Tema 2

Determinación de sistemas de aprendizaje automático (machine learning)

Diccionario de términos de aprendizaje automático

- **Conjunto de datos (dataset)**: son los datos que se emplean para, junto con los algoritmos, crear las reglas que permiten predecir comportamientos. Se pueden organizar en un cierto orden o bien como simples documentos de texto no estructurados.
- **Variables descriptivas**: almacenan informacion con un cierto tipo de datos
- **Etiquetas**: corresponden a los atributos que se quieren predecir
- **Tipos de datos**: son los valores que pueden tomar las variables. Pueden ser:
  - **numéricos**: corresponden a los números reales
  - **categóricos**: corresponden a cadenas de caracteres que identifican categorías – "alto", "bajo", por ejemplo-
  - **ordinales:** son cadenas de caracteres ordenadas, que tambien corresponden a categorias, pero en este caso ordenadas.
- **Modelo**: es el producto del entrenamiento de los datos con los algoritmos de aprendizaje y los datos de test, y que permite hacer predicciones al aplicarlo a nuevos datos.

Fuentes de datos

Existen multiples fuentes de datos reales con los que trabajar. Entre ellas podemos destacar

- Kaggle: https://www.kaggle.com/datasets
- AWS de Amazon: https://registry.opendata.aws

Conjuntos de datos de Scikit-learn, una librería de Python que se emplea en Aprendizaje automático

1. Introducción al aprendizaje automático

1.1. ¿Qué es el aprendizaje automático?

Existen múltiples definiciones del aprendizaje automático, entre ellas las de Samuel – "aprendizaje automático es el campo de estudio que da al computador la habilidad de aprender sin haber sido explícitamente programado para ello"-, pero todas ellas coinciden en lo esencial, y es que las herramientas de aprendizaje automático permiten que un computador aprenda de los datos y empleando una serie de algoritmos -diferentes según el tipo de problema a resolver- cree un conjunto de reglas que le permiten detectar patrones de comportamiento en los nuevos datos.

Todo proceso de aprendizaje automático tiene una serie de etapas que pasamos a enumerar y describir

1.2. Etapas del aprendizaje automático

1.2.1. Selección del algoritmo de aprendizaje supervisado

El primer paso es identificar los algoritmos de aprendizaje que vamos a emplear, empezando por los algoritmos puramente estadísticos para limpieza y preprocesamiento de datos, seguidos de los algoritmos de reducción de componentes, y a continuación los algoritmos de aprendizaje automático propiamente dichos.

1.2.2. Obtención y preprocesamiento de datos

En esta fase se recopila la información de diferentes fuentes, tanto información proporcionada por expertos como almacenada en bases de datos u otras fuentes en la Red, se identifican los atributos relevantes y se aplican las técnicas de muestreo. Así mismo se emplean los algoritmos estadísticos indicados anteriormente para las tareas previas de limpieza y formateo de datos -discretización-, incluyendo el tratamiento de los nulos -que puede alterar el funcionamiento de los algoritmos- a través de técnicas de imputación, así como los valores atípicos, que se deben estudiar a partir de técnicas de tipo exploratorio para ver su influencia en los modelos y que se pueden detectar estableciendo valores máximos y mínimos de los mismos, o comprobando si la diferencia entre media y mediana -varianza- de una variable es muy grande, o eliminando el ruido -procesamiento de imágenes-. Una vez hecho esto se deben normalizar los datos para que todas las variables tengan pesos equivalentes en el modelo. Por último, se emple

-arán las técnicas de reducción de dimensiones para facilitar las tareas.

1.2.3. División en conjunto de entrenamiento y prueba

En esta fase se dividen los datos en dos conjuntos diferentes.

Uno, el de entrenamiento -un 70-80 por ciento de los datos-, se utiliza junto con los algoritmos para crear las reglas que nos permiten detectar patrones. El otro, el de prueba -un 20-30 por ciento de los datos-, se utiliza para probar las reglas generadas.

Una alternativa para evitar el sobreajuste -esto es, que el modelo no reacciona bien ante datos nuevos porque se ajustó demasiado a los de entrenamiento- es dividir el conjunto de datos en tres, datos de entrenamiento, de validación -que se usan para evitar el sobreajuste- y de prueba.

Otra alternativa es la validación cruzada. En este método se divide el conjunto de datos en diferentes particiones y se calcula la media de los resultados del modelo con cada partición. Existen varios métodos de validación cruzada, de los que el más conocido es K-folds. El método K-folds consiste en dividir el subconjunto formado por los datos de entrenamiento y validación en varios subconjuntos llamados k-folds -en los que cada subconjunto tiene un bloque de datos para entrenamiento y varios para validación-. Esto nos permite obtener diferentes valores que se promedian para obtener un valor de salida. Existen otros métodos de validación cruzada como Random o Leave-one-out, que comentamos someramente en el cuadro de la derecha.

1.2.4. Configuración del algoritmo

Se implementa aquí el algoritmo de aprendizaje automático que se decidió utilizar y se definen los parámetros – que no son determinados por el ingeniero si no por el propio modelo durante el aprendizaje- e hiperparámetros -parámetros creados por el propio modelo durante el entrenamiento cuyos valores no genera el experto humano, pero que si puede ajustar para mejorar la respuesta del modelo a partir de unos valores genéricos o bien de la experiencia, ajustándolos para mejorar el rendimiento del modelo, tales como el tamaño de los conjuntos de validación y test-.

1.2.5. Entrenamiento del modelo

Consiste en emplear el conjunto de entrenamiento para obte

Esquema de las fases del aprendizaje automático

Validación cruzada random: en este caso el conjunto de datos de prueba y de entrenamiento se divide de manera aleatoria, en vez de en bloques como en K-fold

Validación Leave-one-out: en cada subconjunto k solo hay un dato de test y el resto son de entrenamiento

-ner las reglas del modelo de aprendizaje automático. Para ello emplearemos los algoritmos de aprendizaje automático que hemos preparado en el apartado anterior.

1.2.6. Prueba del modelo

Aquí se emplean los datos de prueba/test para predecir comportamientos usando las reglas del modelo que se ha creado.

1.2.7. Evaluación y optimización del modelo

Permite determinar el grado de acierto del modelo en la predicción de comportamientos. Se emplean técnicas como la matriz de confianza y otras que veremos a lo largo de estas páginas. Una vez aplicadas se pueden modificar los hiperparámetros para evitar los problemas de subajuste -debido a una diferencia muy grande entre valor predicho y valor real, o sesgo- y sobreajuste -donde la varianza, como hemos comentado antes, es muy alta. Entre ellos está el número de veces que iteramos los modelos -epochs-, o el gradiente -la diferencia entre valores consecutivos-. Así mismo se analizarán las variables de entrada y salida y una función objetivo cuyo valor debe minimizarse. Una vez ajustado el modelo debe validarse con datos diferentes a los empleados en la fase de ajuste.

1 epoch = Es una iteración completa de los datos, el modelo aprende de todos los datos una vez.

Si se entrenan 100 epochs, el modelo verá el conjunto de datos 100 veces.

A más epochs, mayor aprendizaje… pero también mayor riesgo de sobreajuste (overfitting).

1.2.8. Exportación y despliegue del modelo

Una vez creado el modelo y evaluado se guarda en el equipo y se utiliza cuando se necesite sin tener que pasar por todas las fases de nuevo.

Como alternativa se puede integrar en una aplicación existente. Para ello es preciso que se proporcionen los datos adecuados al modelo que se ha implementado, tanto de fuentes modernas como antiguas, para así poder tener una imagen lo más completa posible, y que estos sean de alta calidad y sean relevantes.

Así mismo se deben automatizar todos los procesos que sea posible para las pruebas y la capacitación del modelo y verificar de manera periódica los resultados del modelo para comprobar que son los esperados.

Esto implica que debe comprobarse continuamente los datos que llegan para verificar si el modelo es aún útil.

Por otro lado, se debe verificar que la aplicación en la que se inserta el modelo tiene un elevado grado de usabilidad y abstracción para facilitar su uso.

Video Diferencia entre aprendizaje supervisado y No supervisado
https://www.youtube.com/watch?v=d1tVOcZwM3s

2. Clasificación de sistemas de aprendizaje automático

2.1. Aprendizaje supervisado

2.1.1. Concepto

En el aprendizaje supervisado todos los registros de datos se etiquetan con un valor resultado correcto, o una categoría objetivo conocida, facilitando la creación de un patrón y la clasificación de los nuevos registros de datos.

2.1.2. Métodos de aprendizaje supervisado

Podemos distinguir dos métodos dentro del aprendizaje supervisado

- *Regresión*: la regresión permite determinar un valor a partir de una serie de valores dados, prediciendo los valores futuros a partir de una serie de datos ya existentes.
- *Clasificación*: en este caso se trata de clasificar los datos según etiquetas que corresponden a valores discretos o categorías.

2.1.3. Algoritmos de aprendizaje supervisado

Dentro de los algoritmos de aprendizaje automático supervisado podemos distinguir los siguientes:

- *Regresión*: existen diferentes algoritmos que implementan regresión: lineal, polinómica o de K vecinos más próximos
- *Análisis discriminante lineal*: es un método de extracción de características para conseguir reducción de la dimensionalidad -esto es, disminuir el número de variables de entrada que se van a analizar en el conjunto de datos eliminando aquellas no importantes-
- *K vecinos más próximos*: algoritmo de clasificación que se basa en buscar los k registros de datos más próximos al que queremos clasificar, cada uno de los cuales tiene una etiqueta que lo asocia a un categoría o grupo determinado, y devuelve una etiqueta que indica a que grupo pertenece el registro. Este algoritmo requiere la definición de una distancia entre registros, que puede ser la distancia euclídea, la de Manhattan o la de Minkowsky.
- *Máquinas lineales de vectores soporte*: clasifican un conjunto de datos separándolos por un hiperplano -que no es sino un plano de n-dimensiones, tantas como variables tengan los registros de datos, y que maximizará el espacio entre ambos conjuntos- en dos grupos - +1 y -1-

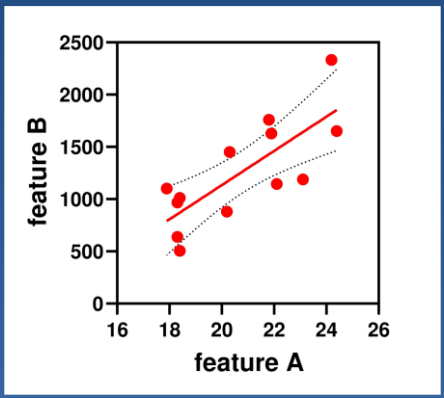
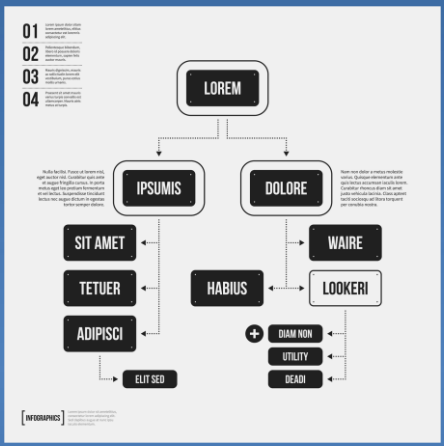


Imagen de una regresión lineal



Ejemplo de clasificación con árboles de decisión

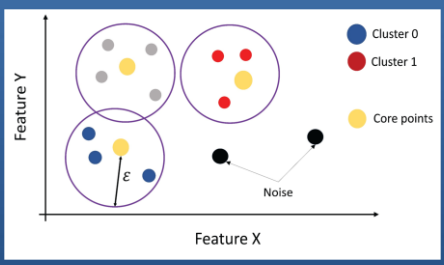
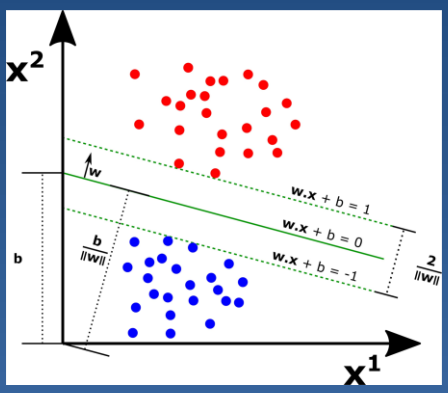
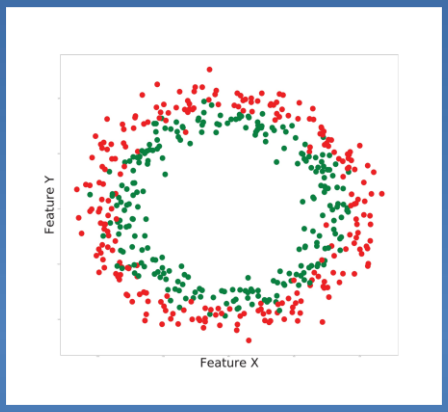


Imagen de clasificación de K vecinos más próximos



Clasificación con maquina de vectores soporte



Clasificación por regresión logística de dos características

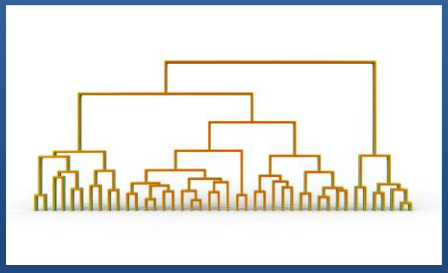


Imagen de un dendograma (diagrama de clusterización en aprendizaje no supervisado)

Los vectores soporte son los registros de datos que están en el límite entre los dos conjuntos. Existen también las máquinas no lineales de vectores soporte, pero están más allá de los objetivos de este módulo.

- *Árboles de decisión*: es un sistema de clasificación en el cual tenemos un árbol cada uno de cuyos nodos es una propiedad o regla que permite evaluar los valores de los atributos de cada registro de datos, y en función de dicha evaluación los registros se separan en las diferentes ramas, y así hasta que lleguemos a los nodos hojas, donde se clasifica la muestra.
- *Clasificador bayesiano ingenuo*: lo hemos estudiado en el módulo de Modelos de Inteligencia Artificial ,y emplea la teoría de la probabilidad para clasificar datos.
- *Regresión logística*: recibe una función lineal como entrada y como salida un valor en [0,1] que es la probabilidad de pertenencia a una u otra clase.

2.2. Aprendizaje no supervisado

2.2.1. Concepto

En el caso del aprendizaje no supervisado no tenemos etiquetas resultado /variables de salida por lo que los algoritmos se centran en buscar patrones entre el conjunto de datos directamente para clasificar.

2.2.2. Métodos de aprendizaje no supervisado

Dentro del aprendizaje no supervisado tenemos dos tipos de métodos:

- *Métodos partitivos*: que parten del conjunto de datos como una única categoría y la va dividiendo en varias categorías cada vez más específicas
- *Métodos aglomerativos*: se considera cada registro de datos como una categoría independiente y se van agrupando hasta obtener las categorías finales que corresponden a los patrones encontrados

2.2.3. Algoritmos de aprendizaje no supervisado

Los algoritmos de aprendizaje no supervisado más empleados son:

- *Análisis de componentes principales*: es un método de aprendizaje no supervisado que permite reducir la dimensionalidad de los datos al transformarlos -rotarlos-, y

que no estén correlacionados entre sí.

- *K-means*: es un conjunto de algoritmos que buscan patrones en los datos para luego agruparlos según sus elementos comunes. El algoritmo minimiza las distancias de las muestras dentro de cada grupo o cluster.
- *Algoritmos aglomerativos*: parte de un número de clusters igual al de registros, y los une en función de su proximidad hasta alcanzar los limites marcados por las condiciones del sistema. Deben estar cohesionados internamente y separados unos de otros.
- *Algoritmos de detección de anomalías*: pueden ser de los siguientes tipos:
  - SVM de una clase
  - Isolation forest
  - DBSCAN
  - HBOS
  - Factor anómalo -y variantes-

Los algoritmos de detección de anomalías permiten entrenar el sistema en la identificación de elementos "normales" para de esta manera identificar los anómalos o los nuevos.

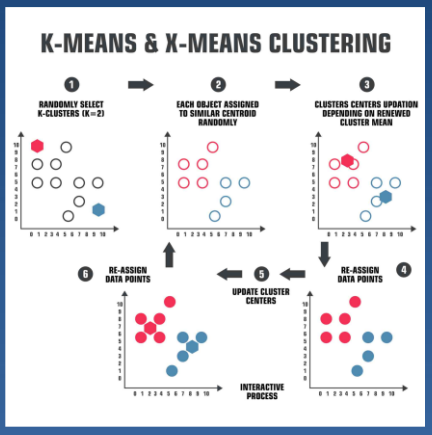
2.3. Aprendizaje semisupervisado

Consiste en emplear a la vez datos etiquetados y no etiquetados – más de los segundos que de los primeros- para mejorar la exactitud de los modelos, considerando que los datos no etiquetados agrupados entre sí corresponden a una etiqueta determinada.

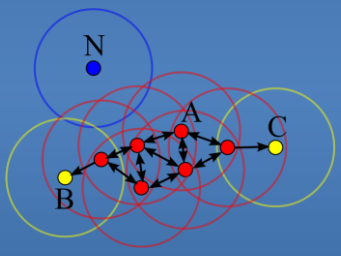
Los algoritmos que emplea son combinaciones de algoritmos supervisados y no supervisados.

2.4. Aprendizaje por refuerzo

Se consigue a través de ensayo-error aplicando el aprendizaje por refuerzo, o retroalimentación, a través de agentes inteligentes, que observan el entorno y toman decisiones que pueden ser acertadas -y reciben una recompensa- o erróneas -y reciben un castigo- a partir de las cuales crean una política que aplican cuando tienen que tomar una decisión determinada. Se usa para resolver problemas en los que los datos llegan continuamente durante un largo periodo de tiempo.

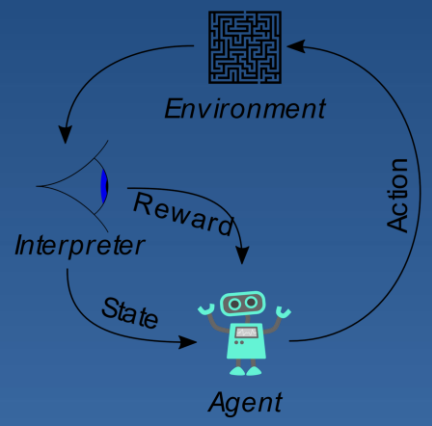


Esquema de funcionamiento de K Means



Ejemplo de algoritmo DBSCAN de detección de anomalias (De Chire - Trabajo propio, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17045963)

Los puntos A son parte del nucleo, los B y C son densamente alcanzables desde el nucleo y N es ruido.



Esquema del funcionamiento de un agente inteligente

Determinación de la edad de los huesos de una mano empleando aprendizaje automático

2.5. Otros criterios de clasificación del aprendizaje automático

2.5.1. Aprendizaje por lotes frente a aprendizaje on-line

El aprendizaje por lotes se produce cuando entrenar un modelo requiere todos los datos disponibles y una gran cantidad de recursos computacionales y físicos, lo cual implica que primero se cree el modelo y luego se ejecute pero sin que pueda seguir aprendiendo debido a ese coste. Es un sistema poco útil cuando es preciso un aprendizaje continuo y que el modelo se actualice rápidamente.

El aprendizaje online permite entrenar los modelos de manera gradual introduciendo los datos poco a poco y es útil cuando tenemos un flujo de datos continuo pero no masivo, y que requiere que los modelos se actualicen constantemente, pero tiene el problema de que si el sistema se adapta demasiado rápido a los cambios puede ser sensible a alteraciones puntuales que no corresponden a la secuencia normal -valores fuera de rango- y el rendimiento del sistema, su capacidad de predicción, disminuirá notablemente.

2.5.2. Aprendizaje basado en instancias frente a aprendizaje basado en modelos

Son dos modelos de aprendizaje con ligeras diferencias.

El aprendizaje basado en instancias aprende a partir de una serie de ejemplos que tiene almacenados en memoria y luego reconoce otros casos por similitud.

Por su parte el aprendizaje basado en modelos genera un modelo a partir de una serie de ejemplos y luego entrenaos los datos de ese modelo y lo aplicamos para hacer predicciones.

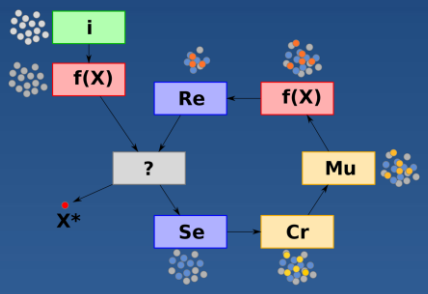
3. Técnicas de aprendizaje automática

Dentro de las técnicas empleadas en aprendizaje automático podemos encontrar:

- *Redes neuronales*: se basan en el concepto de neurona artificial, por analogía con las neuronas del cerebro. Reciben estímulos del exterior, que procesan a través de una función con parámetros esos estímulos y los pesos.
- *Aprendizaje basado en casos*: emplea un conjunto de casos almacenados en una base de caso de entre los que selecciona los que se parecen al caso que se plantea co-

como problema a resolver y los adapta a la situación concreta empleando reglas basadas en el conocimiento sobre el dominio. Una vez resuelto el caso se añade a la base de datos de casos.

- *Razonamiento inductivo*: emplea una serie de ejemplos para construir un conjunto de reglas del tipo SI…ENTONCES. Corresponden a algoritmos del tipo de los árboles de decisión o los sistemas basados en reglas que se han visto en el módulo Modelos de Inteligencia Artificial
- *Razonamiento evolutivo*: emplea algoritmos genéticos, que se han explicado de manera somera en el módulo de Modelos de Inteligencia artificial. A partir del cruce entre sí de las soluciones iniciales permite obtener las definitivas.

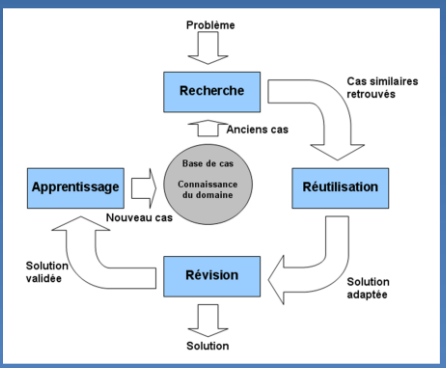


Ejemplo de las fases de un algoritmo genético

4. Dificultades asociadas a los problemas de aprendizaje automático

Entre los problemas que surgen a la hora de generar los modelos de aprendizaje automático podemos destacar:

- *Cantidad insuficiente de datos de entrenamiento*: se pueden necesitar millones de registros de datos para entrenar un modelo correctamente, para lo cual se usan técnicas como el aprendizaje por transferencia, en el que se usan modelos previamente entrenados con datos o imágenes genéricas -líneas, puntos, formas básicas-, para luego entrenarlos con la información que tenemos, limitando así la cantidad de datos propios necesarios.
- Datos de entrenamiento que no corresponden fielmente al universo de ejemplos: provoca que el modelo resultante no puede predecir correctamente al estar sesgados los datos de entrenamiento bien porque se hayan seleccionado más datos de unas categorías que de otras sin que ello sea reflejo de su distribución natural -sesgo muestral- o bien por la variabilidad natural de la muestra.
- *Datos de mala calidad*: es semejante al problema anterior, puede deberse a valores fuera de rango por fallos puntuales del sistema de recogida de datos, o registros de datos incompletos porque falte algún atributo, y se puede corregir con técnicas de tratamiento de datos.
- *Atributos poco importantes*: al recoger la información de los ejemplos, los registros de datos, puede suceder que ha



Esquema del razonamiento basado en casos5

Esquema de la metodología CRISP-DM

One hot encoding

Permite convertir una variable categórica en un conjunto de columnas de datos binarios, tantas como valores admite la variable categórica. En esas columnas, cada una tiene un nombre referido al valor que toma la variable categórica, y tendra un 1 en la columna cuyo nombre coincide con el valor de la variable categorica, mientras que el resto de las columnas están a 0.

Ejemplo : Variable categorica: estatura {alta, media, baja}

Tabla original

Estatura
Alta
Media
baja

Conversion a binario

estatura	Estatura_alta	Estatura_media	Estatura_baja
Alta	1	0	0
Media	0	1	0
baja	0	0	1

Normalización

Permite transformar una distribucion de datos original en otra normalizada con los valores resultantes circunscritos al intervalo [0,1]. Se emplea cuando la mayor parte de los atributos son ordinales o nominales.

$$x_i = \frac{x_i - min(X)}{max(X) - min(X)}$$

-ya atributos que se recogen que no tienen información relevante, para lo cual tenemos que aplicar las técnicas de selección de características como la reducción de dimensionalidad (PCA), o la extracción de características, que implica la combinación de características.

- *Generalizar mal a partir de pocos datos*: recibe el nombre de sobreajuste e implica que el modelo responderá mal ante datos nuevos que no correspondan con los que se han usado en el entrenamiento. Es debido a la complejidad del problema a evaluar que hace que no generalice bien.
- *Generalizar creando un modelo demasiado simple que no corresponde a la realidad*: recibe el nombre de subajuste y se produce al no haber entrenado el modelo con suficientes datos por lo que no puede identificar la función que relaciona las variables descriptivas o de entrada con la de salida o destino.

5. La metodología CRISP-DM

CRISP-DM – Cross Industry Standard Process for Data Mining- es el protocolo estándar para la implementación de ML.

Tiene las siguientes fases:

- *Conocer el dominio o ámbito en el que se debe resolver el problema*, que debe ser detallado correctamente para encuadrar la solución en un ámbito concreto del aprendizaje automático. También es importante saber cuál es el método actual para resolver el problema, para determinar que método de aprendizaje automático es el más adecuado.

  En esta fase es importante verificar todas las hipótesis que se hacen sobre el ámbito de conocimiento en el que se enmarca el sistema con el que tratamos.

- *Conocimiento de los datos*: esto es, conocer el origen de los datos que se van a emplear y su estructura, así como el tipo de datos de cada variable o atributo de los registros de datos de entrada. Previamente es preciso tener instaladas las herramientas necesarias para obtener esos datos, y descargarlos.

- *Preparación de los datos*: esta etapa permite dividir el conjunto de datos en uno de prueba, otro de test, y en los proyectos profesionales otro de validación previo al de test. Todos son representativos del problema a estudiar.

Así mismo en esta fase se pueden visualizar los datos a través de herramientas gráficas, así como llevar a cabo la tarea de preprocesado de datos para limpiarlos de datos no útiles, repetidos o incompletos permite generar un modelo útil para las predicciones. Así mismo en esta etapa se debe transformar las variables de tipo categórico en binarias, normalizar las variables numéricas para que estén en el mismo rango o sustituir los valores vacíos por valores predichos

- *Modelado del sistema*: para esto se probarán diferentes modelos hasta encontrar el que se ajusta al resultado buscado.

  Estos modelos se pueden perfeccionar a través de técnicas de búsqueda exhaustiva -modificar todos los hiperparámetros- o aleatorizada -solo se modifican combinaciones aleatorias de hiperparámetros-.

  También se pueden combinar para obtener un mejor rendimiento.

- *Evaluación*: se prueba el modelo con los datos de test para verificar que predice correctamente los comportamientos del sistema estudiado. Para ello se emplean las funciones o métricas de coste, que miden la diferencia entre los resultados obtenidos por el modelo y los predichos para medir la distancia entre los mismos, la cual debe de ser mínima para que el modelo sea válido. Algunas de esas métricas como el error medio absoluto (MAE) o el error cuadrático medio (MSE) se indican en el cuadro lateral izquierdo, y se estudiarán con más profundidad en el tema 3.

- *Despliegue*: se integra el modelo en la organización poniéndolo en producción, bien a través de un sitio web que ofrezca el servicio como un servicio web -API REST-, o se puede desplegar en la nube. Asimismo se tiene que comprobar en vivo el rendimiento de este sistema para verificar que no disminuye, bien a través de código -que modifique los hiperparámetros o recopile nuevos datos e informe a través de una serie de alertas de pérdidas de rendimiento del sistema- o de expertos humanos.

6. Herramientas para aprendizaje automático

Existen múltiples herramientas que se pueden emplear en aprendizaje automático. Aquí nos centraremos en las que vamos a emplear en este curso:

Estandarización o normalización z-score

Consiste en transformar una distribución de datos original en una distribución normal estándar en la que la media sea 0 y la desviación estándar sea 1 y las proporciones entre los datos no cambien respecto a la distribución original. Se emplea cuando los atributos son mayoritariamente numéricos o categóricos. La expresión que se utiliza es

$$x_i = \frac{x_i - \mu(X)}{\sigma(X)}$$

Error Cuadrático Medio

$$MSE(X,y) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - p(X_i))^2$$

donde $p(X_i)$ es el valor predicho de la muestra i de X ,e y_i, el valor objetivo de la muestra.

Error Medio absoluto

$$MAE(X,y) = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - p(X_i)|$$

Logo de Python. De www.python.org - www.python.org, GPL, https://commons.wikimedia.org/w/index.php?curid=34991651

Funciones de matplotlib

Función	Utilidad
plt.plot(X,Y,etiqueta)	Grafica Y frente a X -dos arrays- y le asigna una etiqueta
plt.title(titulo)	Pone un título a la grafica
plt.xlabel(etiqueta)	Etiqueta para el eje X
plt.ylabel(etiqueta)	Etiqueta para el eje Y
plt.legend()	leyenda
Plt.bar(X,Y,etiqueta,color, align)	Crea un diagrama de X barras, con Y alturas
plt.show()	Muestra la gráfica

Importación de bibliotecas

Se emplea el comando:

import nombre_biblioteca as nombre

Ejemplos: **import pandas as pd, import numpy as np, import matplotlib.pyplot as plt**

Funciones de Pandas

Funcion	utilidad
df=pd.read_csv(ruta)	Abre un .csv almacenado en un archivo en la ruta indicada y lo carga en un conjunto de datos
print(df)	Muestra por pantalla un conjunto de datos
df.head()	Muestra las primeras cinco filas del conjunto de datos
df.columns	Muestra las columnas del conjunto de datos
df.shape	Devuelve filas y columnas del conjunto de datos
df.iloc[x] | df.iloc[:x] | df.iloc[0,1] |df.iloc[x_1,x_2..],0:i]|df.iloc[x,y]| df.iloc[x,'n_col']	Devuelve la fila en posicion x| las x primeras filas| las primeras i columnas de las filas x_1, x_2, etc...| la celda en posicion (x,y)|la columna de nombre n_col
df.describe()	Descripcion de los datos de un conjunto de datos
df.drop(columns=['n_col'] |df.drop_duplicate()	Borra la columna de nombre n_col
Df.dropna() |df.drop(subset=[lista parametros] | df.dropna(how="all")	Borra los datos desconocidos de todos el conjunto de datos, de una o de varias columnas o de todo un registro
Df.fillna(parametros)	Sustituye los valores NaN por un valor introducido por parámetro
df.to_csv('n_archivo.csv', index=False)	Guarda el conjunto de datos en un .csv sin una columna de indice -index=False

Funciones de Numpy

Función	Utilidad
v=np.array([x_1,x_2,...,x_i])	Crea un array de i elementos
v.shape()	Muestra las dimensiones del array
v=np.array([x_1,x_2,...,x_i][y_1,y_2,...y_j])	Crea una matriz de ixj elementos
v=np.zeros(N)	Crea una matriz de NXN ceros
v=np.random.random(x,y)	Crea una matriz de números reales aleatorios entre 0 y 1
v=np.array(lista)	Define un array a partir de una lista de Python
v=np.arrange(N)	Crea un array de números secuenciales
matriz=v.reshape(i,j)	Crea una matriz a partir de un vector o de otra con i filas y j columnas
+|-|*/multiply(X,Y)| / divide(X,Y)	Operaciones con matrices

- *Python*: es un lenguaje de programación no tipado, con una sintaxis sencilla y que se emplea en aprendizaje automático. Python tiene una serie de librerías con algoritmos muy potentes entre los que están:
  - *Numpy*: permite el manejo de arrays a través de una serie de funciones
  - *Scikit-learn*: contiene funciones que implementan diferentes algoritmos de aprendizaje automático supervisado y no supervisado, así como un conjunto de conjuntos de datos predefinidos
  - *Pandas*: permite cargar archivos de datos en diferentes formatos, visualizarlos y gestionarlos.
  - *Matplotlib*: permite visualizar los datos de manera gráfica así como ver el rendimiento de los modelos

  En el apartado de prácticas podemos ver ejemplos del uso de algunas de estas librerías.

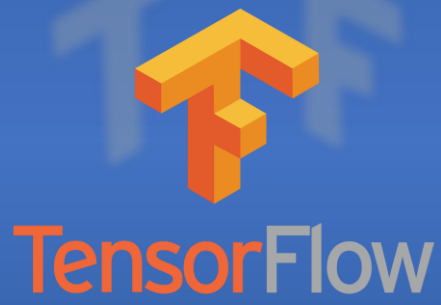
  Python puede instalarse en el equipo, con distribuciones como Anaconda, que incluye las anteriores librerías, así como un IDE propio, un gestor de paquetes -conda- y un editor interactivo -Jupyter Notebook-
- *Google Colab*: es un entorno de programación web semejante a Jupyter Notebook que permite usar máquinas virtuales que permiten utilizar GPUs. Puede utilizar numpy, scikit-learn, matplotlib o pandas entre otras.
- *TensorFlow*: es una biblioteca de Google de programación numérica dedicada al Deep Learning, semejante a Numpy y que permite la computación distribuida, con compiladores just in time que optimiza los cálculos y la ejecución. Emplea a bajo nivel C++ y tiene diferentes implementaciones o kernels para cada operación, en función de que se ejecute en una CPU, una GPU o una unidad de procesamiento tensorial o TPU. Tiene una serie de APIS de alto nivel como Keras, y otras de bajo nivel en Python, Java, C++ y otros lenguajes como JavaScript. TensorFlow puede ser utilizado en Colab.
- *Keras*: es un API de alto nivel que permite realizar proyectos de aprendizaje profundo o Deep Learning con redes neuronales. TensorFlow tiene una implementación de Keras.

7. Aplicaciones del aprendizaje automático

Entre las aplicaciones del aprendizaje automático podemos destacar las siguientes:

- Clasificar productos en una cadena de montaje -redes neuronales convolucionales-
- Detección de tumores empleando imagen artificial generada por computador -redes neuronales convolucionales-
- Clasificación de textos -procesamiento del lenguaje natural, redes neuronales convolucionales o recurrentes-
- Realizar tareas de inteligencia de negocio – rendimiento de inversiones, ingresos por ventas, etc… con regresión, redes neuronales convolucionales o recurrentes
- Reconocimiento de voz -redes neuronales convolucionales o recurrentes-
- Marketing orientado al cliente – agrupamiento, clusters, redes neuronales-
- Crear jugadores virtuales para un juego -agentes inteligentes-

Logo de Keras



Logo de TensorFlow