

# APRENDE MACHINE LEARNING

---

TEORÍA +  
PRÁCTICA  
PYTHON

ESCRITO POR  
JUAN IGNACIO BAGNATO

BASADO EN  
EL CONTENIDO DEL BLOG



# ¿Qué es el Machine Learning?

Veamos algunas definiciones existentes sobre Machine Learning para intentar dar comprensión a esta revolucionaria materia.

## Definiendo Machine Learning

El Machine Learning -traducido al Español como Aprendizaje Automático ó Aprendizaje de máquinas- es un subcampo de la Inteligencia Artificial que busca resolver el “cómo construir programas de computadora que mejoran automáticamente adquiriendo experiencia”.

Esta definición implica que el programa que se crea con ML no necesita que el programador indique explícitamente las reglas que debe seguir para lograr su tarea si no que este mejora automáticamente.

En los últimos años han surgido grandes volúmenes de datos de diversas fuentes públicas -big data- y el Aprendizaje Automático relacionado al campo estadístico consiste en extraer y reconocer patrones y tendencias para comprender qué nos dicen los datos. Para ello, se vale de algoritmos que pueden procesar Gygas y/o Terabytes en tiempos razonables y obtener información útil.

## Una Definición Técnica

Podemos encontrar la siguiente definición técnica sobre Aprendizaje Automático:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

La experiencia E hace referencia a grandes volúmenes de datos recolectados (Big Data) para la toma de decisiones T y la forma de medir su desempeño P para comprobar que esos algoritmos mejoran con la adquisición de más experiencia.

## Diagrama de Venn

Drew Conway creó un simpático diagrama de Venn en el que inerrelaciona diversos campos. Aquí copio su versión al Español:



Diagrama de Venn

En esta aproximación al ML, podemos ver que es una intersección entre conocimientos de Matemáticas y Estadística con Habilidades de Hackeo del programador.

## Aproximación para programadores

Los programadores sabemos que los algoritmos de búsqueda pueden tomar mucho tiempo en concluir y que cuanto mayor sea el espacio de búsqueda crecerán exponencialmente las posibilidades de combinación de una respuesta óptima, haciendo que los tiempos de respuesta tiendan al infinito o que tomen más tiempo de lo que un ser humano pueda tolerar (por quedarse sin vida o por impaciencia).

Para poder resolver este tipo de situaciones surgen soluciones de tipo heurísticas que intentan dar «intuición» al camino correcto a tomar para resolver un problema. Estos logran buenos resultados en tiempos menores de procesamiento pero muchas veces su intuición es arbitraria y pueden fallar.

Los algoritmos de ML intentan utilizar menos recursos computacionales para «entrenar» grandes volúmenes de datos e ir aprendiendo por sí mismos. Podemos dividir el ML en 2 grandes categorías: Aprendizaje Supervisado o Aprendizaje No Supervisado. Hay una tercer categoría llamada “Aprendizaje por Refuerzo” pero no será tratada en este libro.

Entre los Algoritmos más utilizados en Inteligencia Artificial encontramos:

- Arboles de Decisión
- Regresión Lineal
- Regresión Logística
- k Nearest Neighbor
- PCA / Principal Component Analysis
- SVM
- Gaussian Naive Bayes
- K-Means
- Redes Neuronales Artificiales
- Aprendizaje Profundo ó Deep Learning

## **Una mención especial a las Redes Neuronales Artificiales**

Una mención distintiva merecen las “RNAs” ya que son algoritmos que imitan al comportamiento de las neuronas humanas y su capacidad de sinápsis para la obtención de resultados, interrelacionando diversas capas de neuronas para darle mayor poder de aprendizaje.

Aunque este código existe desde hace más de 70 años, en la última década han evolucionado notoriamente (en paralelo a la mayor capacidad tecnológica de procesamiento, memoria RAM y disco, la nube, etc.) y están logrando impresionantes resultados para analizar textos y síntesis de voz, traducción automática de idiomas, procesamiento de lenguaje natural, visión artificial, análisis de riesgo, clasificación y predicción y la creación de motores de recomendación.

## **Resumen**

El Machine Learning es una nueva herramienta clave que posibilitará el desarrollo de un futuro mejor para la humanidad brindando inteligencia a robots, coches y hogares. Las Smart Cities, el IOT (Internet of things) ya se está volviendo una realidad y también las aplicaciones de Machine Learning en asistentes como Siri, las recomendaciones de Netflix o sistemas de navegación autónoma en drones. Para los ingenieros o informáticos es una disciplina fundamental para modelar, construir y transitar este nuevo futuro.

# Regresión Lineal con Python

## ¿Qué es la regresión lineal?

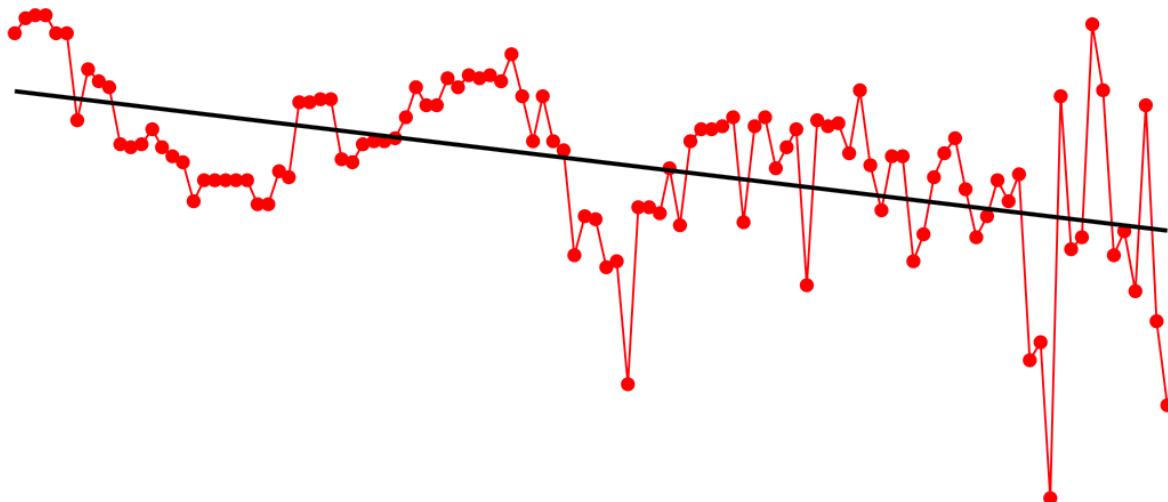
La [regresión lineal<sup>30</sup>](#) es un [algoritmo<sup>31</sup>](#) de [aprendizaje supervisado<sup>32</sup>](#) que se utiliza en Machine Learning y en estadística. En su versión más sencilla, lo que haremos es “dibujar una recta” que *nos indicará la tendencia* de un conjunto de datos continuos (si fueran discretos, utilizaríamos [Regresión Logística<sup>33</sup>](#)). En estadísticas, *regresión lineal es una aproximación para modelar la relación entre una variable escalar dependiente “y” y una o mas variables explicativas nombradas con “X”*.

Recordemos rápidamente la fórmula de la recta:

$$Y = mX + b$$

Donde Y es el resultado, X es la variable, m la pendiente (o coeficiente) de la recta y b la constante o también conocida como el “punto de corte con el eje Y” en la gráfica (cuando X=0)

### The development in Pizza prices in Denmark from 2009 to 2018



Aquí vemos un ejemplo donde vemos datos recabados sobre los precios de las pizzas en Dinamarca (los puntos en rojo) y la linea negra es la tendencia. Esa es la línea de regresión que buscamos que el algoritmo aprenda y calcule sólo.

<sup>30</sup>[https://es.wikipedia.org/wiki/Regresi%C3%B3n\\_lineal](https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal)

<sup>31</sup><http://www.aprendemachinelearning.com/principales-algoritmos-usados-en-machine-learning/>

<sup>32</sup><http://www.aprendemachinelearning.com/aplicaciones-del-machine-learning/#supervisado>

<sup>33</sup><http://www.aprendemachinelearning.com/regresion-logistica-con-python-paso-a-paso/>

## ¿Cómo funciona el algoritmo de regresión lineal en Machine Learning?

Recordemos que los algoritmos de Machine Learning Supervisados<sup>34</sup>, aprenden por sí mismos y -en este caso- a obtener automáticamente esa “recta” que buscamos con la tendencia de predicción. Para hacerlo se mide el error con respecto a los puntos de entrada y el valor “Y” de salida real. El algoritmo deberá minimizar el coste de una función de [error cuadrático](#)<sup>35</sup> y esos coeficientes corresponderán con la recta óptima. Hay diversos métodos para conseguir minimizar el coste. Lo más común es utilizar una versión vectorial y la llamada [Ecuación Normal](#)<sup>36</sup> que nos dará un resultado directo.

NOTA: cuando hablo de “recta” es en el caso particular de regresión lineal simple. Si hubiera más variables, hay que generalizar el término.

## Un Ejercicio Práctico

En este ejemplo cargaremos un [archivo .csv de entrada](#)<sup>37</sup> obtenido por [webscraping](#)<sup>38</sup> que contiene diversas URLs a artículos sobre Machine Learning de algunos sitios muy importantes como [Techcrunch](#)<sup>39</sup> o [KDnuggets](#)<sup>40</sup> y como características de entrada -las columnas- tendremos:

- Title: título del artículo
- url: ruta al artículo
- Word count: la cantidad de palabras del artículo,
- # of Links: los enlaces externos que contiene,
- # of comments: cantidad de comentarios,
- # Images video: suma de imágenes (o videos),
- Elapsed days: la cantidad de días transcurridos (al momento de crear el archivo)
- # Shares: nuestra columna de salida que será la “cantidad de veces que se compartió el artículo”.

A partir de las características de un artículo de machine learning intentaremos predecir, cuantas veces será compartido en Redes Sociales. Haremos una primer predicción de [regresión lineal simple](#)<sup>41</sup> - con una sola variable predictora- para poder graficar en 2 dimensiones (ejes X e Y) y luego un ejemplo de [regresión Lineal Múltiple](#), en la que utilizaremos 3 dimensiones (X,Y,Z) y predicciones.

NOTA: el archivo .csv contiene mitad de datos reales, y otra mitad los generados de manera aleatoria, por lo que las predicciones que obtendremos no serán reales.

<sup>34</sup><http://www.aprendemachinelearning.com/aplicaciones-del-machine-learning/#supervisado>

<sup>35</sup>[https://es.wikipedia.org/wiki/Error\\_cuadr%C3%A1tico\\_medio](https://es.wikipedia.org/wiki/Error_cuadr%C3%A1tico_medio)

<sup>36</sup>[https://en.wikipedia.org/wiki/Linear\\_least\\_squares\\_\(mathematics\)#Derivation\\_of\\_the\\_normal\\_equations](https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics)#Derivation_of_the_normal_equations)

<sup>37</sup>[http://www.aprendemachinelearning.com/articulos\\_ml/](http://www.aprendemachinelearning.com/articulos_ml/)

<sup>38</sup><http://www.aprendemachinelearning.com/ejemplo-web-scraping-python-ibex35-bolsa-valores/>

<sup>39</sup><https://techcrunch.com/tag/machine-learning/>

<sup>40</sup><https://www.kdnuggets.com>

<sup>41</sup>[https://en.wikipedia.org/wiki/Simple\\_linear\\_regression](https://en.wikipedia.org/wiki/Simple_linear_regression)

## Requerimientos para hacer el Ejercicio

Para realizar este ejercicio, crearemos una [Jupyter notebook<sup>42</sup>](#) con código Python y la librería Scikit-Learn muy utilizada en Data Science. Recomendamos utilizar la suite de [Anaconda<sup>43</sup>](#). Podrás descargar los archivos de [entrada csv<sup>44</sup>](#) o visualizar la [notebook online<sup>45</sup>](#).

## Predecir cuántas veces será compartido un artículo de Machine Learning.

### Regresión lineal simple en Python (con 1 variable)

Aquí vamos con nuestra notebook! Comencemos por importar las librerías que utilizaremos:

```

1 # Imports necesarios
2 import numpy as np
3 import pandas as pd
4 import seaborn as sb
5 import matplotlib.pyplot as plt
6 %matplotlib inline
7 from mpl_toolkits.mplot3d import Axes3D
8 from matplotlib import cm
9 plt.rcParams['figure.figsize'] = (16, 9)
10 plt.style.use('ggplot')
11 from sklearn import linear_model
12 from sklearn.metrics import mean_squared_error, r2_score

```

Leemos el archivo csv y lo cargamos como un dataset de Pandas. Y vemos su tamaño.

```

1 #cargamos los datos de entrada
2 data = pd.read_csv("articulos_ml.csv")
3 #veamos cuantas dimensiones y registros contiene
4 data.shape

```

Nos devuelve (161,8) Veamos esas primeras filas:

---

<sup>42</sup><http://data-speaks.luca-d3.com/2018/03/python-para-todos-2-jupyter-notebook.html>

<sup>43</sup><https://www.anaconda.com/download/>

<sup>44</sup>[http://www.aprendemachinelearning.com/articulos\\_ml/](http://www.aprendemachinelearning.com/articulos_ml/)

<sup>45</sup>[https://github.com/jbagnato/machine-learning/blob/master/Ejercicio\\_Regresion\\_Lineal.ipynb](https://github.com/jbagnato/machine-learning/blob/master/Ejercicio_Regresion_Lineal.ipynb)

```

1 #son 161 registros con 8 columnas. Veamos los primeros registros
2 data.head()

```

	Title	url	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
0	What is Machine Learning and how do we use it ...	https://blog.signals.network/what-is-machine-l...	1888	1	2.0	2	34	200000
1	10 Companies Using Machine Learning in Cool Ways	NaN	1742	9	NaN	9	5	25000
2	How Artificial Intelligence Is Revolutionizing...	NaN	962	6	0.0	1	10	42000
3	Dbrain and the Blockchain of Artificial Intell...	NaN	1221	3	NaN	2	68	200000
4	Nasa finds entire solar system filled with eig...	NaN	2039	1	104.0	4	131	200000

<sup>46</sup>

Se ven algunos campos con valores NaN (nulos) por ejemplo algunas urls o en comentarios. Veamos algunas estadísticas básicas de nuestros datos de entrada:

```

1 # Ahora veamos algunas estadísticas de nuestros datos
2 data.describe()

```

	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
count	161.000000	161.000000	129.000000	161.000000	161.000000	161.000000
mean	1808.260870	9.739130	8.782946	3.670807	98.124224	27948.347826
std	1141.919385	47.271625	13.142822	3.418290	114.337535	43408.006839
min	250.000000	0.000000	0.000000	1.000000	1.000000	0.000000
25%	990.000000	3.000000	2.000000	1.000000	31.000000	2800.000000
50%	1674.000000	5.000000	6.000000	3.000000	62.000000	16458.000000
75%	2369.000000	7.000000	12.000000	5.000000	124.000000	35691.000000
max	8401.000000	600.000000	104.000000	22.000000	1002.000000	350000.000000

<sup>47</sup>

Aquí vemos que la media de palabras en los artículos es de 1808. El artículo más corto tiene 250 palabras y el más extenso 8401. Intentaremos ver con nuestra relación lineal, si hay una correlación entre la cantidad de palabras del texto y la cantidad de Shares obtenidos. Hacemos una visualización en general de los datos de entrada:

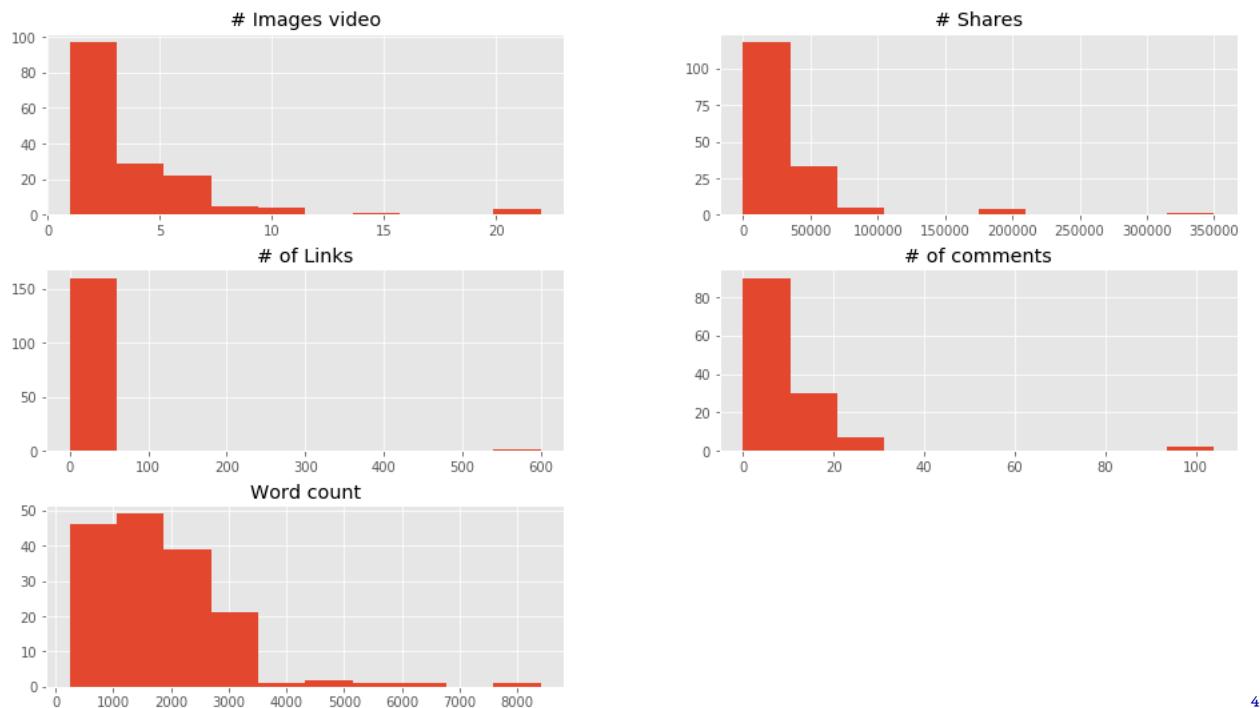
```

1 # Visualizamos rápidamente las características de entrada
2 data.drop(['Title','url', 'Elapsed days'],1).hist()
3 plt.show()

```

<sup>46</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg\\_lineal\\_filas\\_iniciales.png](http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg_lineal_filas_iniciales.png)

<sup>47</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg\\_lineal\\_stats\\_base.png](http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg_lineal_stats_base.png)



48

En estas gráficas vemos entre qué valores se concentran la mayoría de registros. Vamos a filtrar los datos de cantidad de palabras para quedarnos con los registros con menos de 3500 palabras y también con los que tengan Cantidad de compartidos menor a 80000. Lo gratificaremos pintando en azul los puntos con menos de 1808 palabras (la media) y en naranja los que tengan más.

```

1 # Vamos a RECORTAR los datos en la zona donde se concentran más los puntos
2 # esto es en el eje X: entre 0 y 3.500
3 # y en el eje Y: entre 0 y 80.000
4 filtered_data = data[(data['Word count'] <= 3500) & (data['# Shares'] <= 80000)]
5
6 colores=['orange','blue']
7 tamanios=[30,60]
8
9 f1 = filtered_data['Word count'].values
10 f2 = filtered_data['# Shares'].values
11
12 # Vamos a pintar en colores los puntos por debajo y por encima de la media de Cantidad de Palabras
13 asignar=[]
14 for index, row in filtered_data.iterrows():
15     if(row['Word count']>1808):
16         asignar.append(colores[0])
17     else:
18         asignar.append(colores[1])

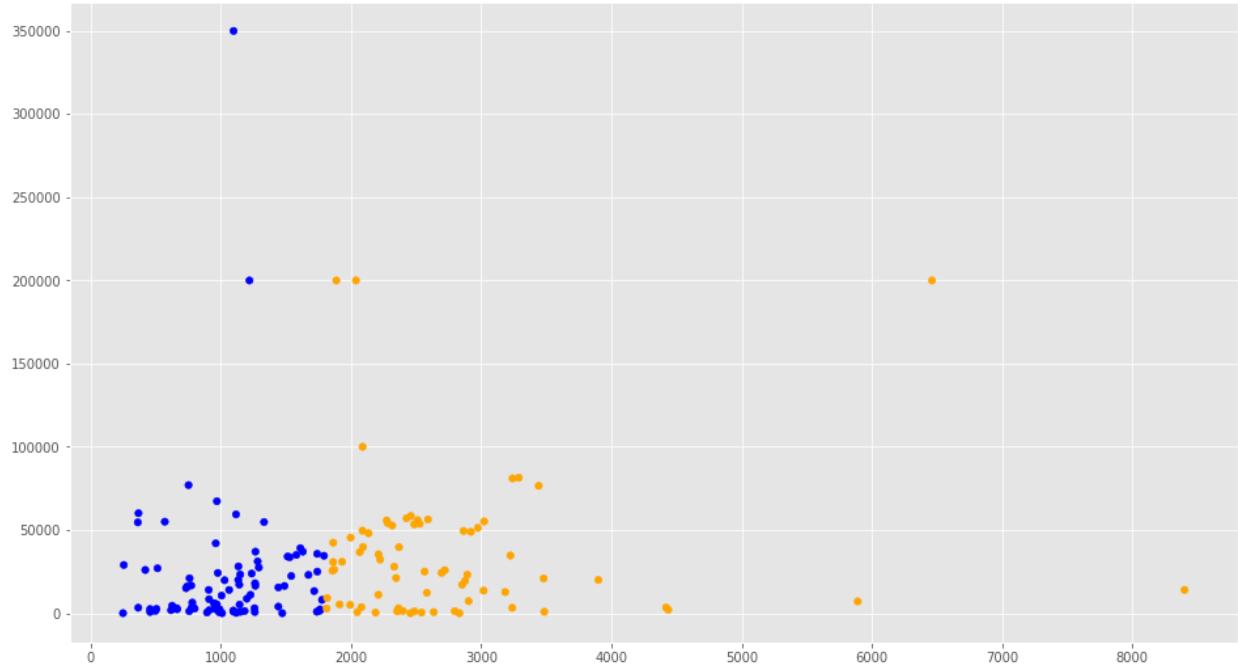
```

<sup>48</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg\\_lin\\_visualiza\\_entradas.png](http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg_lin_visualiza_entradas.png)

```

19         asignar.append(colores[1])
20
21 plt.scatter(f1, f2, c=asignar, s=tamanios[0])
22 plt.show()

```



49

## Regresión Lineal con Python y SKLearn

Vamos a crear nuestros datos de entrada por el momento sólo Word Count y como etiquetas los # Shares. Creamos el objeto LinearRegression y lo hacemos “encajar” (entrenar) con el método fit(). Finalmente imprimimos los coeficientes y puntajes obtenidos.

```

1 # Asignamos nuestra variable de entrada X para entrenamiento y las etiquetas Y.
2 dataX = filtered_data[["Word count"]]
3 X_train = np.array(dataX)
4 y_train = filtered_data['# Shares'].values
5
6 # Creamos el objeto de Regresión Lineal
7 regr = linear_model.LinearRegression()
8
9 # Entrenamos nuestro modelo
10 regr.fit(X_train, y_train)

```

<sup>49</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg\\_lineal\\_grafica\\_pal\\_vs\\_shares.png](http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg_lineal_grafica_pal_vs_shares.png)

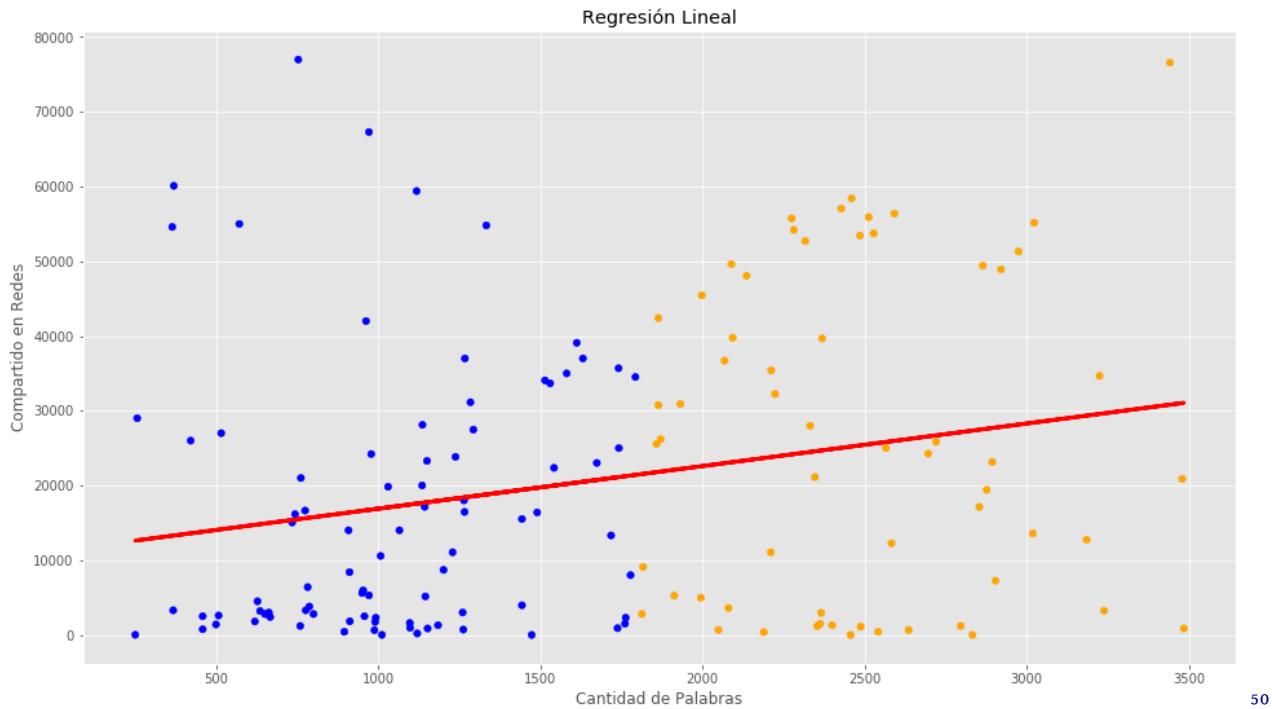
```
11
12 # Hacemos las predicciones que en definitiva una línea (en este caso, al ser 2D)
13 y_pred = regr.predict(X_train)
14
15 # Veamos los coeficientes obtenidos, En nuestro caso, serán la Tangente
16 print('Coefficients: \n', regr.coef_)
17 # Este es el valor donde corta el eje Y (en X=0)
18 print('Independent term: \n', regr.intercept_)
19 # Error Cuadrado Medio
20 print("Mean squared error: %.2f" % mean_squared_error(y_train, y_pred))
21 # Puntaje de Varianza. El mejor puntaje es un 1.0
22 print('Variance score: %.2f' % r2_score(y_train, y_pred))
```

```
1 Coefficients: [5.69765366]
2 Independent term: 11200.303223074163
3 Mean squared error: 372888728.34
4 Variance score: 0.06
```

De la ecuación de la recta  $y = mX + b$  nuestra pendiente “m” es el coeficiente 5,69 y el término independiente “b” es 11200. Tenemos un Error Cuadrático medio enorme... por lo que en realidad este modelo no será muy bueno prediciendo ;) Pero estamos aprendiendo a usarlo, que es lo que nos importa ahora :) Esto también se ve reflejado en el puntaje de Varianza que debería ser cercano a 1.0.

## Visualicemos la Recta

Veamos la recta que obtuvimos:



## Predicción en regresión lineal simple

Vamos a intentar probar nuestro algoritmo, suponiendo que quisiéramos predecir cuántos “compartir” obtendrá un artículo sobre ML de 2000 palabras.

```

1 #Vamos a comprobar:
2 # Quiero predecir cuántos "Shares" voy a obtener por un artículo con 2.000 palabras,
3 # según nuestro modelo, hacemos:
4 y_Dosmil = regr.predict([[2000]])
5 print(int(y_Dosmil))

```

Nos devuelve una predicción de 22595 “Shares” para un artículo de 2000 palabras.

---

<sup>50</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg\\_lineal\\_recta\\_1\\_variable.png](http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/reg_lineal_recta_1_variable.png)

# Regresión Lineal Múltiple en Python

## (o “Regresión con Múltiples Variables”)

Vamos a extender el ejercicio utilizando más de una variable de entrada para el modelo. Esto le da **mayor poder** al algoritmo de Machine Learning, pues de esta manera podremos obtener predicciones más complejas. Nuestra “ecuación de la Recta”, ahora pasa a ser:

$$Y = b + m_1 X_1 + m_2 X_2 + \dots + m(n) X(n)$$

(y deja de ser una recta) **En nuestro caso, utilizaremos 2 “variables predictivas” para poder graficar en 3D**, pero recordar que *para mejores predicciones podemos utilizar más de 2 entradas* y prescindir del gráfico. Nuestra primer variable seguirá siendo la **cantidad de palabras** y la segunda variable la crearemos artificialmente y será la **suma de 3 columnas de entrada**: la cantidad de enlaces, comentarios y cantidad de imágenes. Vamos a programar!

```

1 #Vamos a intentar mejorar el Modelo, con una dimensión más:
2 # Para poder graficar en 3D, haremos una variable nueva que será la suma de los enla\
3 ces, comentarios e imágenes
4 suma = (filtered_data["# of Links"] + filtered_data['# of comments'].fillna(0) + fil\
5 tered_data['# Images video'])
6
7 dataX2 = pd.DataFrame()
8 dataX2["Word count"] = filtered_data["Word count"]
9 dataX2["suma"] = suma
10 XY_train = np.array(dataX2)
11 z_train = filtered_data['# Shares'].values

```

Ya tenemos nuestras 2 variables de entrada en XY\_train y **nuestra variable de salida pasa de ser “Y” a ser el eje “Z”**. Creamos un nuevo objeto de Regresión lineal con SKLearn pero esta vez tendrá las dos dimensiones que entrenar: las que contiene XY\_train. Al igual que antes, imprimimos los coeficientes y puntajes obtenidos:

```
1 # Creamos un nuevo objeto de Regresión Lineal
2 regr2 = linear_model.LinearRegression()
3
4 # Entrenamos el modelo, esta vez, con 2 dimensiones
5 # obtendremos 2 coeficientes, para graficar un plano
6 regr2.fit(XY_train, z_train)
7
8 # Hacemos la predicción con la que tendremos puntos sobre el plano hallado
9 z_pred = regr2.predict(XY_train)
10
11 # Los coeficientes
12 print('Coefficients: \n', regr2.coef_)
13 # Error cuadrático medio
14 print("Mean squared error: %.2f" % mean_squared_error(z_train, z_pred))
15 # Evaluamos el puntaje de varianza (siendo 1.0 el mejor posible)
16 print('Variance score: %.2f' % r2_score(z_train, z_pred))

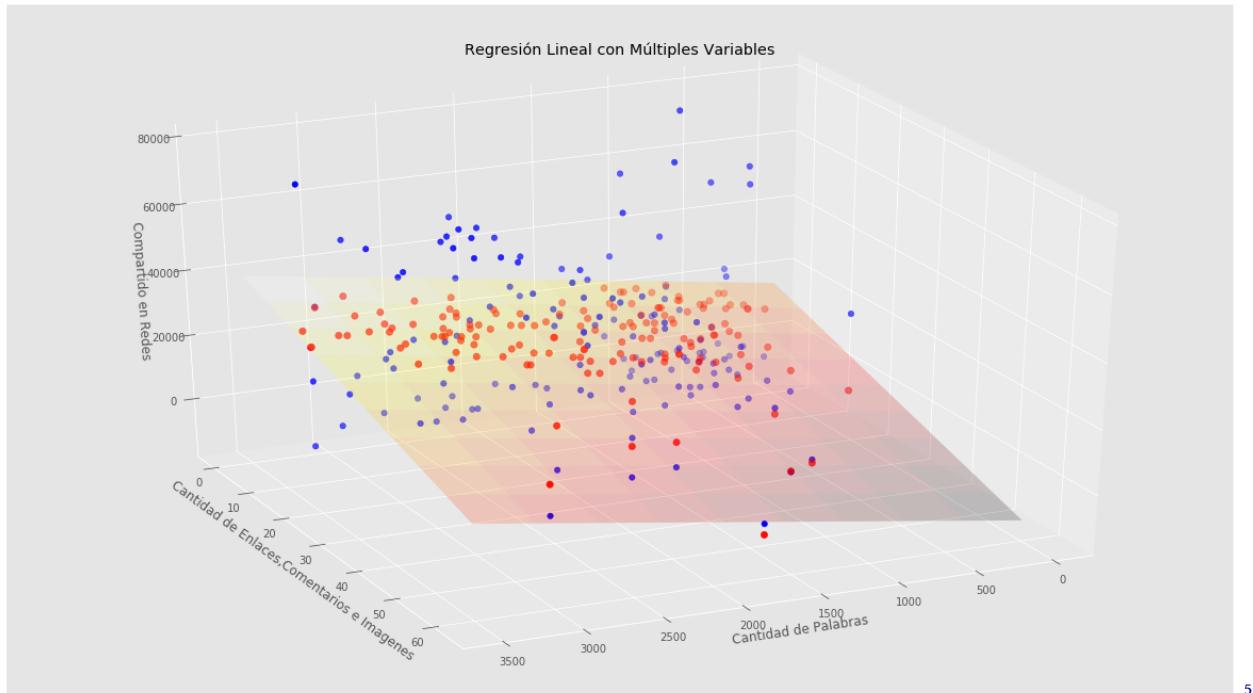
1 Coefficients: [ 6.63216324 -483.40753769]
2 Mean squared error: 352122816.48
3 Variance score: 0.11
```

Como vemos, obtenemos 2 coeficientes (cada uno correspondiente a nuestras 2 variables predictivas), pues ahora lo que graficamos no será una linea si no, **un plano en 3 Dimensiones**. El error obtenido sigue siendo grande, aunque algo mejor que el anterior y el puntaje de Varianza mejora casi el doble del anterior (aunque sigue siendo muy malo, muy lejos del 1).

## Visualizar un plano en 3 Dimensiones en Python

Graficaremos nuestros puntos de las características de entrada en color azul y los puntos proyectados en el plano en rojo. Recordemos que en esta gráfica, el eje Z corresponde a la “altura” y representa la cantidad de Shares que obtendremos.

```
1 fig = plt.figure()
2 ax = Axes3D(fig)
3
4 # Creamos una malla, sobre la cual graficaremos el plano
5 xx, yy = np.meshgrid(np.linspace(0, 3500, num=10), np.linspace(0, 60, num=10))
6
7 # calculamos los valores del plano para los puntos x e y
8 nuevoX = (regr2.coef_[0] * xx)
9 nuevoY = (regr2.coef_[1] * yy)
10
11 # calculamos los correspondientes valores para z. Debemos sumar el punto de intercep\
12 ción
13 z = (nuevoX + nuevoY + regr2.intercept_)
14
15 # Graficamos el plano
16 ax.plot_surface(xx, yy, z, alpha=0.2, cmap='hot')
17
18 # Graficamos en azul los puntos en 3D
19 ax.scatter(XY_train[:, 0], XY_train[:, 1], z_train, c='blue', s=30)
20
21 # Graficamos en rojo, los puntos que
22 ax.scatter(XY_train[:, 0], XY_train[:, 1], z_pred, c='red', s=40)
23
24 # con esto situamos la "camara" con la que visualizamos
25 ax.view_init(elev=30., azim=65)
26
27 ax.set_xlabel('Cantidad de Palabras')
28 ax.set_ylabel('Cantidad de Enlaces, Comentarios e Imagenes')
29 ax.set_zlabel('Compartido en Redes')
30 ax.set_title('Regresión Lineal con Múltiples Variables')
```



Podemos rotar el gráfico para apreciar el plano desde diversos ángulos modificando el valor del parámetro `azim` en `view_init` con números de 0 a 360.

## Predicción con el modelo de Múltiples Variables

Veamos ahora, que predicción tendremos para un artículo de 2000 palabras, con 10 enlaces, 4 comentarios y 6 imágenes.

```

1 # Si quiero predecir cuántos "Shares" voy a obtener por un artículo con:
2 # 2000 palabras y con enlaces: 10, comentarios: 4, imágenes: 6
3
4 z_Dosmil = regr2.predict([[2000, 10+4+6]])
5 print(int(z_Dosmil))

```

Esta predicción nos da 20518 y probablemente sea un poco mejor que nuestra predicción anterior con 1 variables.

## Resumen

Hemos visto cómo utilizar SKLearn en Python para crear modelos de Regresión Lineal con 1 o múltiples variables. En nuestro ejercicio no tuvimos una gran confianza en las predicciones. Por

---

<sup>51</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/regresion\\_lineal\\_3d\\_plano.png](http://www.aprendemachinelearning.com/wp-content/uploads/2018/05/regresion_lineal_3d_plano.png)

ejemplo en nuestro primer modelo, con 2000 palabras nos predice que podemos tener 22595 pero el margen de error haciendo raíz del error cuartico medio es más menos 19310. Es decir que escribiendo un artículo de 2000 palabras lo mismo tenemos 3285 Shares que 41905. En este caso usamos este modelo para aprender a usarlo y habrá que ver en otros casos en los que sí nos brinde predicciones acertadas. Para mejorar nuestro modelo, deberíamos utilizar más dimensiones y encontrar datos de entrada mejores.

**Atención:** también es posible, que no exista ninguna relación fuerte entre nuestras variables de entrada y el éxito en Shares del artículo...

## Recursos y enlaces

- Descarga la [Jupyter Notebook<sup>52</sup>](#) y el [archivo de entrada csv<sup>53</sup>](#)
- ó puedes [visualizar online<sup>54</sup>](#)
- o ver y descargar desde mi cuenta [github<sup>55</sup>](#)

Otros enlaces con Artículos sobre Regresión Lineal (en Inglés)

- [Introduction to Linear Regression using python<sup>56</sup>](#)
- [Linear Regression using Python SkLearn<sup>57</sup>](#)
- [Linear Regression Detailed View<sup>58</sup>](#)
- [How do you solve a linear regression problem in python<sup>59</sup>](#)
- [Python tutorial on LinearRegression with Batch Gradient Descent<sup>60</sup>](#)

---

<sup>52</sup>[http://www.aprendemachinelearning.com/ejercicio\\_Regresion\\_lineal/](http://www.aprendemachinelearning.com/ejercicio_Regresion_lineal/)

<sup>53</sup>[http://www.aprendemachinelearning.com/articulos\\_ml/](http://www.aprendemachinelearning.com/articulos_ml/)

<sup>54</sup>[https://github.com/jbagnato/machine-learning/blob/master/Ejercicio\\_Regresion\\_Lineal.ipynb](https://github.com/jbagnato/machine-learning/blob/master/Ejercicio_Regresion_Lineal.ipynb)

<sup>55</sup><https://github.com/jbagnato/machine-learning>

<sup>56</sup><https://aktechthoughts.wordpress.com/2018/03/26/introduction-to-linear-regression-using-python/>

<sup>57</sup><https://dzone.com/articles/linear-regression-using-python-scikit-learn>

<sup>58</sup><https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86>

<sup>59</sup><http://lineardata.net/how-do-you-solve-a-linear-regression-machine-learning-problem-in-python/>

<sup>60</sup><http://ozzieliu.com/2016/02/09/gradient-descent-tutorial/>

# Regresión Logística

## Introducción

Utilizaremos algoritmos de Machine Learning en Python para resolver un problema de Regresión Logística<sup>61</sup>. A partir de un conjunto de datos de entrada (características), nuestra salida será discreta (y no continua) por eso utilizamos Regresión Logística (y no Regresión Lineal<sup>62</sup>). La Regresión Logística es un Algoritmo Supervisado<sup>63</sup> y se utiliza para clasificación<sup>64</sup>.

Vamos a clasificar problemas con dos posibles estados “SI/NO”: binario o un número finito de “etiquetas” o “clases”: múltiple.

Algunos Ejemplos de Regresión Logística son:

- Clasificar si el correo que llega es Spam o No es Spam.
- Dados unos resultados clínicos de un tumor clasificar en “Benigno” o “Maligno”.
- El texto de un artículo a analizar es: Entretenimiento, Deportes, Política ó Ciencia.
- A partir de historial bancario conceder un crédito o no.

Confiaremos en la implementación del paquete Scikit-learn de Python para ponerlo en práctica.

## Ejercicio de Regresión Logística en Python

Para el ejercicio he creado un archivo csv<sup>65</sup> con datos de entrada a modo de ejemplo para clasificar si el usuario que visita un sitio web usa como sistema operativo Windows, Macintosh o Linux. Nuestra información de entrada son 4 características que tomé de una web que utiliza Google Analytics y son:

- Duración de la visita en Segundos
- Cantidad de Páginas Vistas durante la Sesión
- Cantidad de Acciones del usuario (click, scroll, uso de checkbox, etc)
- Suma del Valor de las acciones (cada acción lleva asociada una valoración de importancia)

---

<sup>61</sup>[https://en.wikipedia.org/wiki/Logistic\\_regression](https://en.wikipedia.org/wiki/Logistic_regression)

<sup>62</sup><http://www.aprendemachinelearning.com/regresion-lineal-en-espanol-con-python/>

<sup>63</sup><http://www.aprendemachinelearning.com/aplicaciones-del-machine-learning/#supervisado>

<sup>64</sup><http://www.aprendemachinelearning.com/principales-algoritmos-usados-en-machine-learning/>

<sup>65</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2017/11/usuarios\\_win\\_mac\\_lin.csv](http://www.aprendemachinelearning.com/wp-content/uploads/2017/11/usuarios_win_mac_lin.csv)

Como la salida es discreta, asignaremos los siguientes valores a las etiquetas:

- 0 - Windows
- 1 - Macintosh
- 2 - Linux

La muestra es pequeña: son 170 registros para poder comprender el ejercicio, pero recordemos que para conseguir buenos resultados siempre es mejor contar con un número abundante de datos que darán mayor exactitud a las predicciones.

## Regresión Logística con SKLearn:

### Identificar Sistema Operativo de los usuarios

Para comenzar hacemos los Import necesarios con los paquetes que utilizaremos en el Ejercicio.

```
1 import pandas as pd
2 import numpy as np
3 from sklearn import linear_model
4 from sklearn import model_selection
5 from sklearn.metrics import classification_report
6 from sklearn.metrics import confusion_matrix
7 from sklearn.metrics import accuracy_score
8 import matplotlib.pyplot as plt
9 import seaborn as sb
10 %matplotlib inline
```

Leemos el archivo csv (por sencillez, se considera que estará en el mismo directorio que el archivo de notebook .ipynb) y lo asignamos mediante Pandas a la variable **dataframe**. Mediante el método **dataframe.head()** vemos en pantalla los 5 primeros registros.

```
1 dataframe = pd.read_csv(r"usuarios_win_mac_lin.csv")
2 dataframe.head()
```

	<b>duracion</b>	<b>paginas</b>	<b>acciones</b>	<b>valor</b>	<b>clase</b>
<b>0</b>	7.0	2	4	8	2
<b>1</b>	21.0	2	6	6	2
<b>2</b>	57.0	2	4	4	2
<b>3</b>	101.0	3	6	12	2
<b>4</b>	109.0	2	6	12	2

A continuación llamamos al método `dataframe.describe()` que nos dará algo de información estadística básica de nuestro set de datos. La Media, el desvío estándar, valores mínimo y máximo de cada característica.

```
1 dataframe.describe()
```

	<b>duracion</b>	<b>paginas</b>	<b>acciones</b>	<b>valor</b>	<b>clase</b>
<b>count</b>	170.000000	170.000000	170.000000	170.000000	170.000000
<b>mean</b>	111.075729	2.041176	8.723529	32.676471	0.752941
<b>std</b>	202.453200	1.500911	9.136054	44.751993	0.841327
<b>min</b>	1.000000	1.000000	1.000000	1.000000	0.000000
<b>25%</b>	11.000000	1.000000	3.000000	8.000000	0.000000
<b>50%</b>	13.000000	2.000000	6.000000	20.000000	0.000000
<b>75%</b>	108.000000	2.000000	10.000000	36.000000	2.000000
<b>max</b>	898.000000	9.000000	63.000000	378.000000	2.000000

Luego analizaremos cuantos resultados tenemos de cada tipo usando la función `groupby` y vemos que tenemos 86 usuarios “Clase 0”, es decir Windows, 40 usuarios Mac y 44 de Linux.

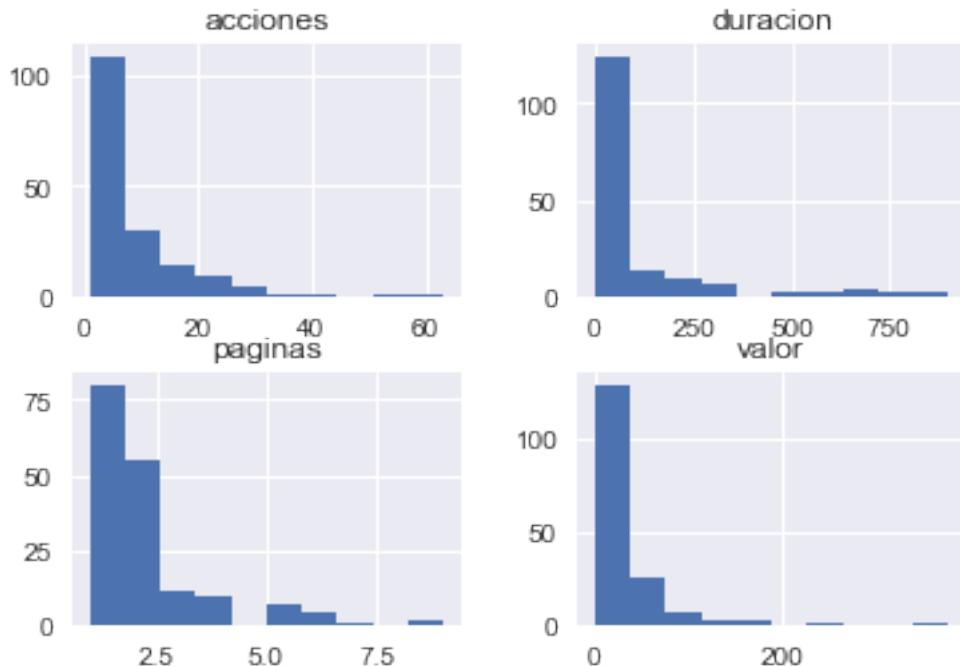
```
1 print(dataframe.groupby('clase').size())
```

```
clase
0    86
1    40
2    44
dtype: int64
```

## Visualización de Datos

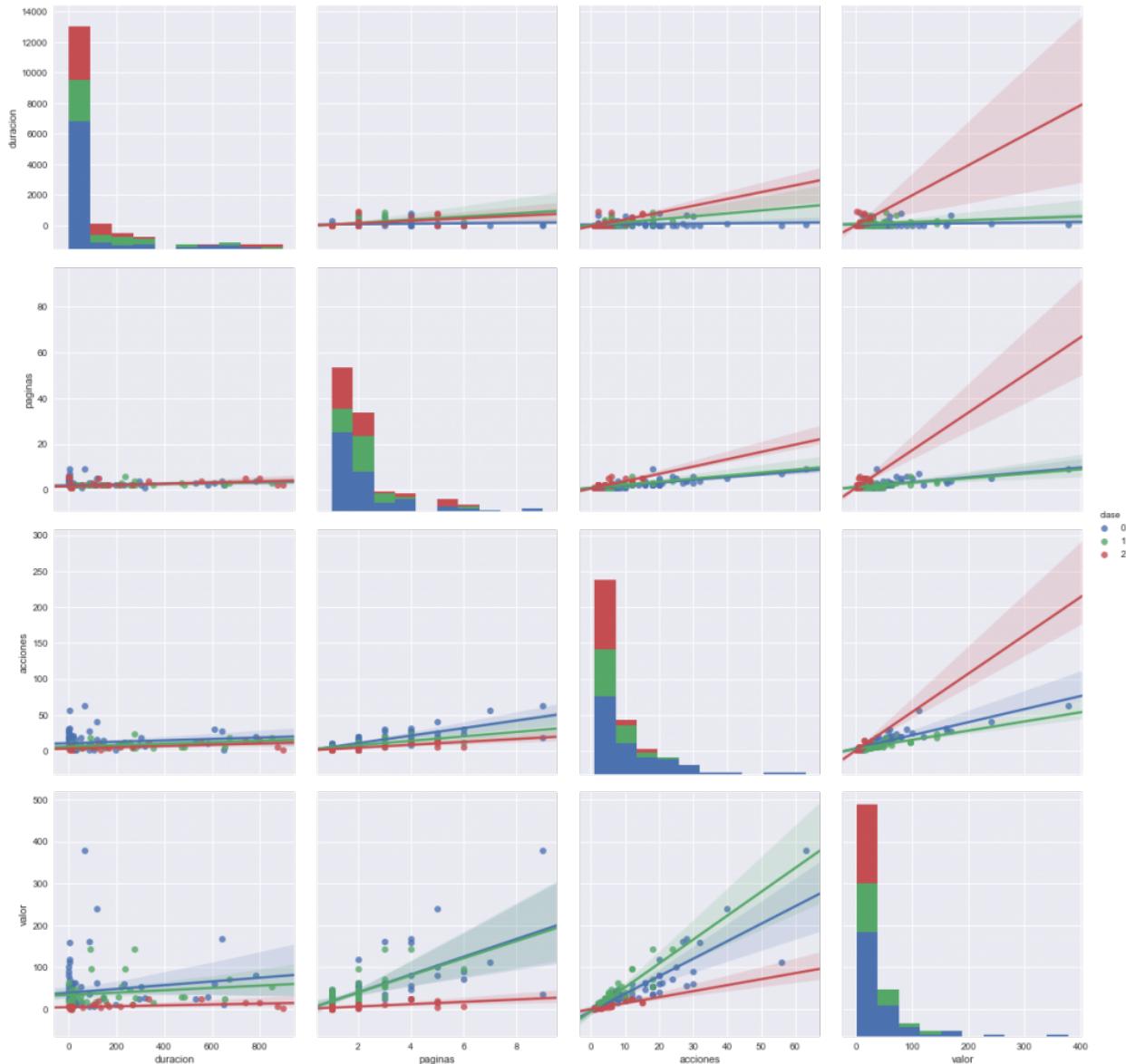
Antes de empezar a procesar el conjunto de datos, vamos a hacer unas visualizaciones que muchas veces nos pueden ayudar a comprender mejor las características de la información con la que trabajamos y su correlación. Primero visualizamos en formato de historial los cuatro Features de entrada con nombres “duración”, “páginas”, “acciones” y “valor” podemos ver gráficamente entre qué valores se comprenden sus mínimos y máximos y en qué intervalos concentran la mayor densidad de registros.

```
1 dataframe.drop(['clase'],1).hist()
2 plt.show()
```



Y también podemos interrelacionar las entradas de a pares, para ver como se concentran linealmente las salidas de usuarios por colores: Sistema Operativo Windows en azul, Macintosh en verde y Linux en rojo.

```
1 sb.pairplot(dataframe.dropna(), hue='clase', size=4, vars=["duracion", "paginas", "acciones", "valor"], kind='reg')
```



## Creamos el Modelo de Regresión Logística

Ahora cargamos las variables de las 4 columnas de entrada en X excluyendo la columna “clase” con el método drop(). En cambio agregamos la columna “clase” en la variable y. Ejecutamos X.shape para comprobar la dimensión de nuestra matriz con datos de entrada de 170 registros por 4 columnas.

```
1 X = np.array(dataframe.drop(['clase'],1))
2 y = np.array(dataframe['clase'])
3 X.shape
```

(170, 4) Y creamos nuestro modelo y hacemos que se ajuste (fit) a nuestro conjunto de entradas X y salidas ‘y’.

```
1 model = linear_model.LogisticRegression()
2 model.fit(X,y)
```

Una vez compilado nuestro modelo, le hacemos clasificar todo nuestro conjunto de entradas X utilizando el método “predict(X)” y revisamos algunas de sus salidas y vemos que coincide con las salidas reales de nuestro archivo csv.

```
1 predictions = model.predict(X)
2 print(predictions)[0:5]
```

```
1 [2 2 2 2 2]
```

Y confirmamos cuan bueno fue nuestro modelo utilizando model.score() que nos devuelve la precisión media de las predicciones, en nuestro caso del 77%.

```
1 model.score(X,y)
```

```
1 0.77647058823529413
```

## Validación de nuestro modelo

Una buena práctica en Machine Learning es la de subdividir nuestro conjunto de datos de entrada en un set de entrenamiento y otro para validar el modelo (que no se utiliza durante el entrenamiento y por lo tanto la máquina desconoce). Esto evitará problemas en los que nuestro algoritmo pueda fallar por “**sobregeneralizar**” el **conocimiento**<sup>66</sup>. Para ello, dividimos nuestros datos de entrada en forma aleatoria (mezclados) utilizando 80% de registros para entrenamiento y 20% para testear.

---

<sup>66</sup><http://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/>

```

1 validation_size = 0.20
2 seed = 7
3 X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, y\
4 , test_size=validation_size, random_state=seed)

```

Volvemos a compilar nuestro modelo de Regresión Logística pero esta vez sólo con 80% de los datos de entrada y calculamos el nuevo scoring que ahora nos da 74%.

```

1 name='Logistic Regression'
2 kfold = model_selection.KFold(n_splits=10, random_state=seed)
3 cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scor\
4 ing='accuracy')
5 msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
6 print(msg)

```

```
1 Logistic Regression: 0.743407 (0.115752)
```

Y ahora hacemos las predicciones -en realidad clasificación- utilizando nuestro “test set”, es decir del subconjunto que habíamos apartado. En este caso vemos que los aciertos fueron del 85% pero hay que tener en cuenta que el tamaño de datos era pequeño.

```

1 predictions = model.predict(X_validation)
2 print(accuracy_score(Y_validation, predictions))

```

```
1 0.852941176471
```

Finalmente vemos en pantalla la “matriz de confusión” donde muestra cuantos resultados equivocados tuvo de cada clase (los que no están en la diagonal), por ejemplo predijo 3 usuarios que eran Mac como usuarios de Windows y predijo a 2 usuarios Linux que realmente eran de Windows.

## Reporte de Resultados del Modelo

```
1 print(confusion_matrix(Y_validation, predictions))
```

```

[[16   0   2]
 [ 3   3   0]
 [ 0   0 10]]

```

También podemos ver el reporte de clasificación con nuestro conjunto de Test. En nuestro caso vemos que se utilizaron como “soporte” 18 registros windows, 6 de mac y 10 de Linux (total de 34 registros). Podemos ver la precisión con que se acertaron cada una de las clases y vemos que por ejemplo de Macintosh tuvo 3 aciertos y 3 fallos (0.5 recall). La valoración que de aquí nos conviene tener en cuenta es la de [F1-score<sup>67</sup>](#), que tiene en cuenta la precisión y recall. El promedio de F1 es de 84% lo cual no está nada mal.

```
1 print(classification_report(Y_validation, predictions))
```

	<b>precision</b>	<b>recall</b>	<b>f1-score</b>	<b>support</b>
0	0.84	0.89	0.86	18
1	1.00	0.50	0.67	6
2	0.83	1.00	0.91	10
<b>avg / total</b>	<b>0.87</b>	<b>0.85</b>	<b>0.84</b>	<b>34</b>

## Clasificación de nuevos valores

Como último ejercicio, vamos a inventar los datos de entrada de navegación de un usuario ficticio que tiene estos valores:

- Tiempo Duración: 10
- Páginas visitadas: 3
- Acciones al navegar: 5
- Valoración: 9

Lo probamos en nuestro modelo y vemos que lo clasifica como un usuario tipo 2, es decir, de Linux.

```
1 X_new = pd.DataFrame({'duracion': [10], 'paginas': [3], 'acciones': [5], 'valor': [9]}
2 })
3 model.predict(X_new)
```

---

<sup>67</sup>[https://en.wikipedia.org/wiki/F1\\_score](https://en.wikipedia.org/wiki/F1_score)

```
1 array([2])
```

Los invito a jugar y variar estos valores para obtener usuarios de tipo Windows o Macintosh.

## Resumen

Durante este artículo vimos cómo crear un modelo de Regresión Logística en Python para poder clasificar el Sistema Operativo de usuarios a partir de sus características de navegación en un sitio web. Este ejemplo se podrá extender a otro tipos de tareas que pueden surgir durante nuestro trabajo en el que deberemos clasificar resultados en valores discretos. Si tuviéramos que predecir valores continuos, deberemos aplicar [Regresión Lineal](#)<sup>68</sup>.

Recuerda descargar los archivos para realizar el Ejercicio:

- [Archivo de Entrada csv](#)<sup>69</sup> (su nombre es usuarios\_win\_mac\_lin.csv)
- [Notebook Jupyter Python](#)<sup>70</sup> (clic derecho y “descargar archivo como...”)
- OPCIÓN 2: Se puede ver online en [Jupyter Notebook Viewer](#)<sup>71</sup>
- OPCIÓN 3: Se puede [visualizar y descargar el Notebook](#)<sup>72</sup> y el [csv](#)<sup>73</sup> desde mi cuenta de [Github](#)<sup>74</sup>

---

<sup>68</sup><http://www.aprendemachinelearning.com/regresion-lineal-en-espanol-con-python/>

<sup>69</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2017/11/usuarios\\_win\\_mac\\_lin.csv](http://www.aprendemachinelearning.com/wp-content/uploads/2017/11/usuarios_win_mac_lin.csv)

<sup>70</sup>[http://www.aprendemachinelearning.com/wp-content/uploads/2017/11/Regresion\\_logistica.ipynb](http://www.aprendemachinelearning.com/wp-content/uploads/2017/11/Regresion_logistica.ipynb)

<sup>71</sup>[http://nbviewer.jupyter.org/github/jbagnato/machine-learning/blob/master/Regresion\\_logistica.ipynb](http://nbviewer.jupyter.org/github/jbagnato/machine-learning/blob/master/Regresion_logistica.ipynb)

<sup>72</sup>[https://github.com/jbagnato/machine-learning/blob/master/Regresion\\_logistica.ipynb](https://github.com/jbagnato/machine-learning/blob/master/Regresion_logistica.ipynb)

<sup>73</sup>[https://github.com/jbagnato/machine-learning/blob/master/usuarios\\_win\\_mac\\_lin.csv](https://github.com/jbagnato/machine-learning/blob/master/usuarios_win_mac_lin.csv)

<sup>74</sup><https://github.com/jbagnato/machine-learning>