Módulo 2 - Aprendizaje supervisado: regresión lineal y logística

Introducción

En este módulo, comenzaremos a aplicar los primeros modelos de *machine learning* para aprendizaje supervisado y sus diversos usos en la vida cotidiana que relacionan, incluso, nuestro día a día con el aprendizaje de máquinas.

Junto con ello, lograremos entender y diferenciar la manera en la que se realiza el aprendizaje de un algoritmo y aplicar métricas que se relacionan con los modelos de regresión lineal y regresión logística para, posteriormente, aplicar procesamientos a los sets de datos, con el fin de que estos logren estar bien preparados y ajustados para su interpretación y uso.

Video de inmersión

Unidad 1. Regresión lineal simple y múltiple

Tema 1. Librería *Scikit Learn*

Scikit learn o sklearn es una de las librerías más utilizadas dentro del mundo de *machine learning*, esta es una biblioteca de aprendizaje automático que es esencial para la programación en Python.

La particularidad de esta librería es que cuenta con distintos algoritmos de clasificación, regresión y agrupamiento, dentro de los cuales se incluye la aplicación de vectores *random forest, gradient boosting, k-means* y DBSCAN. Los cuales son utilizados para realizar aprendizajes supervisados y no supervisados.

Esta librería está diseñada para interactuar con las bibliotecas numéricas y científicas clásicas de Python, como son Pandas, Numpy y SciPy.

Scikit learn se encuentra inspirado dentro del ambiente de Python y utiliza NumPy ampliamente para operaciones de matriz y álgebra lineal de alto rendimiento.

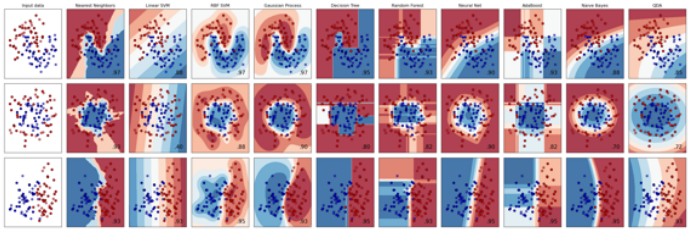
Esta librería es muy importante, ya que nos permitirá realizar todos nuestros análisis para entrenar y testear bases en ambientes supervisados y no supervisados.

Algunas aplicaciones relacionadas con Scikit learn permiten realizar análisis como los que se detallan a continuación:

1. Clasificación

- Identificación de categorías a las cuales pertenece un objeto y variable.
- **Aplicaciones:** detección de *spam*, reconocimiento de imágenes.
- **Algoritmos:** *super vector machine* - SVM, KNN - vecinos más cercanos, *random forest* - bosque aleatorio y más...

Figura 1. Clasificadores

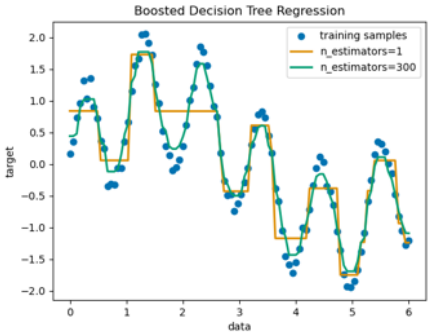


Fuente: [Imagen sin título sobre clasificadores]. (s. f.). https://bit.ly/3SYVvFe

Regresión

- Predecir un atributo de valor continuo, asociado con un objeto.
- **Aplicaciones:** respuesta a medicamentos, precios de acciones.
- **Algoritmos:** SVR, KNN - vecinos más cercanos, *random forest* - bosque aleatorio y más...

Figura 2. Regresión

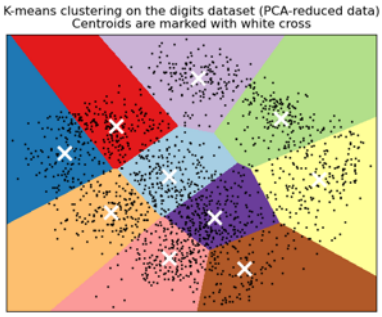


Fuente: [Imagen sin título sobre regresión]. (s. f). https://bit.ly/3fPkkoj

3. Agrupación

- Agrupación automática de objetos similares en conjuntos.
- **Aplicaciones:** segmentación de clientes, agrupación de resultados de experimentos.
- **Algoritmos:** *K-means*, agrupamiento espectral, desplazamiento medio y más...

Figura 3. Agrupación

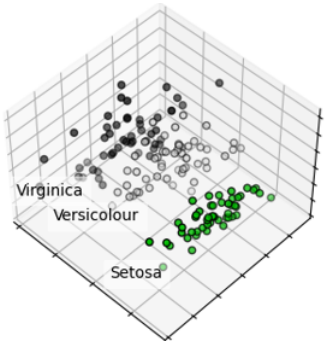


Fuente: [Imagen sin título sobre agrupación]. (s. f.). https://bit.ly/3RUuBNm

4. Reducción de dimensionalidad

- Reducir el número de variables aleatorias para considerar.
- **Aplicaciones:** visualización, aumento de la eficiencia.
- **Algoritmos:** PCA, selección de características, factorización de matrices no negativas y más...

Figura 4. Reducción de dimensionalidad

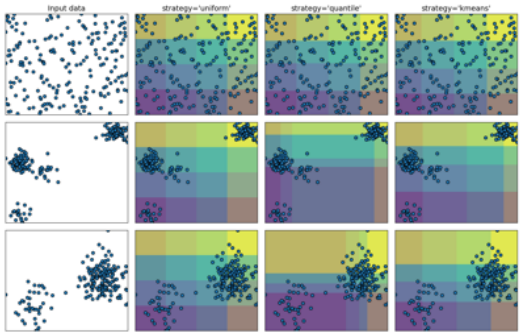


Fuente: [Imagen sin título sobre reducción de dimensionalidad]. (s. f.). https://bit.ly/3MqSP0P

5. Preprocesamiento

- Extracción y normalización de características.
- **Aplicaciones:** transformación de datos de entrada, como texto, para su uso con algoritmos de aprendizaje automático.
- **Algoritmos:** preprocesamiento, extracción de características y más...

Figura 5. Preprocesamiento



Fuente: [Imagen sin título sobre preprocesamiento]. (s. f.). https://bit.ly/3eph8iV

Si necesitas saber más acerca de Scikit learn, puedes consultar su página oficial. En ella, podrás encontrar todos los avances y versiones respecto a esta librería.

https://scikit-learn.org/stable/

Actividad de repaso

La empresa Máquina Artificial, dedicada a la importación de computadores para el cono sur de la región, está pensando en utilizar machine learning para predecir datos de los posibles clientes. Para ello, debe utilizar librerías que le permitan entrenar y

testear para realizar sus modelos. Tú, como analista de data science, recomendarías utilizar:

Market Rain.

Sklearn.

Random forest book.

Anaconda.

Pandas.

Justificación

Tema 2. Regresión lineal simple

Regresión lineal es un método estadístico que tiene como objetivo modelar la relación existente entre una variable continua (comúnmente descrita en el eje Y de un plano cartesiano) y una o más variables independientes (descrita como el eje X dentro de un plano cartesiano) mediante el ajuste de una ecuación lineal.

En nuestro caso, denominaremos regresión lineal simple cuando, dentro de nuestros datos, solo exista una variable independiente (eje X); y regresión lineal múltiple, cuando hay más de una variable independiente que sea parte de un set de datos.

Dependiendo de cada caso, a la variable continua se le denominará variable dependiente o variable respuesta, y a las variables independientes como: regresores, predictores o *features*.

Se debe tener en consideración que, además de ser un método estadístico, la regresión lineal también cumple con ser clasificada como un algoritmo de aprendizaje supervisado que se utiliza en *machine learning*.

Matemáticamente, en su versión más sencilla, lo que haremos es dibujar una recta que nos indicará la tendencia de un conjunto de datos continuos.

En estadísticas, regresión lineal es una aproximación para modelar la relación entre una variable escalar dependiente “y” y una o más variables explicativas nombradas con “X”.

[Para adentrarnos un poco más en su uso] recordemos rápidamente la fórmula de la recta:

$$Y = mX + b$$

Donde

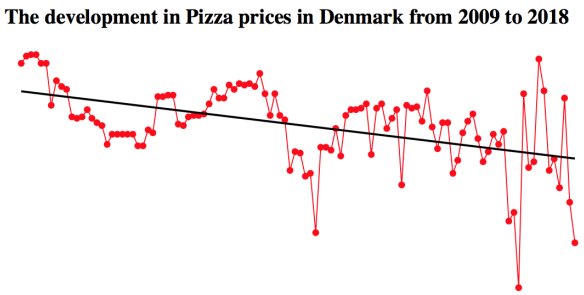
Y: es el resultado.

X: variable independiente.

m: la pendiente o inclinación de la recta.

b: punto de corte con el eje Y en la gráfica (cuando X = 0). (Gianareas, 2019, https://platzi.com/blog/regresion-lineal-en-espanol-con-python/).

Figura 6. Regresión lineal simple



Fuente: [Imagen sin título sobre regresión lineal simple]. (s. f.). https://bit.ly/3RTAnPw

Aquí hay un ejemplo donde vemos datos recabados sobre los precios de las pizzas en Dinamarca (los puntos en rojo) y la línea negra es la tendencia, donde la finalidad, en nuestro caso, será predecir los precios de las pizzas que más se acerquen a la recta para ver el nivel de acierto de nuestras predicciones ante cambios futuros.

Actividad de repaso

Según lo aprendido sobre regresión lineal, determina cuál de las siguientes opciones correspondería a un modelo que permita

predecir información de este tipo:

No es necesario tener variables independientes para su aprendizaje.

Es un modelo no supervisado.

Es un modelo supervisado.

Es un modelo híbrido de estudio.

Justificación

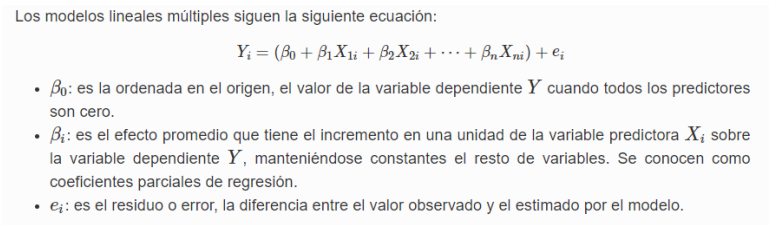
Tema 3. Regresión lineal múltiple

La regresión lineal múltiple cumple con ser un modelo estadístico que permite generar un modelo lineal en el que el valor de la variable dependiente o respuesta (Y) se determina a partir de un conjunto de variables independientes llamadas predictores (X1, X2, X3…). (Amat, 2016, https://www.cienciadedatos.net/documentos/25_regresion_lineal_multiple).

Es decir, a diferencia de la regresión lineal simple, esta puede utilizar más de una variable independiente para realizar sus predicciones.

Este modelo es una extensión de la regresión lineal simple, por lo que es fundamental comprender esta última. Los modelos de regresión múltiple pueden emplearse para predecir el valor de la variable dependiente o para evaluar la influencia que tienen los predictores sobre ella. (Amat, 2016, https://www.cienciadedatos.net/documentos/25_regresion_lineal_multiple).

Figura 7. Modelos



Fuente: Amat, 2016, https://bit.ly/3CkyqFK

Condiciones para una regresión lineal múltiple

- **No colinealidad o multicolinealidad**

En los modelos lineales múltiples, los predictores deben ser independientes, no debe de haber colinealidad entre ellos.

La colinealidad ocurre cuando un predictor está linealmente relacionado con uno o varios de los otros predictores del modelo o cuando es la combinación lineal de otros predictores. Como consecuencia de la colinealidad, no se puede identificar de forma precisa el efecto individual que tiene cada una de las variables colineales sobre la variable respuesta, lo que se traduce en un incremento de la varianza de los coeficientes de regresión estimados hasta el punto de que resulta prácticamente imposible establecer su significancia estadística. Además, pequeños cambios en los datos provocan grandes cambios en las estimaciones de los coeficientes. Si bien la colinealidad propiamente dicha existe solo si el coeficiente de correlación simple o múltiple entre algunas de las variables independientes es 1, esto raramente ocurre en la realidad. Sin embargo, es frecuente encontrar la llamada casi colinealidad o multicolinealidad no perfecta.

- **Parsimonia**

Este término hace referencia a que el mejor modelo es aquel capaz de explicar con mayor precisión la variabilidad observada en la variable respuesta empleando el menor número de predictores, por lo tanto, con menos asunciones.

Relación lineal entre los predictores numéricos y la variable respuesta

Cada predictor numérico tiene que estar linealmente relacionado con la variable respuesta Y, mientras los demás predictores se mantienen constantes, de lo contrario, no se puede introducir en el modelo. La forma más recomendable de comprobarlo es representando los residuos del modelo frente a cada uno de los predictores. Si la relación es lineal, los residuos se distribuyen de forma aleatoria en torno a cero. Estos análisis son solo aproximados, ya que no hay forma de saber si realmente la relación es lineal cuando el resto de predictores se mantienen constantes.

- **Tamaño de la muestra**

No se trata de una condición de por sí, pero si no se dispone de suficientes observaciones, predictores que no son realmente influyentes podrían parecerlo. Por lo general, se recomienda que el número de observaciones sea, como mínimo, entre 10 y 20 veces el número de predictores del modelo.

La gran mayoría de las condiciones se verifican utilizando los residuos, por lo tanto, se suele generar primero el modelo y, posteriormente, validar las condiciones. De hecho, el ajuste de un modelo debe verse como un proceso iterativo en el que se ajusta el modelo, se evalúan sus residuos y se mejora. Así, hasta llegar a un modelo óptimo. (Amat, 2016, https://www.cienciadedatos.net/documentos/25_regresion_lineal_multiple).

Actividad de repaso

Dentro de la predicción de datos, el análisis de los datos cumple un rol fundamental para poder generar buenos resultados. En el caso de la regresión lineal múltiple, una de sus principales características es:

- Puede utilizar más de una variable independiente para llevar a cabo el modelo.
- Solo utiliza una variable independiente para poder llevar a cabo el modelo.
- Puede realizarse sin la necesidad de tener una variable independiente.
- Puede realizarse solo para validar los modelos, sin necesidad de entrenar.
- Se puede realizar para hacer aprendizaje no supervisado.

Justificación

Tema 4. División de datos

En los temas anteriores, hemos logrado revisar los fundamentos básicos para entender un modelo de regresión simple y múltiple.

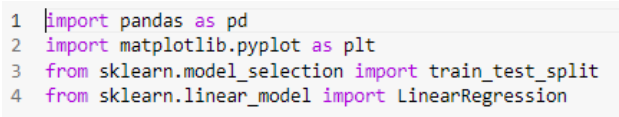
Con ello, ahora pondremos a prueba los conocimientos básicos para aplicar un modelo de regresión lineal.

Para realizar esto, utilizaremos un set de datos relacionados con el salario y años de experiencia que tiene una persona.

Para ello, comenzaremos trabajando con los siguientes pasos de cálculo.

Consideremos que, para estos pasos, utilizaremos un set de datos que están preprocesados, por lo cual realizaremos pasos de limpieza e imputación de datos.

Figura 8: Paso 1 - Importar librerías que se van a utilizar



Fuente: elaboración propia.

En el caso de Sklearn:

En la línea 3, se está importando un paquete que permitirá hacer la división de los datos en entrenamiento y test.

IMPORTANTE: recuerda que todo modelo, primero, se debe entrenar y, luego, se debe poner a prueba mediante test, por lo que es fundamental contar con esta línea de código en cualquier tipo de modelo que uno quiera realizar.

En el caso de la línea 4, se está importando información para la realización de una regresión lineal.

Paso 2 - Cargar los datos

Lo siguiente que vamos a hacer es cargar nuestra información, cargar los datos que ya tenemos disponibles y visualizar las 5 primeras filas, mediante la función *head*.

Figura 9. Código

Fuente: elaboración propia.

Lo importante aquí es entender el set de datos y su composición.

En él, vamos a encontrar la siguiente información:

- Experiencia: que son los años de experiencia de los programadores.
- Salario: cantidad de dinero ganada según la experiencia.

Paso 3 - Revisar cuánta información tiene almacenada el set de datos

Figura 10. Código

```
1  dataset.shape
(30, 2)
```

Fuente: elaboración propia.

En nuestro caso, son 30 observaciones que relacionan dos columnas: el salario y la experiencia.

Paso 4 - Dividir los datos

Con la información que tenemos, es suficiente para dividir los datos en un set de entrenamiento y testeo.

Recordemos que la variable X son los datos que nos van a servir para poder hacer las predicciones. En este caso, corresponde a la variable *Experiencia*.

Con el siguiente código obtendremos la separación de las variables.

Figura 11. Código

```
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

Fuente: elaboración propia.

En el caso de X, el código recorre todas las filas y descarta la última columna.
En el caso de Y, considera todas las filas y solo considera la última columna.

Paso 5:

Ahora viene una parte totalmente importante, que es la de entrenar la información.

Para ello, se crearán 4 variables:

- X_train: entrenamiento para variable x.
- X_test: test para variable x.
- Y_train: entrenamiento para variable y.
- Y_test: test para variable y.

A cada uno de estos datos los vamos a dividir en entrenamiento y validación con el método train_test_split (como ya se ha importado en el paso 1).

Según muestra el siguiente código.

Figura 12. Código

```
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

Fuente: elaboración propia.

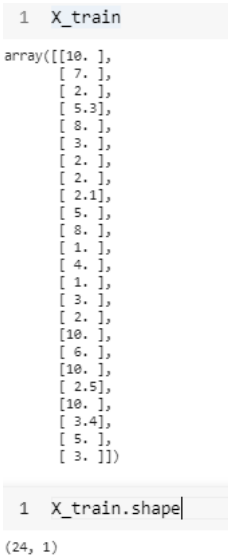
En el caso de test_size: quiere decir que el 80 % de los datos serán exclusivamente para entrenamiento y el restante va a ser para test.

En el caso de random_state = 0, siempre tendremos los mismos datos para evaluar cuando ejecutemos el modelo.

Paso 6 - Revisar información de entrenamiento y test

En este caso, vemos los datos que se están entrenando con la variable X_train.

Figura 13. Código



Fuente: elaboración propia.

Observamos que la variable de entrenamiento está tomando solo 24 datos de los 30 iniciales, lo que indica que es el 80 % de la data, según lo revisado en el paso 5.

De esta forma, la división de datos nos prepara para, posteriormente, comenzar a aplicar en la práctica los modelos de regresión logística u otra que quisiéramos aplicar.

Actividad de repaso

Dentro de la predicción de datos de la regresión lineal simple, es necesario dividir los datos en entrenamiento y testeo. En este caso, ¿cuál es una proporción adecuada para poder realizar la división de manera correcta?

- 60 % entrenamiento – 40 % testeo.
- 30 % entrenamiento – 70 % testeo.
- 50 % entrenamiento – 50 % testeo.
- 20 % entrenamiento – 80 % testeo
- 80 % entrenamiento – 20 % testeo.

Justificación

Unidad 2. Regresión logística e indicadores

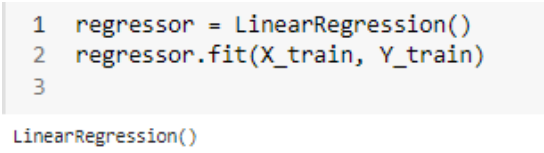
Tema 1. Creación de modelos de regresión lineal

Con base en lo aprendido en el tema anterior, realizaremos nuestro primer modelo de regresión logística, el cual nos permitirá entender cómo se entrena y visualiza un modelo de aprendizaje de *machine learning.*

En el ejercicio práctico, daremos continuidad al modelo de datos de salarios y experiencia realizado anteriormente. Para ello, definiremos la variable *regressor*, a la cual se le asignará el modelo de regresión lineal haciendo uso de la librería de Scikit learn.

Para entrenar un modelo en su totalidad, tendremos que entregar valores X e Y. Para realizarlo, llamaremos la función *fit*, de la siguiente forma:

Figura 14. Código



Fuente: elaboración propia.

Con este código ejecutado, ya hemos creado y almacenado nuestro primer modelo de regresión logística.

El método *fit* es la clave, ya que lo veremos en todos los modelos futuros:

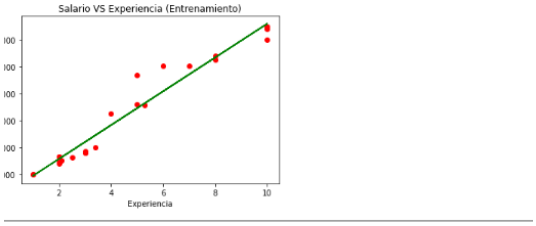
No olvidemos la estructura

- Cargar datos.
- Dividir datos en entrenamiento y prueba.
- Entrenar el modelo con *fit* para lograr el aprendizaje.

Una vez que tenemos el aprendizaje, es importante saber qué es lo que ha aprendido el modelo. Para ello, graficaremos los datos con el siguiente código:

Figura 15. Código

```
viz_train = plt
viz_train.scatter(X_train, Y_train, color = 'Red')
viz_train.plot(X_train, regressor.predict(X_train), color = 'Green')
viz_train.title('Salario VS Experiencia (Entrenamiento)')
viz_train.xlabel('Experiencia')
viz_train.ylabel('Salario')
viz_train.show()
```



Fuente: elaboración propia.

Dentro de esta gráfica, estamos visualizando la manera en la que hacemos que la máquina tenga conocimiento de nuestro algoritmo a través de la información que hemos entregado previamente como etiqueta (base de datos con los valores ya conocidos).

Podemos darnos cuenta de que hay puntos que están muy cerca de la recta, lo cual significa que la máquina aprendió mejor; y hay algunos datos que están muy alejados de la recta, que significa que no se encontró una relación adecuada con la variable X (años de experiencia).

Pero ¿cómo podemos estar seguros de que lo que nos está mostrando el modelo es correcto?

Ahora, realizamos la misma información para visualizar los datos de test.

Figura 16. Código

```
viz_test = plt
viz_test.scatter(X_test, Y_test, color = 'Red')
viz_test.plot(X_train, regressor.predict(X_train), color='Green')
viz_train.title('Salario VS Experiencia (Prueba)')
viz_train.xlabel('Experiencia')
viz_train.ylabel('Salario')
viz_test.show()
```

Fuente: elaboración propia.

Según lo visualizado en los datos de prueba, podemos darnos cuenta de que el modelo aprendido tiene pocos datos, en comparación con la gráfica vista anteriormente. Esto se debe a que estamos graficando el 20 % de la información, según la división de entrenamiento y testeo que se realizó.

En este caso, tenemos pocos datos con los cuales estamos validando nuestro conocimiento para tener un modelo mejor.

Por lo cual, se recomienda tener un set de datos que tenga mucha más información, para que la máquina pueda aprender de la mayor cantidad de datos posibles. De esta forma, al graficar nuestros datos, nuevamente, podemos encontrar algunos que estén más cercanos a la línea de la gráfica.

Para poder ver qué tan certera es nuestra predicción, aprenderemos nuestro primer indicador del modelo, el cual se denomina Score.

Figura 17. Código

```
1  regressor.score(X_test, Y_test)
```

```
0.7862437374732997
```

Fuente: elaboración propia.

Esto quiere decir que el 78 % de los nuevos datos que utilicemos para validar el aprendizaje se hará de manera correcta; el restante no se hará de manera adecuada.

Por lo cual, tendremos mejores resultados a medida que tengamos mayor cantidad de datos para poder enseñar a la máquina.

Con esta información, ya hemos creado nuestro primer modelo de regresión lineal simple.

Actividad de repaso

Dentro de la predicción de datos, podría ocurrir que la información sea infraajustada: ¿cuándo ocurre esto?

Cuando tenemos muchos datos para entrenar el modelo.

El infraajuste no ocurre dentro de modelos de regresión lineal.

Cuando tenemos muy pocos datos para entrenar el modelo.

El infraajuste ocurre solo en modelos de regresión lineal múltiple.

Es necesario contar con datos menores a 10 registros para poder realizar una buena predicción de datos.

Justificación

Tema 2. Regresión logística

En las secciones anteriores, hemos aprendido acerca de modelos de regresión lineal y regresión múltiple. Con el modelo de datos de experiencia y salario te habrás dado cuenta de que solo hemos trabajado con datos enteros y flotantes.

En esta instancia, vamos a trabajar con otro ejemplo. ¿Qué sucedería si a los atributos del *data set*, ahora, tuviésemos que asignarles valores verdaderos o falsos, si y no o 0 y 1?

¿Qué hacemos cuando tenemos variables de tipo cualitativos o etiquetas que corresponden a una clasificación? En este caso, la regresión lineal o múltiple no nos podría servir para realizar este modelo.

Ahora tendremos que realizar un nuevo modelo que se adecúe a estas características. De esta manera, aparece dentro de nuestro aprendizaje el modelo de regresión logística.

Este modelo, al igual que los anteriores, cumple con los mismos métodos vistos, pero ahora tendremos que usar la información de otra forma para poder clasificar.

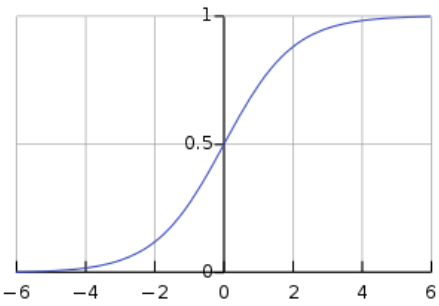
Veamos un ejemplo de salud:

Imagina que tenemos un set de datos de pacientes que tienen características de salud que están relacionadas con diabetes. Con base en los atributos medidos para cada paciente, tú puedes tener el resultado de sus exámenes como etiqueta de salida: Tiene diabetes / No tiene diabetes.

Con esta información, a partir de los pacientes que están enfermos, podemos predecir si nuevos pacientes también lo están, para poder clasificar esta información.

Para ello, utilizaremos una nueva gráfica, que es la regresión logística, que tendrá las siguientes características:

Figura 18. Código



Fuente: elaboración propia.

En la parte superior, cercano a 1, corresponde a tu binario “sí” (tiene diabetes).

En la parte inferior, cuando es cercano a 0, significa “no” (no tiene diabetes).

Si estamos en el punto medio, tenemos que dividir esa información y saber qué tan cercana está a 1 o a 0.

Por ejemplo, si el valor está en 0.60, entonces, decimos que sí tiene diabetes, aunque su predicción es lejana al valor 1, pero nuestro objetivo de salida es etiquetar a los pacientes en 1 o 0.

Para ello, utilizaremos un set de datos de diabetes. Comenzaremos trabajando con los

siguientes pasos de cálculo.

Figura 19: Paso 1 - Importar librerías que se van a utilizar

```
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Fuente: elaboración propia.

En el caso de Sklearn:

En la línea 3, está un paquete que permitirá hacer la división de los datos en entrenamiento y test.

Importante: esto es algo que nunca puedes olvidar. En todo modelo, primero, se debe entrenar y, luego, se debe poner a prueba mediante test, por eso es fundamental contar con esta línea de código en cualquier tipo de modelo que uno quiera realizar.

En el caso de la línea 4: se está importando información para la realización de una regresión logística.

Paso 2 - Cargar los datos

Lo siguiente que vamos a hacer es, justamente, cargar nuestra información, cargar nuestros datos que ya tenemos disponibles.

Figura 20. Código

```
1 diabetes = pd.read_csv('diabetes.csv')
2 diabetes.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Fuente: elaboración propia.

En él, encontraremos que la variable *outcome* será la salida 0 y 1, según la persona tenga diabetes o no tenga diabetes.

Figura 21: Paso 3 - Revisar cuánta información tiene almacenada el set de datos

```
1  diabetes.shape
```

```
(768, 9)
```

Fuente: elaboración propia.

En nuestro caso, son 768 observaciones que relacionan a 9 variables con la diabetes.

Paso 4: Dividir los datos

Con la información que tenemos, es suficiente para dividir los datos para entrenar.

Recordemos que X son los datos que nos van a servir para hacer las predicciones.

Con el siguiente código, obtendremos la separación de las variables.

Figura 22. Código

```
feature_cols = ['Pregnancies', 'Insulin', 'BMI', 'Age', 'Glucose', 'BloodPressure', 'DiabetesPedigreeFunction']
x = diabetes[feature_cols]
y = diabetes.Outcome
```

Fuente: elaboración propia.

En el caso de X, el código recorre todas las filas y no considera la variable de salida *Outcome*.

En el caso de Y, considera todas las filas y solo considera la última columna *Outcome*.

Paso 5

Ahora viene una parte totalmente importante, que es la de entrenar la información.

Para ello, se crearán 4 variables:

- X_train: entrenamiento para variable X.
- X_test: test para variable X.
- Y_train: entrenamiento para variable Y.
- Y_test: test para variable Y.

A cada uno de estos datos lo vamos a dividir en entrenamiento y validación con el método train_test_split (como ya se ha importado en el paso 1).

Según muestra el siguiente código.

Figura 23. Código

```
1  X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
```

Fuente: elaboración propia.

En el caso de test_size: quiere decir que el 75 % de los datos serán exclusivamente para entrenamiento y el restante va a ser para test.

En el caso de random_state = 0, siempre tendremos los mismos datos para evaluar cuando ejecutemos el modelo.

Paso 6 - Revisar información de entrenamiento y test

Paso 7 - Definiremos la variable logreg, a la que se le asignará el modelo de regresión lineal, según el uso de Scikit learn.

Para entrenar un modelo en su totalidad, tendremos que entregar valores para X y para Y. Para realizarlo, llamaremos la función *fit*, de la siguiente forma:

Figura 24. Código

```
1  logreg = LogisticRegression()
2  logreg.fit(X_train, y_train)
3  y_pred = logreg.predict(X_test)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Fuente: elaboración propia.

Con este código ejecutado, ya hemos creado y almacenado nuestro primer modelo de regresión logística.

El método .*fit* es la clave, ya que **lo veremos en todos los modelos futuros:**

No olvides la estructura:

- Cargar datos.
- Dividir datos en entrenamiento y prueba.
- Entrenar el modelo con .*fit* para lograr el aprendizaje.

Una vez que tenemos el aprendizaje, revisaremos qué valores nos entrega la variable de salida predicha en test.

Figura 25. Código

```
1  y_pred
```

```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0])
```

Fuente: elaboración propia.

Estos valores corresponderán a lo que ha aprendido de predicción la variable X_test.

Con esta información, ya hemos creado nuestro primer modelo de regresión logística.

Actividad de repaso

Dentro de la predicción de datos, ¿en qué casos se debe hacer uso de la regresión logística?

- Cuando tenemos que predecir variables de salida de tipo cuantitativas.
- Cuando tenemos que predecir variables de salida de tipo cualitativas.
- No existe discriminación entre variables para su uso.

Solo en casos que existe infraajuste.

Solo en casos que existe sobreajuste.

Justificación

Tema 3. Evaluación de modelos de regresión logística

Según lo revisado en los temas anteriores, ya hemos sido capaces de crear nuestros primeros modelos de regresión lineal y regresión logística. Ahora, ha llegado el momento de revisar con el *data set* de diabetes si, dadas las características de pacientes que han sido previamente clasificados en el modelo supervisado, este es capaz de crear una predicción para saber si la persona tiene diabetes o no, de acuerdo con lo aprendido.

¿Cómo podemos saber si un modelo lo está haciendo bien?

Hay que tener mucho cuidado cuando estamos trabajando con *data sets* en los cuales ya va incluida una etiqueta de salida, porque esos datos serán la guía para que el modelo pueda entregar un diagnóstico. Por lo cual, si *la data* trae etiquetas erróneas, el modelo también aprenderá erróneamente.

Para realizar esto, regularmente, se trabaja con una matriz denominada **matriz de confusión**, la cual está compuesta de 4 cuadrantes, que son divididos según las observaciones que se cuentan en la base y aquellas predicciones que son capaces de realizarse con el algoritmo de aprendizaje.

La siguiente tabla presenta la composición de la matriz de confusión.

Tabla 1. Predicción

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Fuente: Zelada, 2017, https://bit.ly/3ExDSbe

Esta matriz constará de 4 cuadrantes clave para realizar el análisis de las observaciones obtenidas, las cuales están divididas de la siguiente forma:

1. Observaciones verdaderas positivas.
2. Verdaderas negativas.
3. Falsas positivas.
4. Falsas negativas.

Otra de las formas es realizar el análisis mediante Scikit learn, utilizando el indicador de exactitud, con el cual podremos validar qué tan cercana es la precisión de nuestro modelo desde 0 a 100.

Es decir, queremos saber si el modelo está haciendo una clasificación adecuada o está muy alejado de 100. En el caso que sea así, entonces, tendríamos que volver a validar con qué datos está aprendiendo nuestro modelo.

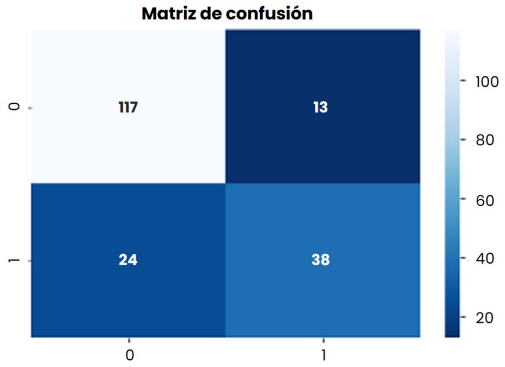
Ahora, realizaremos nuestro código para evaluar visualmente las relaciones de la matriz de confusión y cómo el modelo está clasificando nuestros datos.

Figuras 26 y 27. Código

```
import numpy as np
cnf_matrix = metrics.confusion_matrix(y_test,y_pred)

class_names = [0,1]
fig,ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

sns.heatmap(pd.DataFrame(cnf_matrix), annot = True, cmap='Blues_r', fmt='g')
plt.tight_layout()
plt.title('Matriz de confusion', y=1.1)
plt.y_label('Etiqueta Actual')
plt.x_label('Etiqueta de Prediccion')
```



Fuente: elaboración propia.

De este código, podremos interpretar los resultados de la matriz de la siguiente forma:

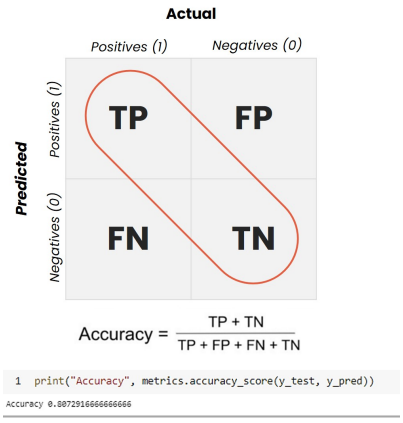
- **117:** es la cantidad de ejemplos que han sido clasificados como **verdaderos positivos** (clasificados correctamente).
- **38:** son aquellos valores clasificados correctamente cuando es negativo. Es decir, que acierta a la predicción de que una persona no tiene diabetes.
- **11:** clasificados incorrectamente como negativos. Es decir, se le asignó la etiqueta no tiene diabetes cuando, en realidad, sí la tenía.
- **24:** son aquellos que fueron clasificados incorrectamente como positivos. Es decir, se le asignó diabetes cuando, en realidad, no tenía.

Ahora, aprenderemos uno de los indicadores principales que se utilizan dentro de un modelo de *machine learning*, el cual se denomina precisión o *accuracy*.

Accuracy: con la métrica de precisión, podemos medir la **calidad** del modelo de *machine learning* en tareas de clasificación, este indicador podrá entregarnos, como resultado, el porcentaje total de elementos clasificados correctamente.

Aplicado, tiene un nivel de probabilidad alto de acertar a las predicciones de datos nuevos que se agreguen a la base de datos.

Figuras 28 y 29. Código



Fuente: elaboración propia.

Actividad de repaso

Dentro de la predicción de datos, a qué definición corresponde la afirmación: "dentro de una matriz de confusión, es el porcentaje total de elementos clasificados correctamente".

F1 Score.

Re-Call.

Testeo.

Accuracy.

Entrenamiento.

Justificación

Tema 4. Matriz de confusión

Tabla 2. Predicción

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Fuente: Zelada, 2017, https://bit.ly/3ExDSbe

Donde

- Los verdaderos positivos (VP) son aquellos que fueron clasificados correctamente como positivos por el modelo.
- Los verdaderos negativos (VN) corresponden a la cantidad de negativos que fueron clasificados correctamente como negativos por el modelo.
- Los falsos negativos (FN) son la cantidad de positivos que fueron clasificados incorrectamente como negativos.
- Los falsos positivos (FP) indican la cantidad de negativos que fueron clasificados incorrectamente como positivos. (Zelada, 2017, https://rpubs.com/chzelada/275494).

Imaginemos el siguiente ejemplo para entender de mejor manera una matriz de confusión:

Un médico tiene cuatro pacientes y a cada uno se le solicitó un examen de sangre. Por error, el laboratorio realizó también un estudio de embarazo, cuando los pacientes llegan el médico les da los resultados.

- A la primera paciente le da la noticia de que está embarazada y ella ya lo sabía, dado que tiene 3 meses de embarazo, es decir, un verdadero positivo.
- Al siguiente paciente le dice que no está embarazado y es una clasificación evidente, dado que es hombre (verdadero negativo).
- La tercera paciente llega y los resultados le indican que no está embarazada, sin embargo, tiene cuatro meses de embarazo, es decir, que la ha clasificado como falso negativo.
- Y, por último, al cuarto paciente los resultados le han indicado que está embarazado, sin embargo, es hombre, por lo cual es imposible, su resultado es un falso positivo.

Lo anterior es un proceso que se realiza por cada instancia que se debe clasificar y nos permite calcular la exactitud y su tasa de error con las siguientes fórmulas.

Exactitud = (verdaderos positivos + verdaderos negativos) / total de datos.

Precisión = verdaderos positivos / (verdaderos positivos + falsos positivos)

***Recall* =** (verdaderos positivos / (verdaderos positivos + falsos positivos)

Error= (falsos positivos + falsos negativos) / total de datos

Actividad de repaso

Dentro de la predicción de datos, a qué definición corresponde la siguiente formulación de datos: (falsos positivos + falsos negativos) / total de datos

- Error.
- Entrenamiento.
- Testeo.
- Accuracy.
- F1 Score.

Justificación

Video de habilidades

Glosario

Referencias

Amat, J. (2016). Introducción a la regresión lineal múltiple. https://www.cienciadedatos.net/documentos/25_regresion_lineal_multiple

Gianareas, J. (2019). Regresión lineal en español con Python. https://platzi.com/blog/regresion-lineal-en-espanol-con-python/

[Imagen sin título sobre agrupación]. (s. f.). https://qu4nt.github.io/sklearn-doc-es/modules/clustering.html#clustering

[Imagen sin título sobre clasificadores]. (s. f.). https://qu4nt.github.io/sklearn-doc-es/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

[Imagen sin título sobre preprocesamiento]. (s. f.). https://qu4nt.github.io/sklearn-doc-es/auto_examples/preprocessing/plot_discretization_strategies.html

[Imagen sin título sobre reducción de dimensionalidad]. (s. f.). https://qu4nt.github.io/sklearn-doc-es/auto_examples/decomposition/plot_pca_iris.html

[Imagen sin título sobre regresión]. (s. f). https://qu4nt.github.io/sklearn-doc-es/auto_examples/ensemble/plot_adaboost_regression.html

[Imagen sin título sobre regresión lineal simple]. (s. f.). https://www.aprendemachinelearning.com/wp-content/uploads/2018/05/pizza_en_dinamarca.png

Zelada, C. (2017). Evaluación de modelos de clasificación. https://rpubs.com/chzelada/275494