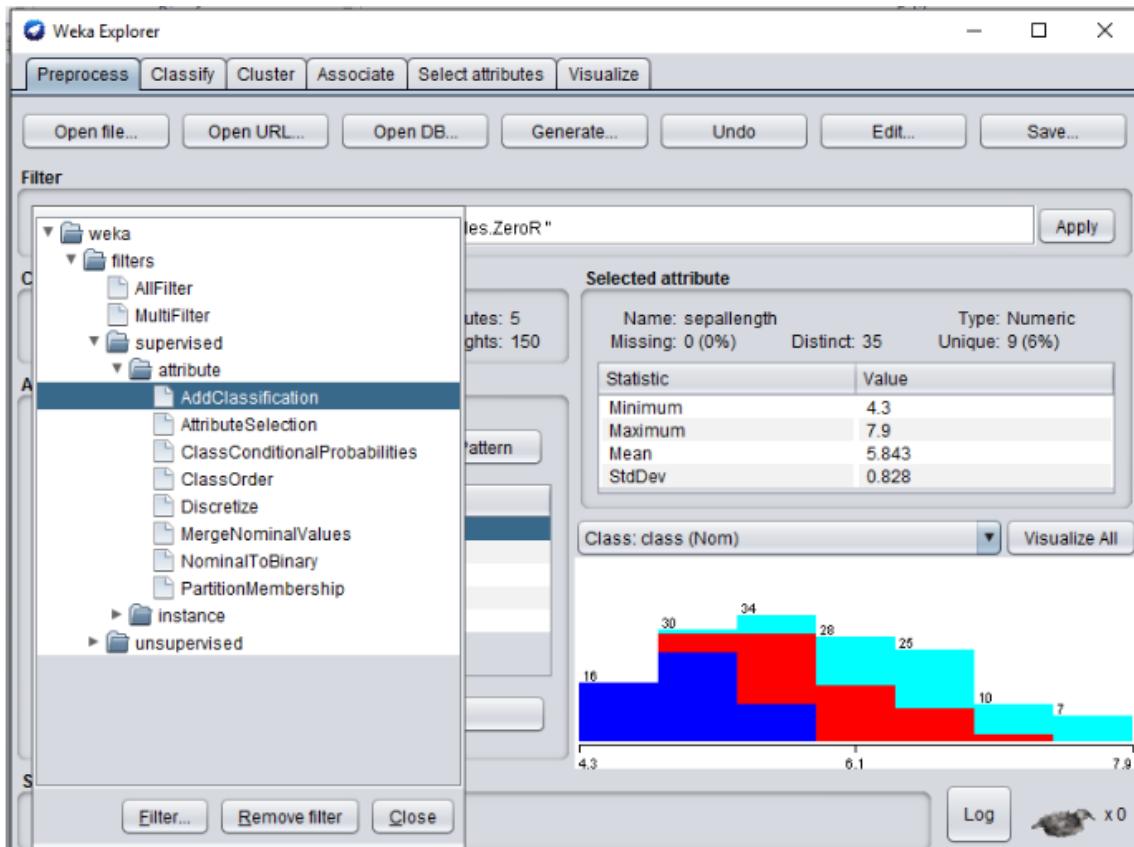
<https://gist.github.com/myui/143fa9d05bd6e7db0114>

- Al cargar este dataset, vemos que se muestran los datos para el atributo que está seleccionado, “sepallength”, si seleccionamos por ejemplo el 3 “petallength” se muestra



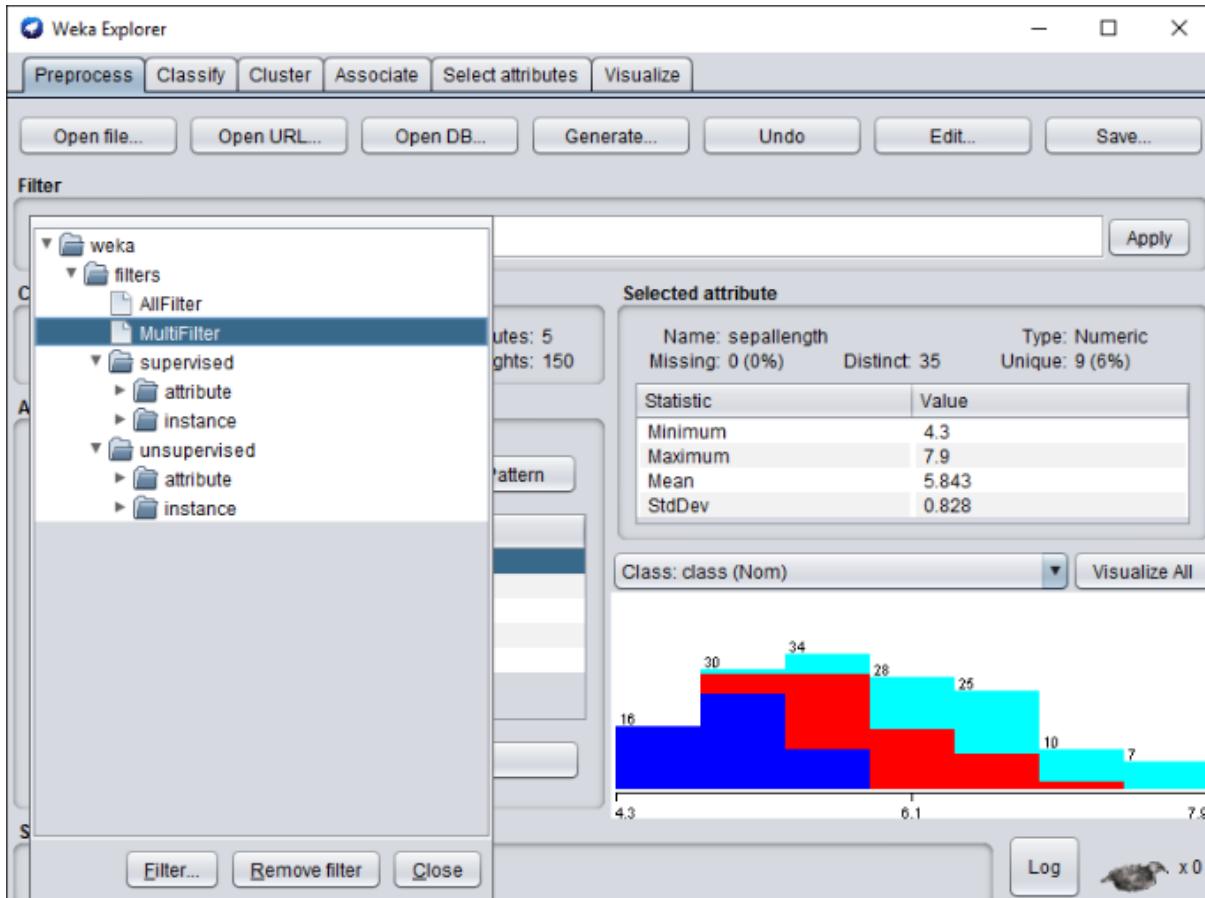
- En la lista de la parte izquierda se muestran los diferentes atributos y en la lista de la parte de la derecha se muestran las características de los registros para el atributo seleccionado, en este caso al ser numéricos se muestra el máximo, mínimo, valor medio y desviación estándar. Permite también eliminar algunos de los atributos seleccionándolo y marcándolos y haciendo clic en el botón inferior **Remove**

- Si hacemos clic en el botón “**Filter Choose**” podemos aplicar filtros

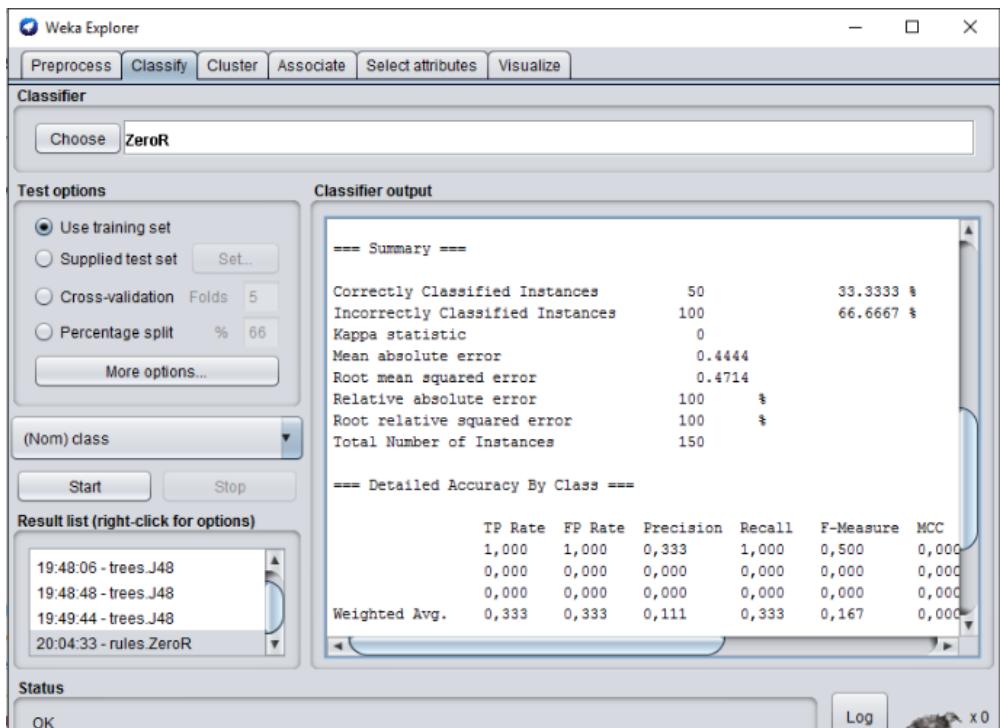


- Weka tiene integrados filtros que permiten realizar manipulaciones sobre los datos. Las operaciones de filtrado pueden aplicarse “**en cascada**”, de manera que cada filtro toma como entrada el conjunto de datos resultante de haber aplicado un filtro anterior. Existe siempre la opción de deshacer la última operación de filtrado aplicada con el botón **Undo**. Los resultados de aplicar filtros se pueden guardar en nuevos ficheros, que también serán de tipo arff, para manipulaciones posteriores. Resumiendo, en esta pestaña de Preprocess preparamos los datos para la posterior minería de datos pudiendo generar varios ficheros a partir de un mismo fichero de datos con distintos filtros.

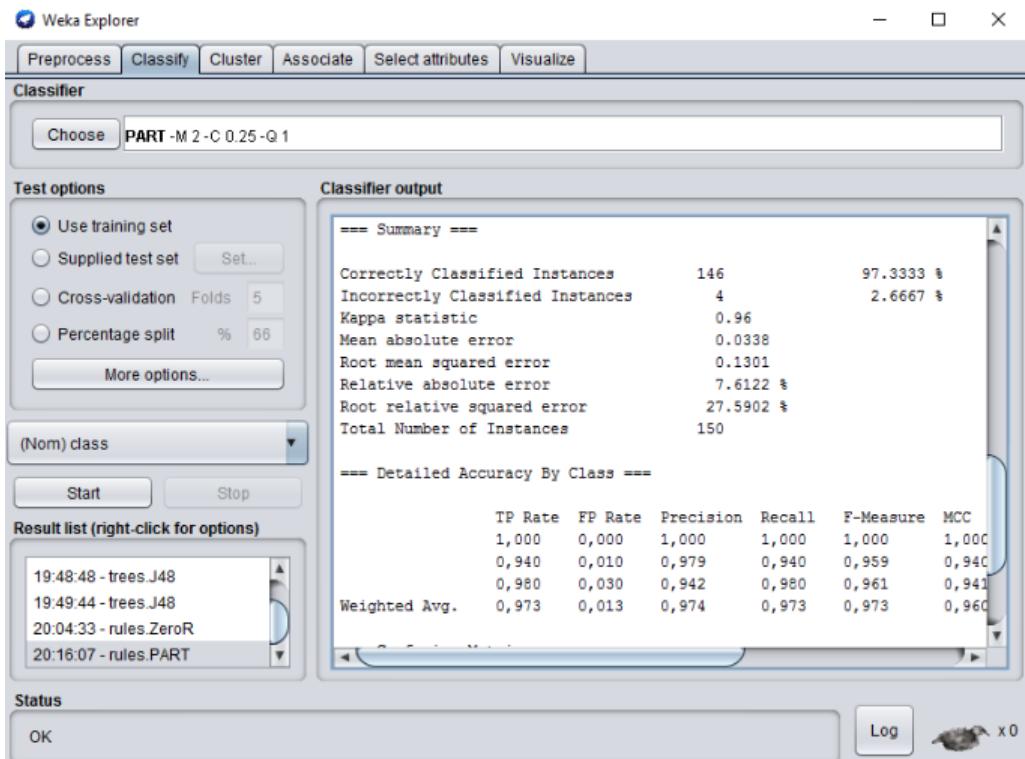
- Los filtros se pueden aplicar sobre instancias o sobre atributos



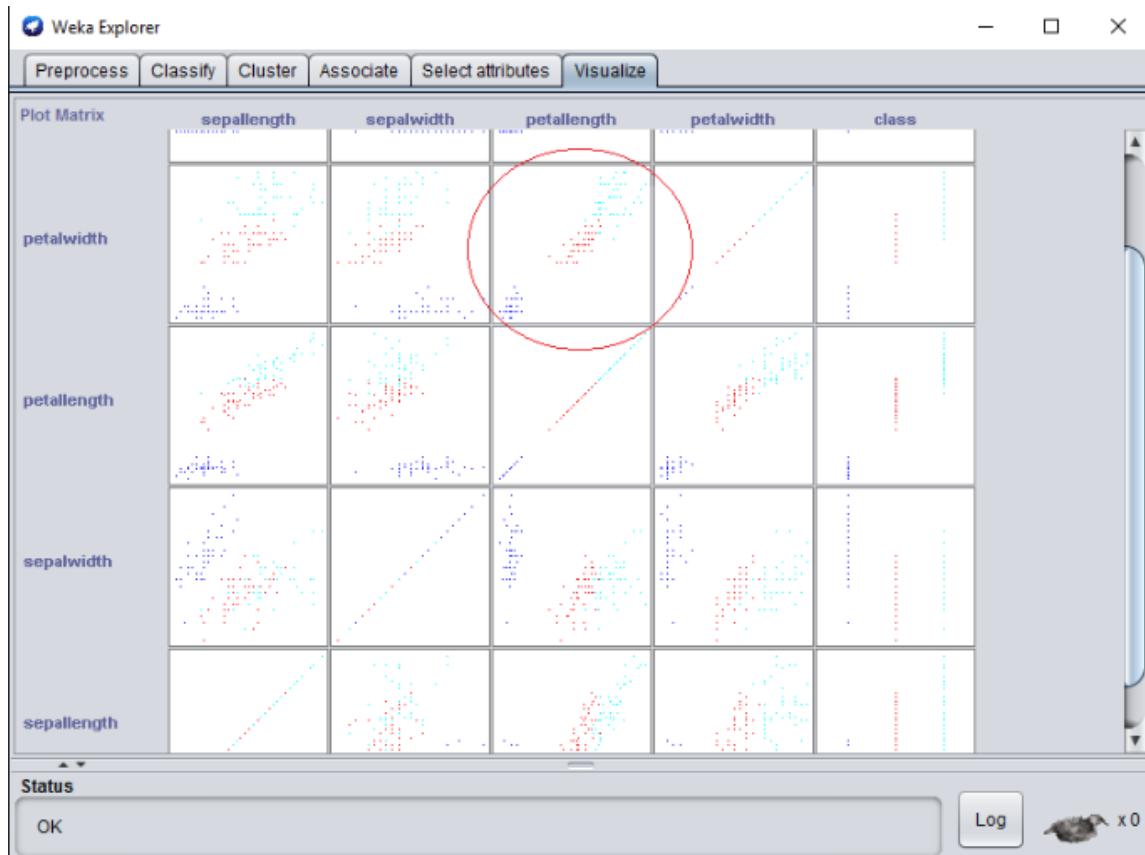
- **Sobre atributos:** es como si tomáramos columnas sobre la tabla de datos, como un “**filtro vertical**” y **sobre instancias:** es como si tomáramos un subconjunto de datos, como un “**filtro horizontal**” sobre la tabla de datos.
- Los filtros no supervisados son no supervisados porque son independientes del algoritmo posterior. Si vemos el contenido de “**unsupervised/instance**” hay filtros para la limpieza de datos como eliminar duplicados, eliminar un porcentaje de datos, etc.
- Aplicamos un algoritmo, en el caso de este fichero de datos, de clasificación u el que muestra por defecto que es **ZeroR**. Este clasificador clasifica a todos los datos con la clase de la clase mayoritaria. Es decir, si el 90% de los datos son A y el 10% son B, clasificará a todos los datos como A.
- Es conveniente utilizar primero este clasificador, porque el porcentaje de aciertos que obtengamos con él, es el que habrá que superar con el resto de clasificadores. Ejecutamos haciendo clic en “**Start**” y verificamos que efectivamente clasifica correctamente solo un 33% que corresponde a la clase setosa con la que clasificado dando error en el resto de datos 66%.



- Probemos ahora otro clasificador con estos mismos datos, ahora el clasificador **PART** que funciona construyendo reglas. Veamos qué resultados proporciona:

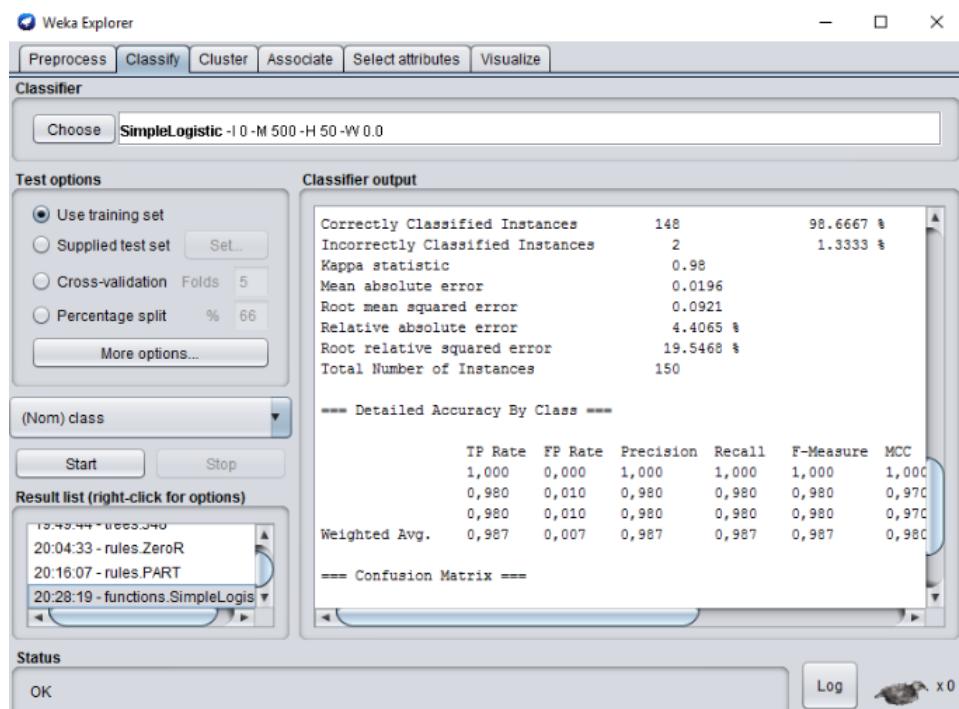


- Para este algoritmo hay una tasa de aciertos del 97,33%, bastante bueno comparado con el anterior. Podemos ver la matriz de confusión donde los aciertos caen en la diagonal y los errores fuera de ella. Vemos que los aciertos están bastante equilibrados, 100%, 94% y 98% lo que tiene sentido ya que los datos iniciales están equilibrados también como vimos en el ArffViewer. Weka ofrece también la información de las reglas que el clasificador ha creado, subiendo el scroll hacia arriba y son estas tres reglas:
 - Si petalwidth <= 0.6: Iris-setosa (50.0)
 - Si petalwidth <= 1.7 AND petallength <= 4.9: Iris-versicolor (48.0/1.0): Iris-virginica (52.0/3.0)
 - Si no se cumplen alguna de las dos primeras reglas lo clasifica como Iris-virginica
- Observemos que este algoritmo ha generado las reglas solo con 2 atributos **petalwidth** y **petallength**, por qué? Veamos que nos muestra la pestaña **Visualize**



- Verificamos en las gráficas que nos muestra que los atributos petalwidth y petallength usándolos juntos son lo que mejor clasifican, obtenemos los grupos más separados y además también vemos porque las iris-setosa era la clase que mejor se clasificaba, 100% ya que en el gráfico se corresponde con el color azul y es el conjunto mejor agrupado.

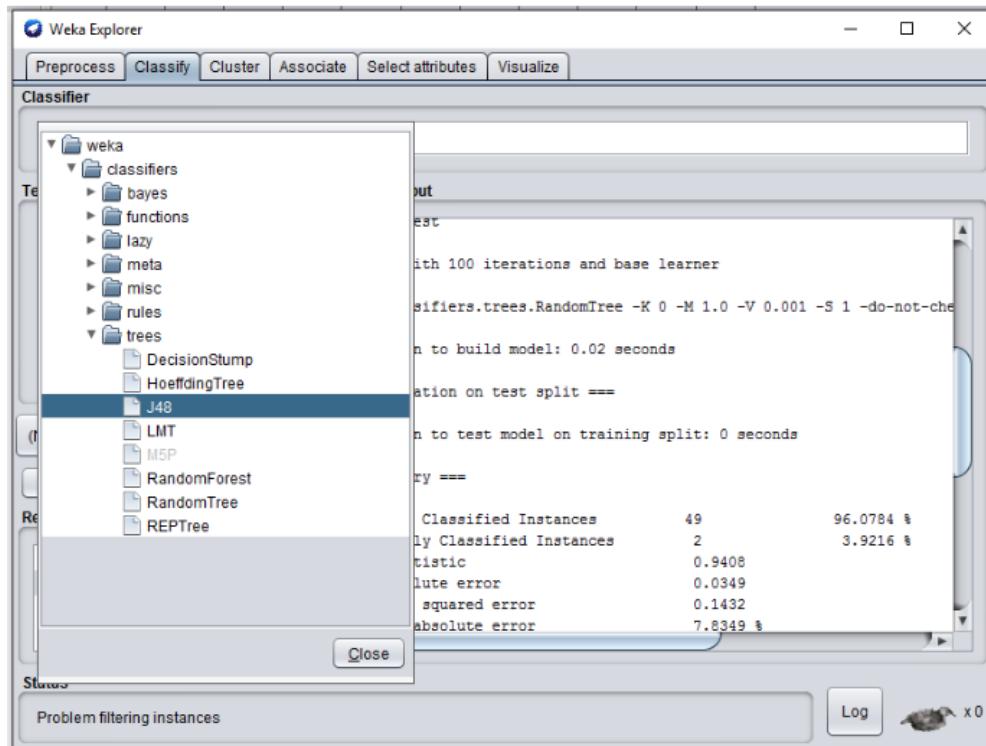
- Podríamos probar también con una regresión logística, que se encuentra en "functions" y obtendremos mejores resultados un 98.6% de aciertos y una matriz de confusión aún mejor.



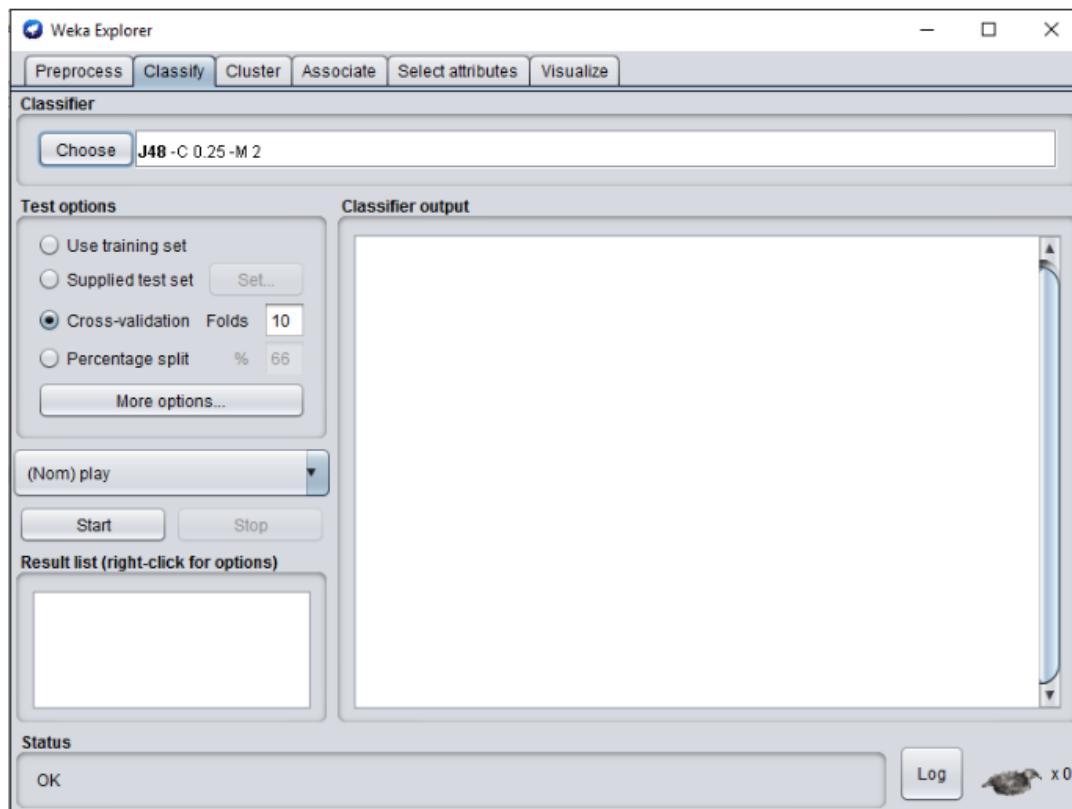
- Vamos a ver otro ejemplo aplicando clasificación concretamente un árbol de decisión a un fichero de datos proporcionado por la instalación de Weka **weather.nominal.arff**

Relation: weather.symbolic					
No.	1: outlook	2: temperature	3: humidity	4: windy	5: play
	Nominal	Nominal	Nominal	Nominal	Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

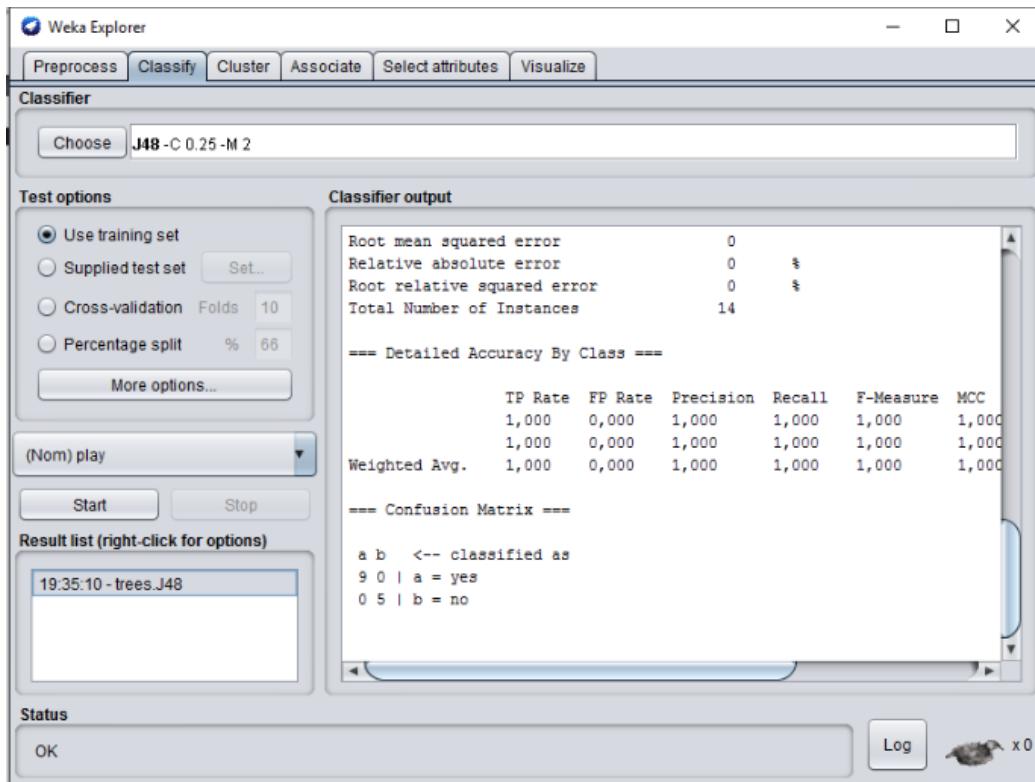
- Aplicaremos un árbol de decisión, que en Weka está implementado como J48



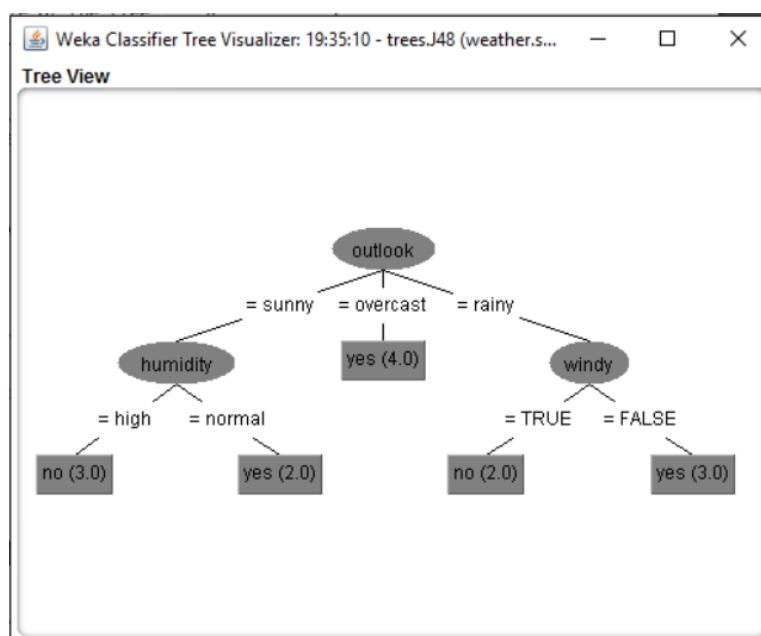
Nos muestra



- Donde muestra los valores por defecto del clasificador, 0.25 y 2 que raramente hay que cambiar ya que proporcionan buenos resultados. Vamos a marcar “**Use training set**” dentro de las **Test options** y ejecutamos haciendo clic en el botón “**start**” y obtenemos el resultado. Vemos que muestra bastante información siendo una de ellas la matriz de confusión bastante útil como se vio en la unidad correspondiente



- Haciendo click en el botón derecho en la lista inferior de la izquierda se muestra un menú; seleccionamos “**Visualize tree**” y se muestra gráficamente el árbol obtenido



- En el mismo menú anterior podemos ver “**Visualize classifier errors**” y verificar los errores que en este caso todas las instalaciones se clasifican correctamente
- Resumiendo, hemos visto lo ponente de esta herramienta y lo transparente que hace la minería de datos ya que abstrae al “**data scientist**” del código subyacente para centrarse en probar y estudiar los resultados.

Recuerda

1

El campo del **aprendizaje automático** y la **inteligencia artificial** ha ganado cada vez más popularidad en los últimos años. Como Big Data es la tendencia más popular en la industria tecnológica en este momento, el aprendizaje automático es increíblemente poderoso para hacer predicciones o sugerencias calculadas basadas en grandes cantidades de datos.

Los **algoritmos de aprendizaje automático** se pueden dividir en 3 grandes categorías: Aprendizaje supervisado, aprendizaje no supervisado y aprendizaje de refuerzo.

2

La **lógica difusa** se utiliza cuando la complejidad de nuestro proceso es muy alta y no van a existir modelo matemáticos precisos que nos sirvan para nuestras conclusiones.

Cuando estamos realizando una **búsqueda** dentro de un problema, podremos diferenciar dos tipos principales, y es realizar una búsqueda cuando tenemos toda la información del problema o realizarla sin tener la información completa.

3

Los **Algoritmos Genéticos** son métodos adaptativos que pueden ser utilizados para resolver problemas de búsqueda y optimización.

El **algoritmo minimax** es un método de decisión que nos permite minimizar la pérdida máxima esperada en juegos con adversario e información perfecta.

4

Un **algoritmo voraz** o también conocidos como Algoritmos Greedy, son aquellos que para resolver un determinado problema siguen una heurística que consiste en elegir la opción más óptima en cada paso local, esperando que se llegue a una solución general óptima.

Descenso gradiente intentará siempre buscar el punto mínimo de una función.

5

6

La Búsqueda Tabú es un procedimiento metaheurístico cuya característica distintiva es el uso de memoria adaptativa y estrategias especiales para la resolución de problemas.

La metaoptimización se basa en el uso de un método de optimización para afinar otro método de optimización.

Los **sistemas multiagente o SMA**, son sistemas que están compuestos por varios sistemas inteligentes los cuales interactúan entre ellos.

7

En el **algoritmo Dijkstra** se realiza un cálculo de los caminos mínimos entre dos nodos del grafo, se calcula el rendimiento dependiendo de la velocidad y la estructura de datos y al ser una búsqueda no informada, explora todo el grafo.

Respuesta: El aprendizaje automático (Machine Learning) también es una parte central de la IA. El aprendizaje sin ningún tipo de supervisión requiere la capacidad de identificar patrones en flujos de entrada, mientras que el aprendizaje con supervisión adecuada implica clasificación y regresiones numéricas.

Respuesta: La clasificación determina la categoría a la que pertenece un objeto.

Respuesta: La regresión se encarga de la obtención de un conjunto de ejemplos numéricos de entrada o salida.

Respuesta: La percepción automática (Machine perception) es la capacidad de usar entradas sensoriales para deducir los diferentes aspectos del mundo.