**Curso: tópicos de inteligencia artificial**

**Preprocesamiento de datos**

¿Qué incluye la Preparación de Datos? El Preprocesamiento de Datos engloba a todas aquellas técnicas de análisis de datos que permite mejorar la calidad de un conjunto de datos de modo que las técnicas de extracción de conocimiento/minería de datos puedan obtener mayor y mejor información (mejor porcentaje de clasificación, reglas con más completitud, etc.)

Los datos reales pueden ser impuros, pueden conducir a la extracción de patrones/reglas poco útiles. Esto se puede deber a:

* Datos Incompletos: falta de valores de atributos
* Datos con Ruido
* Datos inconsistentes (incluyendo discrepancias)

La preparación de datos puede generar un conjunto de datos más pequeño que el original, lo cual puede mejorar la eficiencia del proceso de Minería de Datos.

El preprocesamiento puede incluye:

* Selección relevante de datos: eliminando registros duplicados, eliminando anomalías
* Reducción de Datos: Selección de características, muestreo o selección de instancias, discretización.

La preparación de datos genera “datos de calidad”, los cuales pueden conducir a patrones/reglas de calidad. Por ejemplo, se puede:

* Recuperar información incompleta.
* Eliminar outliers
* Resolver conflictos

Importar datos desde un archivo csv.

#importar dataset  
dataset=pd.read\_csv("Data.csv")

Dividir los conjuntos de información en diferentes vectores arreglos de datos. En Python las matrices se especifican como [fila,columna].

#conjunto de características  
X=dataset.iloc[:,0:3].values  
#vector de clases  
Y=dataset.iloc[:,3].values

Reemplazar los datos faltantes o nulos por un valor numérico, por lo general se suele reemplazar por la media de todos los datos de la columna (variable) donde faltan los valores, considerando que los datos siguen una distribución normal.

from sklearn.impute import SimpleImputer

from sklearn.compose import ColumnTransformer

#reemplazar datos faltantes con la media de los demás datos  
imputer=SimpleImputer(missing\_values=np.nan, strategy='mean', fill\_value=None, verbose=0, copy=True, add\_indicator=False)  
imputer.fit(X[:, 1:3])  
X[:, 1:3]=imputer.transform(X[:, 1:3])

Una variable categórica es aquella que permite clasificar una serie de datos por medio de valores fijos asociados a una cualidad o categoría concreta. La variable categórica, a diferencia de las variables cardinales o continuas (que permiten cálculos numéricos), clasifica a los individuos o casos. Normalmente toman valores representados por números enteros, como el uno o el cero, pero estos son solo eso, representaciones. Estas variables se pueden representar como variables dummy o indicadoras. Estas variables toman dos valores usualmente, cero y uno. Los dos valores significan que la observación pertenece a una de dos categorías. Las variables dummy o indicadoras sirven para identificar categorías o clase a las que pertenecen las observaciones.

Codificando datos categóricos en numéricos.

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

#Codificando datos categoricos  
labelencoder\_X=LabelEncoder()  
X[:,0]=labelencoder\_X.fit\_transform(X[:,0])

Convertir a variables dummy

#Traducir categoricos a dummy  
ct=ColumnTransformer([("Country", OneHotEncoder(), [0])], remainder="passthrough")  
X=ct.fit\_transform(X)

La estandarización de los datos mejora los resultados cuando los valores entre variables tienen dimensiones muy diferentes. Entre -1 y 1 para que ninguna característica influya más que otra. La estandarización ajusta los datos para que su media sea 0 y la normalización ajusta los valores entre un mínimo y un máximo definido*. La variable Y no se debe escalar para clasificación*.

Graphical user interface, application, table, Word

Description automatically generated

#escalar datos  
sc\_X=StandardScaler()  
X=sc\_X.fit\_transform(X\_train)

Dividir set de entrenamiento en entrenamiento y pruebas.

#separar conjunto de datos  
X\_train, X\_test, Y\_train, Y\_test=train\_test\_split(X,Y,test\_size=0.2, random\_state=0)

**Métricas de evaluación**

**Regresiones**

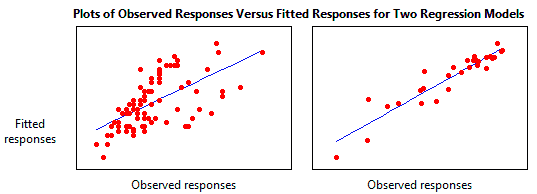
Métrica del

El es una medida estadística de qué tan cerca están los datos de la línea de regresión ajustada. También se conoce como coeficiente de determinación, o coeficiente de determinación múltiple si se trata de regresión múltiple. La definición de es bastante sencilla: es el porcentaje de la variación en la variable de respuesta que es explicado por un modelo lineal. Es decir:

= Variación explicada / variación total

El siempre está entre 0 y 100%:

* 0% indica que el modelo no explica ninguna porción de la variabilidad de los datos de respuesta en torno a su media.
* 100% indica que el modelo explica toda la variabilidad de los datos de respuesta en torno a su media.



El modelo de regresión de la izquierda explica el 38% de la varianza, mientras que el de la derecha explica el 87,4%. Cuanto mayor sea la varianza explicada por el modelo de regresión, más cerca estarán los puntos de los datos de la línea de regresión ajustada. En teoría, si un modelo pudiera explicar el 100% de la varianza, los valores ajustados siempre serían iguales a los valores observados y, por lo tanto, todos los puntos de los datos estarían sobre la línea de regresión ajustada.

* El no puede determinar si las estimaciones y predicciones de los coeficientes están sesgadas, y es por eso por lo que se deben examinar las gráficas de residuos.
* El no indica si un modelo de regresión es adecuado. Se puede tener un valor bajo del para un modelo adecuado o un valor alto del para un modelo que no se ajusta a los datos.

De forma matemática se expresa como:

Métrica del ajustado

ajustado es una medida corregida de bondad de ajuste (precisión de modelo) para los modelos lineales. Identifica el porcentaje de varianza en el campo de destino que se explica por la entrada o las entradas. tiende a estimar de forma optimista el ajuste de la regresión lineal. Siempre aumenta a medida que el número de efectos se incluye en el modelo. ajustado intenta corregir esta sobrestimación. ajustado puede disminuir si un efecto específico no mejora el modelo.

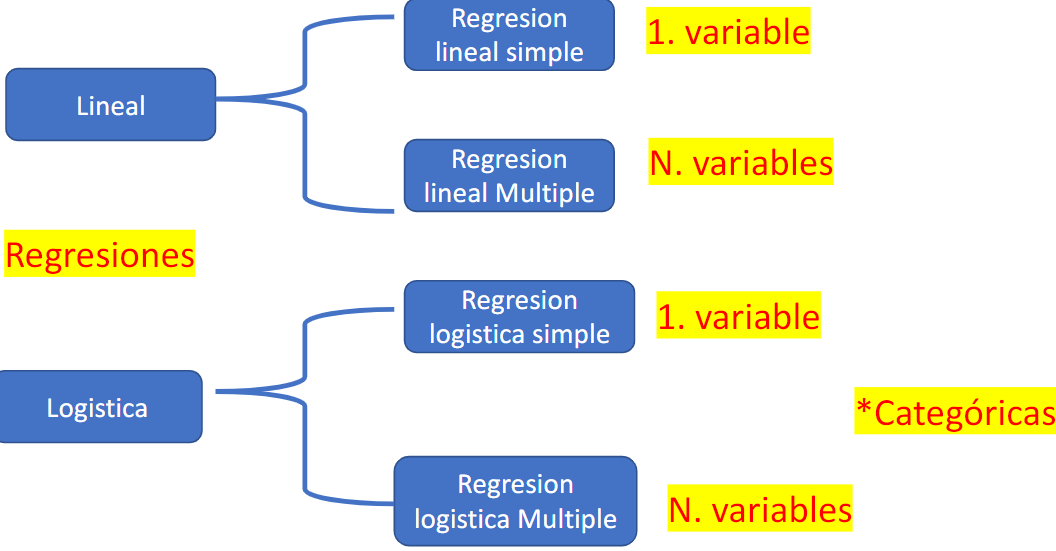
ajustado se calcula dividiendo el error cuadrático medio residual por el error cuadrático total (que es la varianza de muestreo del campo objetivo). A continuación, al resultado se le resta 1. es siempre menor que o igual a . Un valor de 1 indica un modelo que predice perfectamente los valores en el campo de destino. Un valor que es menor o igual que 0 indica un modelo que no tiene ningún valor predictivo. En el mundo real, ajustado se encuentra entre estos valores.

P: Numero de variables independientes(características)

n: Tamaño de la muestra

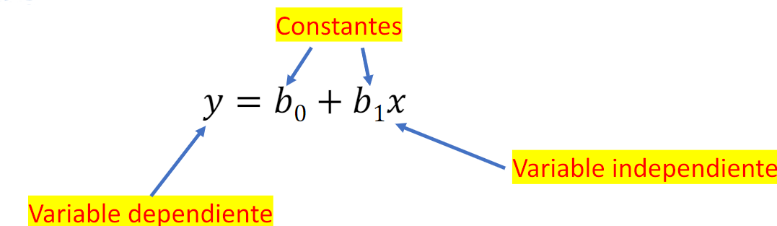
**Regresiones**

El análisis de regresión es una herramienta de frecuente uso en estadística. La cual permite investigar las relaciones entre diferentes variables cuantitativas. Esto, mediante la formulación de ecuaciones matemáticas. Visto de otro modo, dicho análisis es un proceso o modelo que analiza el vínculo entre una variable dependiente y una o varias variables independientes. Así, a partir de dicho estudio, se halla una relación matemática. Una de las principales aplicaciones del análisis de regresión es la proyección con diferentes escenarios. Esto, teniendo en cuenta el grado de influencia (en estadística se conoce a esto como correlación) sobre la variable dependiente.



**Regresión lineal simple**

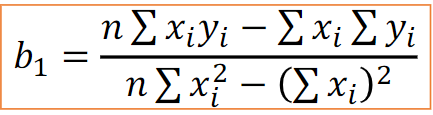
Pretende relacionar una variable dependiente con una o más variable independientes. Supóngase que se tiene un conjunto de n pares de observaciones (xi,yi), se busca encontrar una recta que describa de la mejor manera cada uno de esos pares observados.



Chart, scatter chart

Description automatically generated

El criterio de **mínimos cuadrados** nos proporciona un valor de y uno de y, tal que



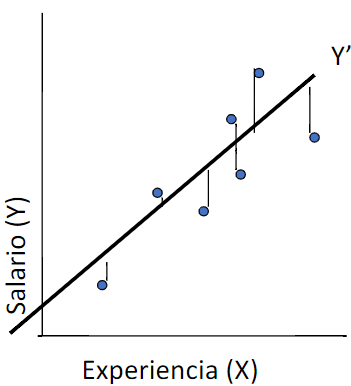
A picture containing text, orange, clock

Description automatically generated

n = número de muestras

= recta propuesta

=punto en el espacio



Entrenamiento y testing de la regresión lineal simple.

from sklearn.linear\_model import LinearRegression

regresion=LinearRegression()  
regresion.fit(X\_train, Y\_train)  
ypred=regresion.predict(X\_test)

Visualización de la regresión lineal.

import matplotlib.pyplot as plt

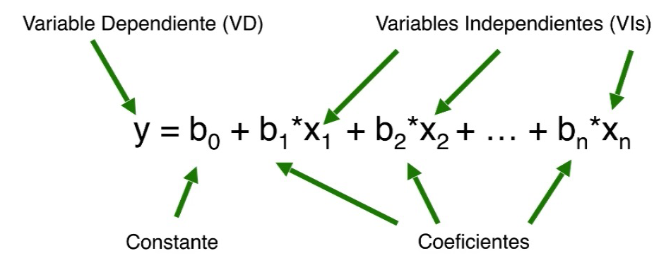
#visualizacion de datos de entrenamiento  
plt.scatter(X\_train, Y\_train, color='red')  
plt.plot(X\_train, regresion.predict(X\_train), color='blue')  
plt.title('Salario vs Experiencia (set entrenamiento)')  
plt.xlabel('Años de experiencia')  
plt.ylabel('Salario')  
plt.grid()  
plt.show()

Chart, scatter chart

Description automatically generated

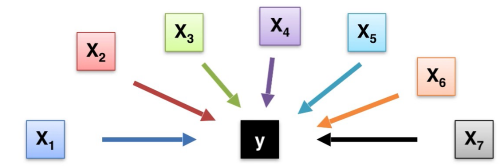
**Regresión lineal múltiple**

Un modelo de regresión lineal múltiple es un modelo estadístico versátil para evaluar las relaciones entre un destino continuo y los predictores. Los predictores pueden ser campos continuos, categóricos o derivados, de modo que las relaciones no lineales también estén soportadas. El modelo es lineal porque consiste en términos de aditivos en los que cada término es un predictor que se multiplica por un coeficiente estimado.



Para la construcción de modelo de regresión múltiple se deben considerar los siguientes factores:

* Quitar las variables que no aporta información relevante
* No por tener más variables se tendrá un mejor resultado



Si se acomodan los datos en forma matricial se tiene que:

La fórmula para encontrar beta es la siguiente:

donde X es la matriz de características y Y el vector del predictor.

Entrenamiento y testing de la regresión lineal múltiple.

from sklearn.linear\_model import LinearRegression

regresion=LinearRegression()  
regresion.fit(X\_train, Y\_train)  
ypred=regresion.predict(X\_test)

El procedimiento de selección de modelos depende de si un predictor categórico está presente o no. Cuando sólo se especifica un predictor continuo, se tienen en cuenta los tres modelos siguientes.

* Un modelo constante que siempre predice la media general.
* El modelo lineal con el predictor único se ha añadido a la constante.
* Modelo cuadrático en el que se añade el predictor cuadrado al modelo lineal.

5 métodos de construir modelos:

1. Exhaustivo (all-in)
2. Eliminación hacia atrás
3. Selección hacia adelante
4. Eliminación Bidireccional (2 y 3)
5. Comparación de Scores

Exhaustivo (all-in)

Se prueban todas las variables del modelo, debido al conocimiento a priori, por obligación o como preparación previa para eliminación hacia atrás.

Eliminación hacia atrás

1. Seleccionar el nivel de significación para permanecer en el modelo (p.e. SL= 0.05)
2. Se calcula el modelo con todas las posibles variables predictoras (All-in)
3. Considera la variable predictora con el p-valor más grande. Si P>SL, entonces vamos al PASO 4, si no vamos a FIN.
4. Se elimina la variable predictora.
5. Ajustar el modelo sin dicha variable\* (volverá calcular coeficientes)

Eliminación hacia adelante

1. Seleccionar el nivel de significación para permanecer en el modelo (p.e. SL= 0.05)
2. Ajustamos todos los modelos de regresión lineal simple y=f(x) Elegimos el que tiene menor p-valor
3. Conservamos esta variable y ajustamos todos los posibles modelos con una variable extra añadiendo a la(s) que ya tenga(s) el modelo hasta el momento
4. Consideramos la variable predictora con el menor p-valor. Si P>SL, vamos al paso 3 si no a FIN

Selección bidireccional

1. Seleccionar el nivel de significación para permanecer en el modelo para entrar y salir del este (p.e. SLENTER= 0.05, SLSTAY=0.05)
2. Llevar a cabo el siguiente paso de selección hacia adelante (con las nuevas variables con: P<SLENTER para entrar).
3. Llevar a cabo todos los pasos de eliminación hacia atrás (las variables antiguas deben tener P<SLSTAY para quedarse).
4. Consideramos la variable predictora con el menor p-valor. Si P>SL, vamos al paso 3 si no a FIN.

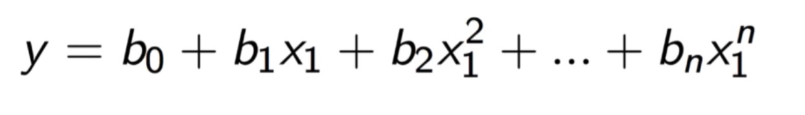
Eliminación hacia atrás antes de aplicar una regresión lineal múltiple para comprobar la importancia de las variables en el modelo.

import statsmodels.api as sm

X\_opt=X[:,[0,1,2,3,4]]  
X\_opt=np.array(X\_opt, dtype=float)  
Regresion\_OLS=sm.OLS(endog=Y, exog=X\_opt).fit()  
print(Regresion\_OLS.summary())

**Regresión Polinomial**

La Regresión Polinomial es un caso especial de la Regresión Lineal y es muy parecido a ella, la diferencia es que los datos acá no son lineales por lo que se debe implementar polinomios de grado n para obtener el modelo. El método estándar para extender la Regresión Lineal a una relación no lineal entre las variables dependientes e independientes ha sido reemplazar el modelo lineal con una función polinomial. Aunque considera que no siempre aumentar el grado del polinomio hará que el modelo mejore, en ocasiones hace más bien que se empeore el algoritmo por lo que esto es un proceso de experimentación para obtener el más adecuado y que reduzca los errores entre el modelo y los datos.



Chart

Description automatically generated

Desafortunadamente, la Regresión Polinomial también tiene un buen número de problemas, a medida que aumentamos la complejidad de la fórmula, el número de características también aumenta, lo que a veces es difícil de manejar. Además, la Regresión Polinomial tiene una tendencia a ajustarse drásticamente, incluso es un simple conjunto de datos unidimensional. Si bien podamos tener una tentación de ajustar un polinomio de mayor grado para obtener un error menos, esto puede resultar en un ajuste excesivo, por esa razón siempre se debe trazar las relaciones para ver el ajuste y concentrarnos de que la curva se ajuste a la naturaleza del problema.

Regresión polinomial con 5 términos.

from sklearn.preprocessing import PolynomialFeatures

#regresion polinomial  
poly\_reg=PolynomialFeatures(degree=5)#numero de terminos  
X\_poly=poly\_reg.fit\_transform(X)  
lin\_reg2=LinearRegression()  
lin\_reg2.fit(X\_poly,Y)

**Arboles de decisiones**

Técnica que permite analizar decisiones secuenciales basada en el uso de resultados y probabilidades asociadas. Pueden usarse para desarrollar una estrategia óptima cuando el tomador de decisiones se enfrenta con:

* Una serie de alternativas de decisión
* Incertidumbre o eventos futuros con riesgo

Ventajas de un árbol de decisión:

* Resume los ejemplos de partida, permitiendo la clasificación de nuevos casos siempre y cuando no existan modificaciones sustanciales en las condiciones bajos las cuales se generan.
* Facilita la interpretación de la decisión adoptada.
* Proporciona un alto grado de compromiso del conocimiento utilizando la toma de decisiones.
* Explica el comportamiento respecto a una determinada tarea de decisión.
* Reduce el número de variables independientes.
* Es una magnifica herramienta para el control de la gestión empresarial.

Los árboles de decisión se utilizan en cualquier proceso que implique toma de decisiones, ejemplo de estos procesos son:

* Búsqueda binaria
* Sistemas Expertos
* Arboles de juego

Los árboles se pueden clasificar en dos tipos que son:

* Árboles de **regresión** en los cuales la variable respuesta Y es **cuantitativa**.
* Árboles de **clasificación** en los cuales la variable respuesta Y es **cualitativa**.

Componentes y estructura:

* Alternativas de decisión en cada punto de decisión.
* Eventos que pueden ocurrir como resultado de cada alternativa de decisión. También son llamados Estados de la naturaleza.
* Probabilidad desde que ocurran los eventos posibles.
* Resultados de las posibles interacciones entre las alternativas de decisión y los eventos. También se les conoce con el nombre de Pagos.

Diagram, timeline

Description automatically generated with medium confidence

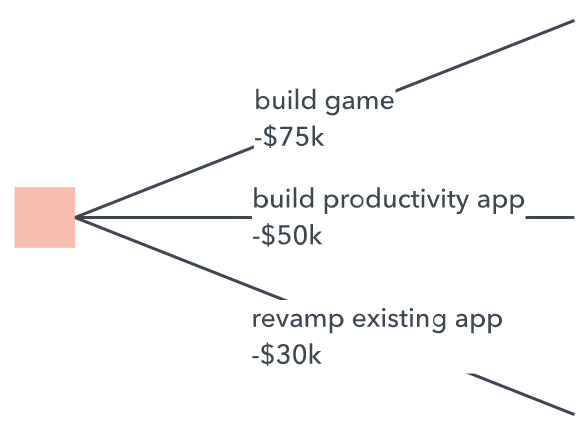
Los árboles de decisión poseen:

Diagram

Description automatically generated with medium confidence

Cómo dibujar un árbol de decisión

1. Comienza con la decisión principal. Dibuja un pequeño recuadro para representar este punto, luego dibuja una línea desde el recuadro hacia la derecha para cada posible solución o acción. Etiquétalas correctamente.



2. Agrega nodos de decisión y probabilidad para expandir el árbol del siguiente modo:

* Si otra decisión es necesaria, dibuja otro recuadro.
* Si el resultado es incierto, dibuja un círculo (los círculos representan nodos de probabilidad).
* Si el problema está resuelto, déjalo en blanco (por ahora).

Diagram

Description automatically generated

Desde cada nodo de decisión, dibuja soluciones posibles. Desde cada nodo de probabilidad, dibuja líneas que representen los resultados posibles. Si deseas analizar tus opciones de forma numérica, incluye la probabilidad de cada resultado y el costo de cada acción.

3. Continúa con la expansión hasta que cada línea alcance un extremo, lo que significa que no hay más decisiones que tomar o resultados probables que considerar. Luego, asigna un valor a cada resultado posible. Puede ser una puntuación abstracta o un valor financiero. Agrega triángulos para indicar los extremos.

Diagram

Description automatically generated

Un **árbol de regresión** consiste en hacer preguntas de tipo ¿? para cada una de las características, de esta forma el espacio de las covariables es divido en hiper-rectángulos y todas las observaciones que queden dentro de un hiper-rectángulo tendrán el mismo valor estimado . En la siguiente figura se ilustra el árbol en el lado izquierdo y la partición del espacio en el lado derecho. La partición del espacio se hace de manera repetitiva para encontrar las variables y los valores de corte de tal manera que se minimice la función de costos

Chart, scatter chart

Description automatically generated

Los pasos para realizar la partición del espacio son:

1. Dado un conjunto de covariables (características), encontrar la covariable que permita predecir mejor la variable respuesta.
2. Encontrar el punto de corte  sobre esa covariable que permita predecir mejor la variable respuesta.
3. Repetir los pasos anteriores hasta que se alcance el criterio de parada.

Árbol de decisión para regresiones en Python.

from sklearn.tree import DecisionTreeRegressor

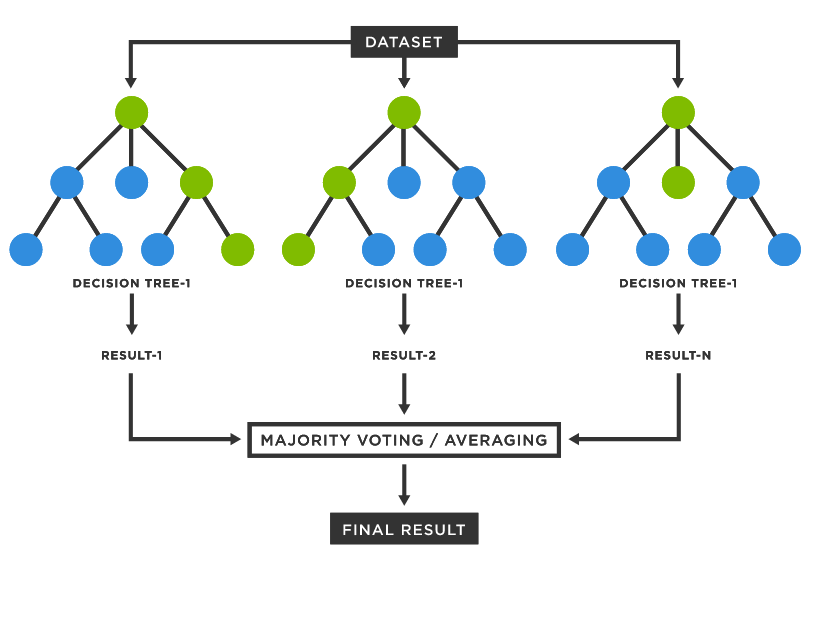
#arbol de desición para regresion  
regresion=DecisionTreeRegressor(random\_state=0)  
regresion.fit(X,Y)

Chart, line chart

Description automatically generated

**Bosques aleatorios de decisión**

Un bosque aleatorio es un algoritmo de Machine Learning supervisado. Es uno de los algoritmos más utilizados debido a su precisión, simplicidad y flexibilidad. El hecho de que pueda usarse para tareas de clasificación y regresión, combinado con su naturaleza no lineal, lo hace altamente adaptable a una variedad de datos y situaciones.



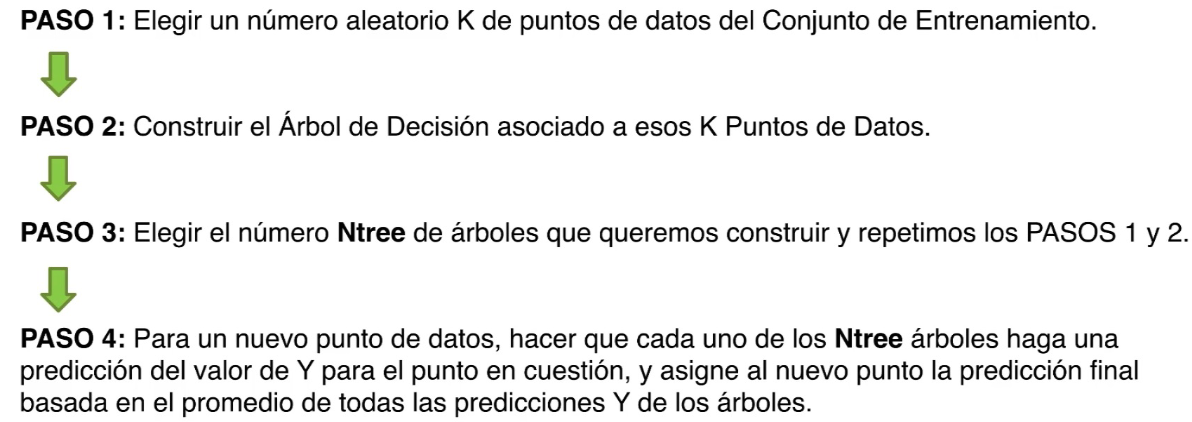
Se le llama “bosque” porque crece un bosque de árboles de decisión. Los datos de estos árboles luego se fusionan para garantizar las predicciones más precisas. Mientras que un árbol de decisiones en solitario tiene un resultado y un rango reducido de grupos, el bosque asegura un resultado más preciso con una mayor cantidad de grupos y decisiones. Tiene el beneficio adicional de agregar aleatoriedad al modelo al encontrar la mejor característica entre un subconjunto aleatorio de características. La teoría es que una gran cantidad de árboles no correlacionados crearán predicciones más precisas que un árbol de decisión individual. Esto se debe a que el volumen de árboles trabaja en conjunto para protegerse entre sí de errores individuales y sobreajuste.

Para que un bosque aleatorio funcione bien, necesita tres cosas:

* Una señal identificable para que los modelos no se limiten a adivinar.
* Las predicciones hechas por los árboles deben tener bajos niveles de correlación con los otros árboles.
* Características que tienen cierto nivel de poder predictivo: GI=GO.

|  |  |
| --- | --- |
| Arboles de decisión | Bosques aleatorios |
| Fórmula un conjunto de reglas a las características de entrenamiento. | Selecciona al azar las observaciones características para construir varios árboles de decisión y lo los promedia. |
| Si los árboles de decisión son muy profundos pueden sufrir de sobre ajuste. | Evita el exceso de adaptación la mayor parte del tiempo, creando subconjuntos aleatorios. |

Construcción de Bosques Aleatorios Regresión



Bosques aleatorios en Python

from sklearn.ensemble import RandomForestRegressor

#arbol de desición para regresion  
regresion=RandomForestRegressor(n\_estimators=10 ,random\_state=0)  
regresion.fit(X,Y)

**Clasificaciones**

Una clasificación es un ordenamiento o una organización de cosas en una serie de categorías o clases. Se pueden clasificar ideas, objetos o cualquier tipo de referente. El concepto de clasificación tiene dos significados: supervisada y no supervisada.

Clasificación supervisada

Este tipo de clasificación cuenta con un conocimiento a priori, es decir para la tarea de clasificar un objeto dentro de una categoría o clase contamos con modelos ya clasificados (objetos agrupados que tienen características comunes). La primera fase tenemos un conjunto de entrenamiento o de aprendizaje (para el diseño del clasificador) y otro llamado de prueba o de validación (para clasificación), estos nos servirán para construir un modelo o regla general para la clasificación. En la segunda fase es el proceso en sí de clasificar los objetos o muestras de las que se desconoce la clase a las que pertenecen.

Clasificación no supervisada

A diferencia de la supervisada no contamos con conocimiento a priori, por lo que tendremos un área de entrenamiento disponible para la tarea de clasificación. A la clasificación no supervisada se la suele llamar también clustering. En este tipo de clasificación contamos con “objetos” o muestras que tiene un conjunto de características, de las que no sabemos a que clase o categoría pertenece, entonces la finalidad es el descubrimiento de grupos de “objetos” cuyas características afines nos permitan separar las diferentes clases

Tipos de clasificadores

Métodos Estadísticos Clásicos

* Clasificador bayesiano simple (naiveBayes)
* Discriminadores lineales

Modelos de dependencias

* Redes Bayesianas

Aprendizaje simbólico

* Regla (3𝜎), arboles de decisión, lógica difusa

Redes Neuronales, SVM

**Clasificador bayesiano simple**

El teorema de Bayes es utilizado para calcular la probabilidad de un suceso, teniendo información de antemano sobre ese suceso. Podemos calcular la probabilidad de un suceso A, sabiendo además que ese A cumple cierta característica que condiciona su probabilidad. El teorema de Bayes entiende la probabilidad de forma inversa al teorema de la probabilidad total. El teorema de la probabilidad total hace inferencia sobre un suceso B, a partir de los resultados de los sucesos A. Por su parte, Bayes calcula la probabilidad de A condicionado a B. Para calcular la probabilidad tal como la definió Bayes en este tipo de sucesos, necesitamos una fórmula. La fórmula se define matemáticamente como:

Text

Description automatically generated

Donde B es el suceso sobre el que tenemos información previa y A(n) son los distintos sucesos condicionados. En la parte del numerador tenemos la probabilidad condicionada, y en la parte de abajo la probabilidad total. En cualquier caso, aunque la fórmula parezca un poco abstracta, es muy sencilla.

Ejemplo del teorema de Bayes:

*Una empresa tiene una fábrica en Estados Unidos que dispone de tres máquinas A, B y C, que producen envases para botellas de agua. Se sabe que la máquina A produce un 40% de la cantidad total, la máquina B un 30%, y la máquina C un 30%. También se sabe que cada máquina produce envases defectuosos. De tal manera que la máquina A produce un 2% de envases defectuosos sobre el total de su producción, la máquina B un 3%, y la máquina C un 5%. Dicho esto, se plantean dos cuestiones:*

P(A) = 0,40 P(D/A) = 0,02

P(B) = 0,30 P(D/B) = 0,03

P(C) = 0,30 P(D/C) = 0,05

*Si un envase ha sido fabricado por la fábrica de esta empresa en Estados Unidos ¿Cuál es la probabilidad de que sea defectuoso?*

*Se calcula la probabilidad total. Ya que, a partir los diferentes sucesos, calculamos la probabilidad de que sea defectuoso.*

P(D) =[P(A) x P(D/A)] + [P(B) x P(D/B)] + [P(C) x P(D/C)] = [0,4 x 0,02] + [0,3 x 0,03] + [0,3 x 0,05] = 0,032

*Siguiendo con la pregunta anterior, si se adquiere un envase y este es defectuoso ¿Cuáles es la probabilidad de que haya sido fabricado por la máquina A? ¿Y por la máquina B? ¿Y por la máquina C? Aquí se utiliza el teorema de Bayes. Tenemos información previa, es decir, sabemos que el envase es defectuoso. Claro que, sabiendo que es defectuoso, queremos saber cuál es la probabilidad de que se haya producido por una de las máquinas.*

P(A/D) = [P(A) x P(D/A)] / P(D) = [0,40 x 0,02] / 0,032 = 0,25

P(B/D) = [P(B) x P(D/B)] / P(D) = [0,30 x 0,03] / 0,032 = 0,28

P(C/D) = [P(C) x P(D/C)] / P(D) = [0,30 x 0,05] / 0,032 = 0,47

*Sabiendo que un envase es defectuoso, la probabilidad de que haya sido producido por la máquina A es del 25%, de que haya sido producido por la máquina B es del 28% y de que haya sido producido por la máquina C es del 47%.*

Los **clasificadores NaiveBayes** (NBC por sus siglas en inglés) son algoritmos de aprendizaje automático simples pero potentes. Se basan en la probabilidad condicional y el teorema de Bayes.

Algoritmo NaiveBayes

* Convertir el conjunto de datos en una tabla de frecuencias
* Crear una tabla de probabilidad calculando las correspondientes a que ocurran los diversos eventos o haciendo una distribución Gaussiana
* La ecuación NaiveBayesse usa para calcular la probabilidad posterior de cada clase
* La clase con la probabilidad posterior más alta es el resultado de la predicción.

Limitaciones

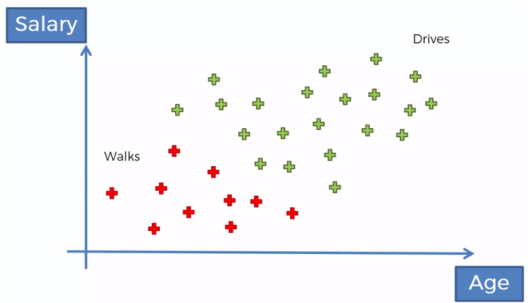
* En muchas ocasiones la suposición de independencia condicional no es válida
* Para variables continuas, existe el problema de discretización
* Alternativas–dependencias:
  + Estructuras que consideran dependencias
  + Mejora estructural del clasificador
* Alternativas–variables continuas:
  + Técnicas de discretización

Mejoramiento

* Eliminación de atributos irrelevantes (selección de atributos)
  + Medir la dependencia entre la clase y atributos (Por ejemplo con la información mutua) y elimina aquellas con poca aportación.
* Verificación de las relaciones de independencia entre atributos y alterando su estructura. (‘Redes Bayesianas’)
  + Eliminación: quitar uno de los dos (redundantes)
  + Unión: juntar los 2 atributos en uno, combinando sus valores
  + Inserción: insertar un atributo “virtual” entre la clase y los dos atributos que los haga independientes.

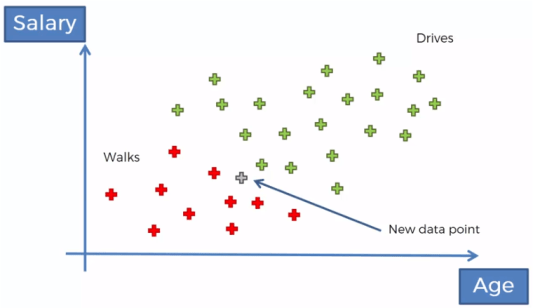
Ejemplo:

Tenemos un conjunto de datos de personas que camina o conducen hacia su trabajo, en relación con su edad y a su salario, por ejemplo.



Personas que camina o conducen hacia su trabajo en relación con la edad y el salario.

Si ahora tenemos la edad y el salario de una nueva persona, queremos clasificarla, de acuerdo con esos datos, si es de las personas que caminan o de las que conducen.



¿Una nueva persona de la que tenemos su edad y su salario, es de las que conduce o camina?

Text

Description automatically generated

Teorema de bayes para clasificar a una nueva persona en base a su edad y su salario.

Chart, scatter chart

Description automatically generated

P(Camina) es el Número de personas que caminan entre el total de observaciones.

Chart, scatter chart

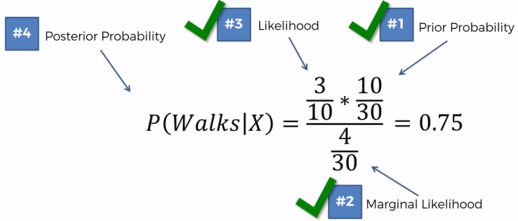
Description automatically generated

P(X) es el Número de observaciones similares al nuevo punto, entre el total de observaciones.

Chart

Description automatically generated

P(X|Camina) es el número de observaciones similares entre los que caminan entre el total de los que caminan.



Aplicando los valores a la formula del teorema.

A screenshot of a computer

Description automatically generated with low confidence

También para los que conducen

Si ahora comparamos los que caminan contra los que conducen tenemos que:

P(Camina|X) > P(Conduce |X)

0.75 > 0.25

Entonces, este nuevo punto que representa la edad y el salario de una persona nueva será clasificado en el grupo de los que caminan.

Chart, scatter chart

Description automatically generated

Clasificador bayesiano simple en Python

from sklearn.naive\_bayes import GaussianNB

#entrenar naive bayes  
classifier=GaussianNB()  
classifier.fit(X\_train, Y\_train)

**Clasificación vecino más cercano (NN y K-NN)**

Es un clasificador multiclase lineal. No define de forma explícita una frontera de separación entre clases. La frontera se define implícitamente a partir de la distancia a las muestras del conjunto de entrenamiento. No requiere una representación de cada imagen en forma de vector, únicamente la definición de una función de distancia entre dos imágenes. Se considera un método de lay learning debido a que la generalización más allá de los datos de entrenamiento es demorada hasta que se hace una pregunta al sistema. La ventaja principal que se obtiene utilizando un método de lazy learning, es que la función objetivo será aproximada localmente, como en el algoritmo vecino más k-próximo. Debido a que la función objetivo es aproximada localmente para cada pregunta al sistema, los sistemas de lazy learning pueden simultáneamente resolver múltiples problemas y gestionar con éxito cambios en el dominio del problema a cambio de un alto costo computacional.

Algoritmo NN

Text

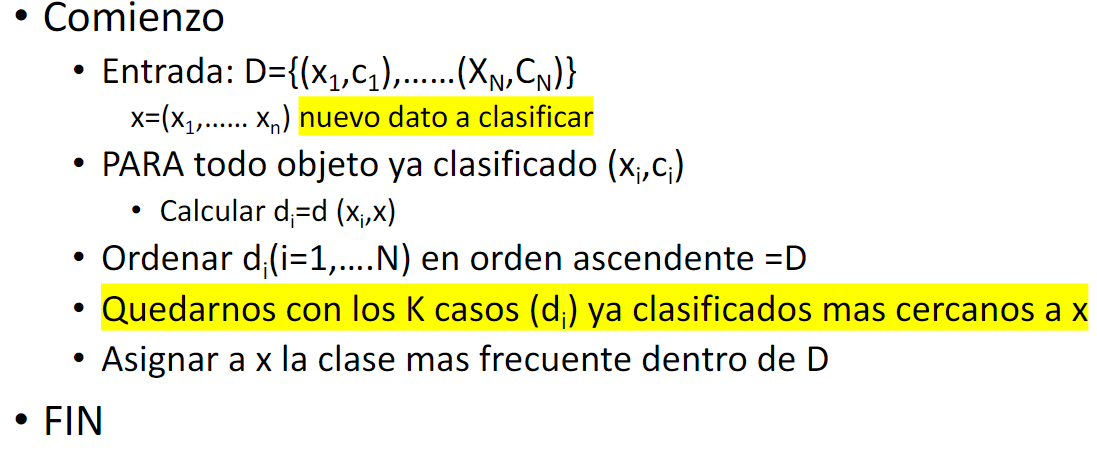
Description automatically generated

Graphical user interface, application

Description automatically generated

K vecinos más cercanos es uno de los algoritmos de clasificación más básicos y esenciales en Machine Learning. Pertenece al dominio del aprendizaje supervisado y encuentra una aplicación intensa en el reconocimiento de patrones, la minería de datos y la detección de intrusos. El algoritmo KNN es uno de los algoritmos de clasificación más simples, incluso con tal simplicidad puede dar resultados altamente competitivos. Este algoritmo consiste en seleccionar un valor de K. Al momento del análisis los K datos más cercanos al valor que se desea predecir será la solución. Acá lo importante es seleccionar un valor de K acorde a los datos para tener una mayor precisión en la predicción.

Algoritmo KNN



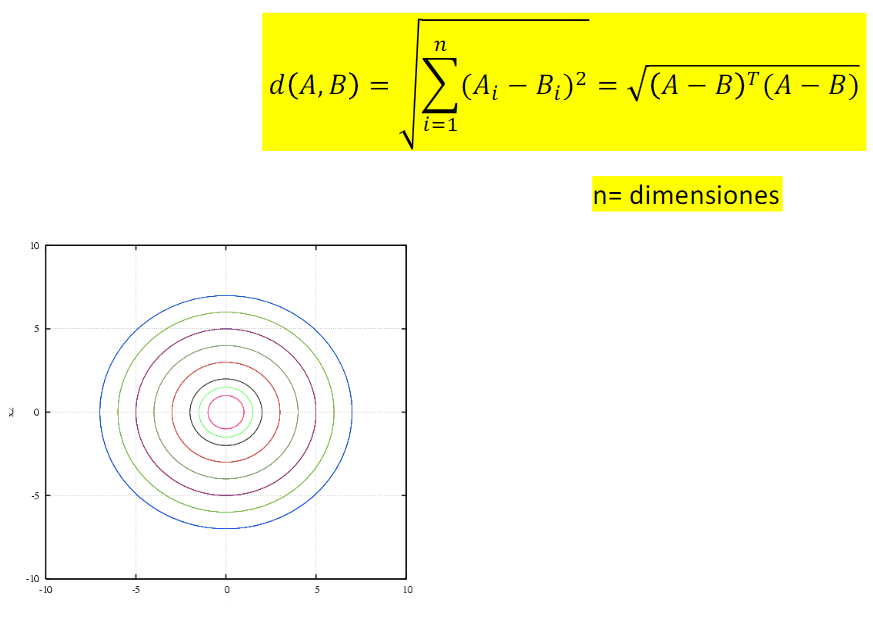
* Si k es muy pequeño el modelo será muy sensitivo a puntos que son atípicos o con ruido.
* Si K es muy grande el modelo tiende a asignar siempre la clase más grande.

¿Qué valor seleccionamos para k, el número de vecinos más cercanos a considerar?

Entrenar el clasificador con diferentes (k) y seleccionar el valor de k que nos del clasificador con mayor rendimiento.

Calculo de distancias

Distancia Euclidiana



Distancia Euclidiana ponderada

Text

Description automatically generated

Chart, radar chart

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Distancia Manhattan

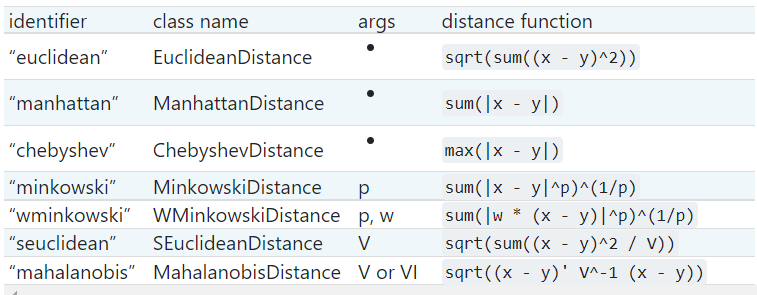
Chart, line chart

Description automatically generated

Clasificador KNN en Python

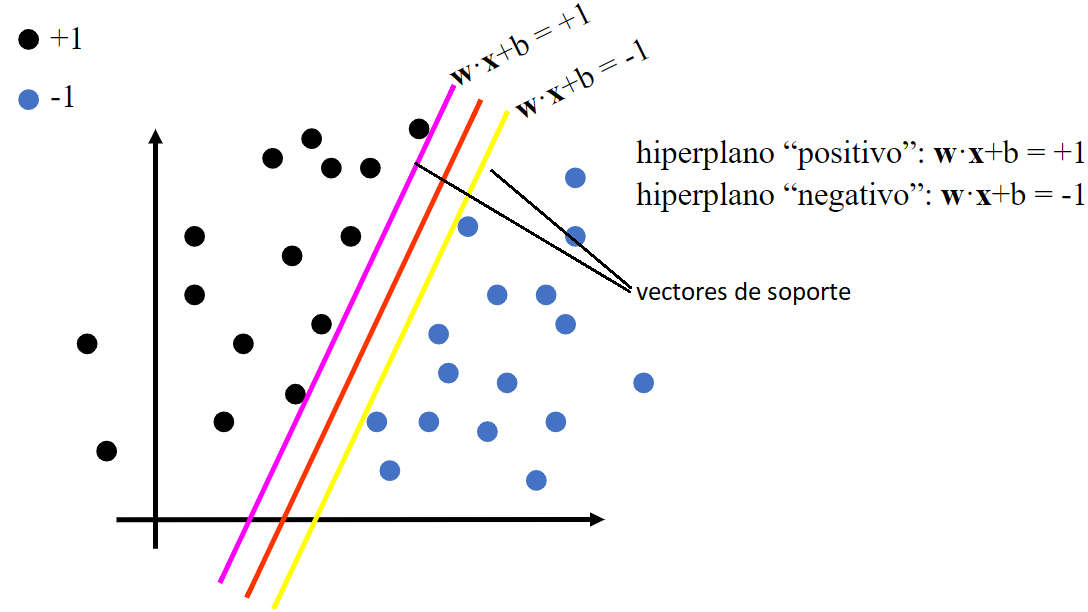
from sklearn.neighbors import KNeighborsClassifier

#entrenar KNN  
classifier=KNeighborsClassifier(n\_neighbors=5, metric= ‘euclidean’)  
classifier.fit(X\_train, Y\_train)



**Máquina con vectores de soporte**

Las máquinas de vectores de soporte son una técnica de machine learning que encuentra la mejor separación posible entre clases. Con dos dimensiones es fácil entender lo que está haciendo. Normalmente, los problemas de aprendizaje automático tienen muchísimas dimensiones. Así que, en vez de encontrar la línea óptima, el SVM encuentra el hiperplano que maximiza el margen de separación entre clases. Los vectores de soporte son los puntos que definen el margen máximo de separación del hiperplano que separa las clases. Se llaman vectores, en lugar de puntos, porque estos «puntos» tienen tantos elementos como dimensiones tenga nuestro espacio de entrada. Es decir, estos puntos multidimensionales se representan con vector de n dimensiones.



Sea d+(d-) la distancia más corta entre el hiperplano positivo (negativo) y el punto positivo (negativo) más cercano. Sea el “margen” la distancia entre los hiperplanos “positivo” y “negativo”. El margen es iguala:

La idea es encontrar un hiperplano con el máximo “margen”. Esto es un problema de optimización:

A picture containing chart

Description automatically generated

Clasificar entre dos clases únicamente con el signo de tal forma nos queda como:

Text

Description automatically generated with medium confidence

Es bastante frecuente que los datos tengan ruido, que no estén etiquetados perfectamente, o que el problema sea tan difícil que, para unos pocos puntos, sea muy complicado clasificarlos correctamente. Para estos casos, podemos decirle al SVM, que preferimos que generalice bien para la mayoría de los casos, aunque algunos pocos casos del conjunto de entrenamiento no estén perfectamente clasificados. Lo que normalmente vamos buscando es la construcción de modelos de aprendizaje automático que generalicen bien.