**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

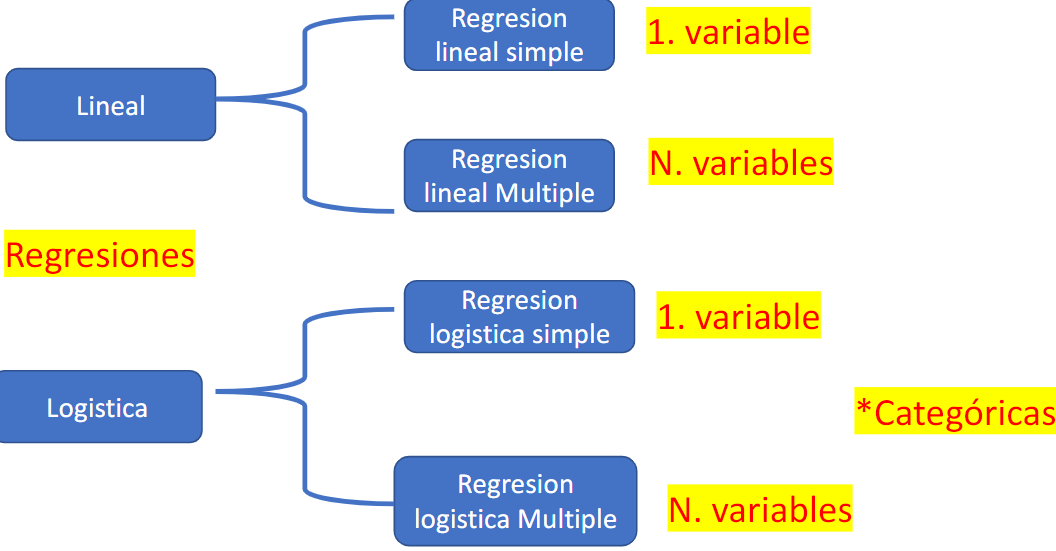
#escalar datos  
sc\_X=StandardScaler()  
X=sc\_X.fit\_transform(X\_train)

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)

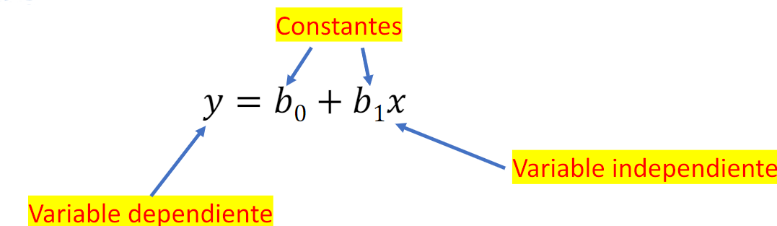
**Regresiones**

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**

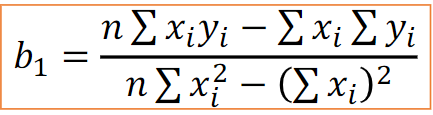
Pretende relacionar una variable dependiente con una o más variable independientes. Supóngase que se tiene un conjunto de n pares de observaciones (xi,yi), se busca encontrar una recta que describa de la mejor manera cada uno de esos pares observados.



Chart, scatter chart

Description automatically generated

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



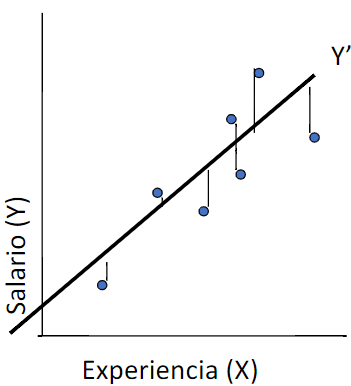
A picture containing text, orange, clock

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

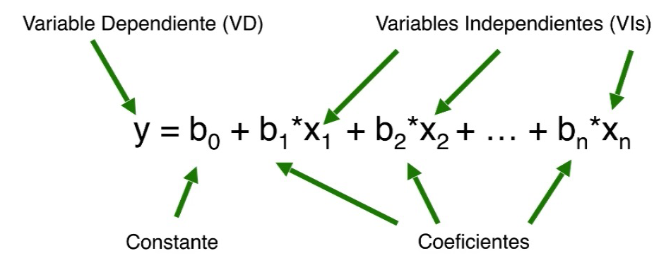
#visualizacion de datos de entrenamiento  
plt.scatter(X\_train, Y\_train, color='red')  
plt.plot(X\_train, regresion.predict(X\_train), color='blue')  
plt.title('Salario vs Experiencia (set entrenamiento)')  
plt.xlabel('Años de experiencia')  
plt.ylabel('Salario')  
plt.grid()  
plt.show()

Chart, scatter chart

Description automatically generated

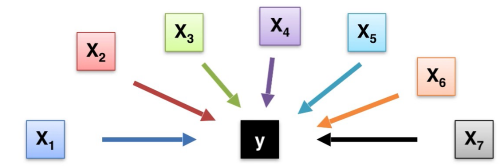
**Regresión lineal múltiple**

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.

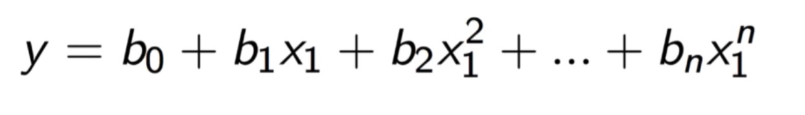
Eliminación hacia atrás antes de aplicar una regresión lineal múltiple para comprobar la importancia de las variables en el modelo.

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.