Trabajo de Laboratorio N°2:

Regresión Lineal Multivariable y

Regularización De Ridge

- **Materia:** Inteligencia Artificial.
- **Carrera:** Ingeniería en Informática.
- **Docente:** Ing. Federico Gabriel D´Angiolo.
- **Estudiante:** Calonge, Federico.

Índice.-

1- Objetivo.

Este Trabajo de Laboratorio consistirá en estudiar los distintos algoritmos de Regresión, obteniendo los resultados que se adquieren de cada uno de ellos.

Para esto analizaremos 2 Data Sets y demostraremos el funcionamiento de distintas regresiones mediante ejemplos realizados en Python con el ambiente Anaconda. Los algoritmos de regresión que analizaremos serán:

- **Regresión Lineal Simple (RLS).**
- **Regresión Lineal Multivariable (RLM):** realizando estimaciones a nuestro modelo mediante el **método de mínimos cuadrados** y mediante la **Regularización de Ridge**.

Explicaremos qué significa usar cada una de estas técnicas, y a partir de estas, obtendremos modelos y visualizaciones de los datos que nos permitan predecir el comportamiento de distintas variables.

2- Introducción: Conceptos previos.

En esta Sección describiremos los temas teóricos y matemáticos para tratarlos a lo largo del desarrollo del Informe y llevarlos a cabo mediante los Algoritmos de Python en el ambiente de Anaconda.

2.1-Variable target vs Variables Feature.

En un Data Set nos encontraremos con distintas variables:

- Nuestras variables "Feature" serán aquellas que describan las características de los datos que analizaremos, serán nuestras variables predictoras. Son nuestras **X**s.
- Nuestra variable "Target" es una sola, es la variable a predecir, el "objetivo" de aplicar alguna técnica de Machine Learning es su predicción. Se predicen mediante las variables "Feature", por esto decimos que la variable "Target" depende de las variables "Feature". Es nuestra **Y**.

2.2-Regresión.

Como vimos anteriormente en el TL1, **Regresión** es una técnica que utilizamos para **predecir respuestas / valores continuos** (a diferencia de las técnicas de **Clasificación,** donde tratamos de **predecir respuestas discretas**, como por ej. clasificar si un mail es Spam o no lo es). De esta manera, con Regresión podemos resolver problemas, como por ejemplo, predecir y observar cambios en la temperatura censada en un ambiente o variaciones en el consumo eléctrico con respecto al tiempo (ya que la temperatura o el consumo eléctrico serían nuestras variables a predecir, nuestros "target" y son **variables continuas**).

Dicho de otra manera, la Regresiòn nace de la necesidad de ajustar alguna función a un conjunto de datos.

Hay muchas técnicas de Regresión, entre ellas están:

- Modelo lineal generalizado (GLM):
  - Regresión Lineal Simple (RLS).
  - Regresión Lineal Múltiple (RLM).
- Regresión NO lineal.
- Ensemble methods.
- Técnicas de Regularización.
- Regresión por pasos.
- Árboles de decisión.
- Redes neuronales (ANN).
- Aprendizaje adaptativo neuro-difuso.
- Support Vector Regression (SVR).
- Gaussian Process Regression (GPR).

Las técnicas de Regresión (y también las de Clasificación) pertenecen a la rama de **Aprendizaje Supervisado** en **Machine Learning** (ML). Mediante el Aprendizaje Supervisado podemos construir modelos de ML basándonos en **datos de entrenamiento etiquetados.** De esta manera, podemos construir un sistema para predecir automáticamente el ingreso mensual de una persona, en función de varios parámetros como la edad, educación, ubicación, etc.

2.2.1-Regresiòn Lineal (RL): Simple (RLS) y Multivariable (RLM).

Como mencionamos anteriormente, la **Regresión Lineal** (RL) es una técnica de aprendizaje supervisado y una de las más usadas en ML. Su fortaleza es su simplicidad e interoperabilidad. Es una técnica “paramétrica” de ML, ya que antes de mirar los datos ya sabemos cuántos parámetros (o coeficientes) vamos a necesitar.

La regresión lineal es un procedimiento estadístico que busca establecer una relación directa o inversa entre dos o más variables. Permite realizar una **predicción del comportamiento** de alguna variable en un determinado punto o momento.

En la ***Imagen 1*** se muestra un ejemplo de RL. Allí observamos que tenemos un conjunto de datos (puntos), y una recta (el modelo lineal que mejor se ajusta al conjunto de datos dado).

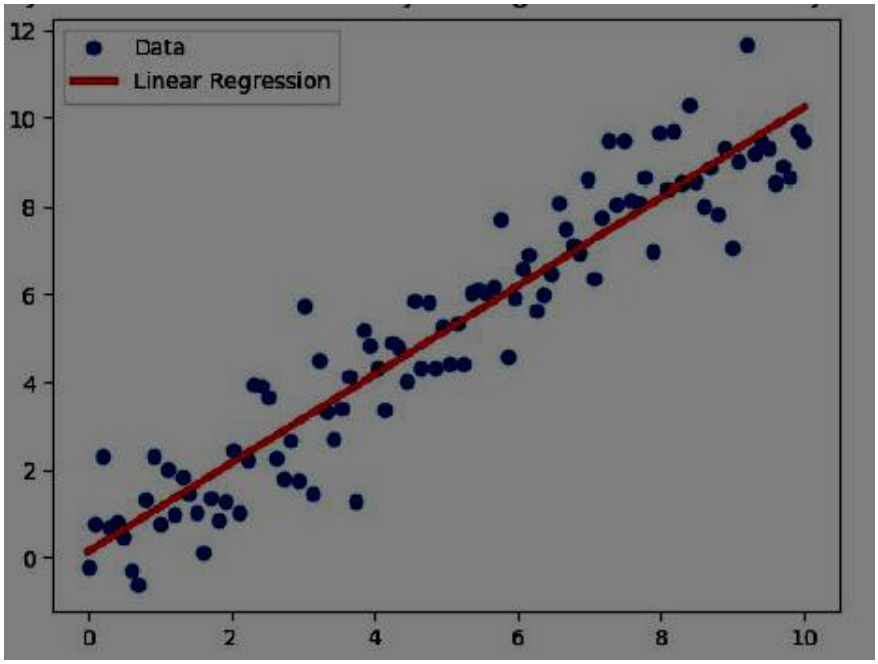


Imagen 1

De esta manera, aplicamos RL para obtener una línea recta estimativa que mejor se ajuste a los datos que tenemos (a nuestra/s variable/s independiente/s "x"). Antes de aplicar este modelo debemos comprobar que nuestro conjunto de datos tiene una relación lineal; y luego así podremos aplicar esta técnica para poder predecir el comportamiento de nuestra Y. El modelo de esta línea recta, la cual nos permitirá predecir valores de y, es el descripto en la ***Fórmula 1***.

$$f(x_i) = \alpha_0 + \sum_{i=1}^{m} \alpha_i x_i \quad donde\ A = \{\alpha_0, \alpha_1, \ldots, \alpha_m\}$$

Fórmula 1

-Donde:

- Y = $f(x_i)$ = vector de salida. Nuestra variable dependiente "y", o también llamada "target", ya que es la que queremos predecir.
- X = vector de entrada. Nuestra/s variable/s independiente/s "x", o también llamadas "feature" (datos de entrada que muestran distintas características de lo que queremos analizar).
- A = vector de coeficientes (estos coeficientes son obtenidos a partir de los datos). Conformado por a_0, a_1, (….), a_m.

De esta manera, cada X representa un dataset de vectores de entrada con valores reales:

X = { $x_1, x_2, \dots x_n$ } donde x_1 e R

Y nuestra **Y** representa cada valor real asociado a estos vectores de entrada:

Y = { $y_1, y_2, \dots y_n$} donde y_n e R

El caso de **Regresión Lineal Simple**, se da cuando existe un único vector de entrada y un vector de salida. Esto es cuando utilizamos solo una variable X, entonces nuestra linea recta necesitará únicamente 2 coeficientes (ao y a1). De esta manera, la expresión se reduce a la ***Fórmula 2***.

$$y = f(x) = a_0 + a_1 . x$$

Fórmula 2

-Donde:

- Y = $f(x)$ = vector de salida. Lo que queremos predecir.
- a_0 = nuestra ordenada al origen (solo para el caso de RLS).
- a_1 = la pendiente de nuestra recta.
- a_0, a_1 = coeficientes obtenidos a partir de los datos.

En Regresión Lineal Simple nos centramos en predecir una variable target Y en función de una variable predictora/feature X. En cambio, en **Regresión Lineal Multivariable** nos centramos en predecir una variable target Y, pero ahora, **en función de 2 o más variables predictoras/feature X**. Cada una de estas X tendrá sus respectivos **coeficientes**, los cuales tendrán como objetivo reducir o incrementar el impacto de las variables en el resultado final.

De esta manera, en la RLM tenemos **varios vectores de entrada** y un solo vector de salida; obteniendo la expresión en la ***Fórmula 3***.

$$y = f(x) = a_0 + a_1 . x_1 + a_2 . x_2 + (\dots) + a_n . n$$

Fórmula 3

-Donde:

- n = número de variables de entrada.
- Xn = vector de entrada n.

El aprendizaje consiste en encontrar cuáles son los mejores parámetros / **coeficientes** para los datos que tengamos. De esta manera, para obtener las constantes del vector A (osea, los **coeficientes** a_0, a_1, (....), a_m) podemos utilizar distintos métodos:

- **Método de mínimos cuadrados**: Es el que utilizaremos para predecir la mayoría de veces en nuestros modelos de datos. Detallado en la sección ***2.3-Método de los mínimos cuadrados, RSS, MSE, RMSE*.**
- **Descenso del gradiente:** consiste en calcular las derivadas parciales de todas las variables que intervienen en el modelo de datos y que son necesarias para poder predecir el vector "Y" del sistema, en otras palabras consiste en calcular la derivada de la función en diferentes puntos con el objetivo de calcular la pendiente de dicho parámetro, en donde mi derivada en ese punto valga 0 (cero), y de esta manera poder obtener un mínimo local de la función. En Sistemas bidimensionales, pueden existir varios puntos locales mínimos, y lo que se buscará es obtener derivadas parciales en distintos puntos que conformará un vector que indicará la dirección en la que la pendiente asciende, este vector resultante se lo denomina Gradiente, cómo queremos obtener el mínimo, tomaremos el sentido opuesto de este vector.
- **Regularizaciones**: por ejemplo **Regularización de Ridge**. Detallado en la sección ***2.7-Regularización de Ridge.***

2.3-Método de los mínimos cuadrados, RSS, MSE, RMSE.

El **método de mínimos cuadrados** es uno de los criterios de evaluación más utilizados en problemas de Regresión para obtener los **mejores parámetros / coeficientes** para los datos que tengamos (osea las constantes del vector A: los **coeficientes** a_0, a_1, (….), a_m) y así obtener nuestra recta lineal que mejor se ajuste a los datos.

. Los "mejores" coeficientes serán los que **minimicen en alguna medida el error**. Lo que buscamos es elegir los coeficientes que **minimicen la suma de los cuadrados de los residuos/errores (RSS o Residual Sum of Squares).** Llamando residuo/error a la diferencia entre el valor de la predicción y el valor real en un punto. RSS lo calculamos como la sumatoria de los cuadrados de los residuos/errores (***Fórmula 4***).

$$RSS = \sum (\hat{Y}_i - Y_i)^2$$

Fórmula 4

De esta manera, podemos obtener Los coeficientes a0 y a1 (sea una RLS o una RLM -la única diferencia que en RLM tendremos más coeficientes como resultado-) a partir de las ***Fórmulas 5 y 6.***

$$\alpha_0 = \frac{\sum_{i=1}^{N} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sum_{i=1}^{N} (x_i - \bar{x})^2}$$

Fórmula 5.

$$\alpha_1 = y - \alpha_1 . \bar{x}$$

Fórmula 6.

Donde:

- xi = valor real en el punto
- 'xi = valor de la predicción.

De esta manera, lo que buscamos al utilizar el método de mínimos cuadrados es obtener un RSS mínimo, para tener el menor error posible en nuestras predicciones.

El **MSE** (Mean Squared Error – **Error cuadrático medio**) *es el* promedio de la sumatoria de los cuadrados de los residuos/errores, se puede expresar como ***MSE = RSS / n*** (donde n es el número de variables de entrada), o sino como en la ***(Fórmula 7).***

$$MSE = \frac{1}{n}\sum(\hat{Y}_i - Y_i)^2$$

Fórmula 7.

El **RMSE** (Root mean square error - **Distancia media cuadrática mínima**) es la raíz cuadrada del error cuadrático medio (***Fórmula 8***).

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \cdot RSS}$$

Fórmula 8.

2.3.1-¿Qué usar: MSE o RMSE?

El **MSE** no es muy intuitivo ya que nos da el error medio al cuadrado. Así que, si nuestro modelo para predecir, por ejemplo, el valor de bienes inmuebles tiene un error cuadrático medio de 1.000.000$ nos daría la impresión de que tiene mucho error. En realidad, el error sería más / menos 1.000 $ en media, porque la raíz cuadrada de 1.000.000 es 1.000 → RMSE. Así, vemos que **RMSE es más intuitivo que MSE**, pero… ¿por qué no usamos el RMSE todo el tiempo envés de MSE?.

Hay 2 razones:

1. Es más costoso computacionalmente. Muchos algoritmos supervisados necesitan calcular el error en cada iteración para aprender de sus errores; así que cuanto más rápido mejor.
2. La forma del **MSE** hace que cierto tipos de algoritmos de optimización (tales como el gradiente descendiente), encuentren la mejor solución más rápido.

Por estas razones, como conclusión podemos decir que normalmente se suele usar el **MSE durante el proceso de aprendizaje** y su raíz cuadrada (el **RMSE**) al final, **para dar una estimación** en términos intuitivos **de la calidad de la predicción**.

2.4-Coeficiente de correlación/Pearson (ρ) VS Coeficiente de determinación (R^2-"Score").

Coeficiente de Correlación / Person (ρ): Es la división entre la covarianza de X e Y, y el producto de los desvíos estándar de X e Y. El resultado de esta operación es una cantidad sin dimensiones: el coeficiente de Person. Matemáticamente se calcula mediante la ***Fórmula 9***.

$$\rho = \frac{cov(x,y)}{\sigma x \, . \, \sigma y}$$

Fórmula 9

El coeficiente **ρ** se encuentra contenido en el intervalo -1< **ρ** <1. El caso de **ρ**= 0 indica la ausencia de cualquier asociación lineal, mientras que los valores -1 y 1 indican relaciones lineales perfectas. En forma más sencilla: **este coeficiente nos dice qué tan lineal es la relación entre 2 variables (X e Y)**. Estamos NORMALIZANDO la covarianza. De esta manera **ρ** la ponemos entre 1 y -1 y así... si **ρ** vale 1 entonces X e Y son lineales y de pendiente positiva. Si **ρ** vale -1 son lineales y de pendiente negativa. Y si **ρ** vale 0,1 entonces NO tendrían una relación lineal, ya que NO podríamos encontrar una recta que relacione ambas variables (esto suponemos generalmente que sucede cuando **ρ** está aproximadamente en el rango: -0,7 < **ρ** < 0,7).

Coeficiente de determinación (R^2-"Score"): Este coeficiente de determinación, en cambio tiene un objetivo ligeramente distinto al **ρ**. **Permite medir/determinar la calidad del modelo para replicar los resultados**, y la proporción de variación de los resultados que puede explicarse por el modelo. Matemáticamente la calculamos como en la ***Fórmula 10*** (es una división entre las varianzas de los datos estimados por el modelo y las varianzas de los datos observados). O de forma análoga como la ***Fórmula 11*** (1 – el cociente de la varianza de los residuos –o la suma de los cuadrados totales- divido la varianza de los datos observados –o la suma de los cuadrados de los residuos-).

$$R^2 = \frac{s_{\hat{y}}}{s_y}$$

Fórmula 10

$$R^2 = 1 - \frac{\sum(y_i - p_i)^2}{\sum(y_i - \bar{y})^2}$$

Fórmula 11

-Donde:

- y_i son los valores que toma la variable Y.
- p_i son los valores de la predicción.
- $\bar{y}$ es el valor medio de los valores que toma la variable Y.

Como vemos en la ***Fórmula 10***, que el R2 representa la proporción de varianza replicada por nuestro modelo, así, cuanto más cerca estemos de 1, más varianza de la variable dependiente conseguiremos explicar con nuestro modelo; y en teoría será un modelo mejor. De esta manera al igual que sucede con el Coeficiente de Pearson buscamos que sea un valor cercano a 1. Entonces:

- SI R^2 nos dà un valor cercano a 1, significa que las predicciones que realizaremos con nuestro modelo seguramente sea acertadas.
- Si R^2 nos dà un valor cercano a 0 màs que a 1 nuestras predicciones no serán acertadas.
- Si R^2 nos dá negativo, es que la predicción puede ser arbitratiamente mala.
- Si R^2 =0, es que la predicción coincide con la esperanza de los valores de la variable a predecir Y.

Como conclusión, podemos decir que **con R^2 medimos la efectividad de los algoritmos (en nuestro caso utilizamos R^2 para medir la efectividad de RLS, RLM y Ridge).**

Cómo implementarlos en código y cuándo utilizamos cada uno:

Scikit-Learn implementa el cálculo de **R^2** mediante la función "r2_score". Lo utilizamos a lo último para evaluar la calidad de nuestro modelo para realizar predicciones.

Y para fijarnos el cálculo del factor **ρ** utilizamos la función "corr" (y podremos realizar una matriz de correlación comparando todas las variables con nuestra variable Y, para ver así cual tiene una relación lineal con esta). Lo utilizamos ANTES de realizar RL para así analizar qué variables nos pueden dar mejores resultados de predicción luego de realizar el modelo.

2.5-Sesgo y Varianza.

- **Sesgo** (o **BIAS** en inglés)**:** Permite saber que tan "leal" es la recta trazada respecto a nuestros datos. Un sesgo alto significa que la recta de nuestro modelo lineal NO se aproxima a nuestros datos, lo que genera un **ALTO MSE**; mientras que un sesgo bajo significa que la recta SI se aproxima a nuestros datos y en consecuencia, tendría un **BAJO MSE**.

- **Varianza**: nos permite saber qué tanto varía el MSE al cambiar los datos de nuestro data set.

2.6-Multicolinealidad y Overfitting.

- **Overfitting**: que nuestro modelo tenga "sobreentrenamiento" significa que tiene una performance muy buena con los datos con los cuales lo construimos (set de entrenamiento), pero falla a gran escala con datos nuevos (set de prueba).
- **Multicolinealidad**: indica una fuerte relación lineal entre 2 o más **variables independientes** (nuestras "x", nuestras variables predictoras) con las cuales construimos el modelo de Regresión Lineal Multivariable. La multicolinealidad generalmente puede producir que los coeficientes estimados de la RLM sean inexactos.
  Para tener en cuenta: la multicolinealidad mide la relación lineal que existe (o no) sólo entre nuestras variables X, **NO entre nuestras variables X y nuestra variable a predecir Y**.
- Una manera de **prevenir el overfitting y la multicolinealidad** es **usando el método de Regularización de Ridge.**

2.7-Regularización de Ridge.

Una Regularización consiste en **añadir una penalización a la función de coste** (esta función de coste es la que buscamos minimizar mediante distintas técnicas de aprendizaje automático; tratando de encontrar los coeficientes que minimicen dicha función). El objetivo de regularizar **es añadir un término que penalice la complejidad del modelo**. Cuando usamos regularización minimizamos la complejidad del modelo a la vez que minimizamos la función de coste.

Esto resulta en modelos más simples que tienden a generalizar mejor. Lo modelos que son excesivamente complejos tienden a caer en **Overfitting**. Es decir, a encontrar una solución que funciona muy bien para los datos de entrenamiento pero muy mal para datos nuevos. Nos interesan los modelos que además de aprender bien, también funcionen tengan un buen rendimiento con datos nuevos.

Hay muchas Regularizaciones, **nosotros utilizaremos la Regularización de Ridge (o también llamada L2).**

La **Regularización de Ridge se utiliza para resolver problemas que ocurren en algunas oportunidades en la RLM: la multicolinealidad y el overfitting.** Cuando se produce esto, las estimaciones mediante mínimos cuadrados tienden a tener un sesgo bajo y variaciones grandes, por lo que nuestro modelo no será muy fiable para predecir. **La Regularización de Ridge añade un sesgo, y de esta manera, permite mejorar nuestro modelo reduciendo los errores.** En conclusión: si tenemos problemas de multicolinealidad u overfitting en nuestras variables predictoras x (con las cuales queremos construir un modelo de RLM), entonces debemos usar la Regularización de Ridge envés de utilizar las estimaciones mediante el método de mínimos cuadrados; de esta manera construiremos un modelo con predicciones más confiables, menor overfitting y eliminando en mayor medida el problema de la multicolinealidad entre variables.

De esta manera, utilizando Regularización de Bridge envés del método de mínimos cuadrados en RLM, ahora nuestros **coeficientes w** (anteriormente llamados a – ver la ***Fórmula 3*** de RLM), se calcularán como en la ***Fórmula 12***.

$$w = (X^T X + \alpha I) X^T y$$

Fórmula 12

-Donde:

- W = nuestros coeficientes.
- X^T = matriz traspuesta de X.
- X = nuestra matriz de entradas.
- ∂ = parámetro alfa de regularización.
- I = matriz de identidad.
- y = nuestro vector de salidas.

Y nuestra variable a predecir será la misma que antes (**Fórmula 13**):

$$Y_{pred} = W_0 + W_1 X_1 + W_2 X_2 + ... W_n X_n$$

Fórmula 13

La única diferencia con RLM es que ahora el funcionamiento será distinto, ya que Ridge le quitará peso a los coeficientes menos importantes, haciendo que estos se tiendan a 0, con el propósito de reducir aún más el valor de la función de costo. El α que encontramos en la ecuación de obtención de nuestros parámetros (también es llamado "parámetro de penalización") es el que le quita peso a los coeficientes. De esta manera, **a mayor α mayor penalización y los coeficientes serán menores**. Notaremos que si $\alpha = 0$, la Regularización de Ridge se comportaría como una RLM utilizando la estimación por el método general de mínimos cuadrados. Por esto, elegir un buen valor de α es crítico. Podemos utilizar valores al azar que tiendan a 0 pero mejor es utilizar la **técnica de "cross validation" (validación cruzada)** para así tener el α que producirá las mejores predicciones. Este método consiste en:

1-Tomamos una grilla de valores de α y calculamos el error de validación cruzada para cada α.

2-Elegimos el valor de α para el cual el error de validación cruzada es menor.

3-Finalmente ajustamos el modelo utilizando todas las covariables y el valor elegido de α.

En nuestro ejemplo (***3.2-Implementación Ejemplo 2 – Propiedades en Boston: RLM y Regularización de Ridge.***) utilizamos un α de 0.5. Lo correcto sería utilizar cross validation para encontrar el α óptimo.

Ahora observaremos las **diferencias en las ecuaciones matemáticas para obtener nuestro RSS mínimo (nuestra función de costo):**

- Utilizando RLM con el método de mínimos cuadrados: ***Fórmula 14.***

$$Cost\ (W) = RSS(W) = \sum_{i=1}^{N} \{y_i\ - \hat{y}_i\}^2\ = \sum_{i=1}^{N} \left\{ y_i\ - \sum_{j=0}^{M} w_j\ x_{ij} \right\}^2$$

Fórmula 14

- Utilizando Regularización de Ridge: se le añade una penalización (que es la suma de los cuadrados de la magnitud de los pesos w). Quedándonos la función de coste descripta en la ***Fórmula 15.***

$$Cost\ (W) = RSS(W) + \ \lambda * (sum\ of\ squares\ of\ weights)$$

$$= \sum_{i=1}^{N} \left\{ y_i\ - \sum_{j=0}^{M} w_j\ x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} w_j^2$$

Fórmula 15

3- Desarrollo.

3.1-Implementeación Ejemplo N°1 – Artículos de Machine Learning: RLS y RLM.

3.1.1-Data set.

En este ejemplo, el Data Set que cargamos será de un archivo csv de entrada obtenido mediante Web Scraping (técnica utilizada mediante programas de software para extraer información de sitios web). Este archivo contiene diversas URLs a artículos que hablan sobre Machine Learning de distintos sitios web.

Como características / features de entrada (nuestras columnas) tendremos:

- **Title:** Titulo 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».

Nota: este archivo .csv contiene mitad de datos reales y otra mitad de datos generados aleatoriamente, por lo que las predicciones que obtendremos no serán reales.

Ejemplo de los primeros 10 registros con sus 8 features:

	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
5	5 ways Data Science and Machine Learning impac...	NaN	761	0	NaN	1	14	21000
6	200 universities just launched 600 free online...	https://qz.com/1120344/200-universities-just-l...	6462	600	28.0	2	170	200000
7	How Machine Learning can help Cryptocurrency T...	https://cryptovest.com/news/how-machine-learni...	753	3	0.0	1	78	77000
8	Tech companies should stop pretending AI won't...	https://www.technologyreview.com/s/610298/tech...	1118	2	NaN	1	62	59400
9	Artificial intelligence is going to completely...	NaN	1581	4	NaN	2	60	35000

3.1.2-¿Qué queremos predecir?

A partir de nuestras features intentaremos predecir cuantas veces será compartido un artículo en Redes Sociales (osea que nuestra variable a predecir "y" será la variable "# Shares").

Implementaremos como inicio una RLS (con 1 sola variable predictora "X") y luego pasaremos a realizar un ejemplo de RLM donde utilizaremos 3 dimensiones (X,Y,Z) para observar gráficamente el plano de nuestro modelo lineal y principalmente, para poder realizar predicciones sobre el # Shares.

3.1.3-Implementación del modelo de RLS.

Anteriormente a implementar el modelo de RLS analizaremos nuestros datos estadísticamente:

	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

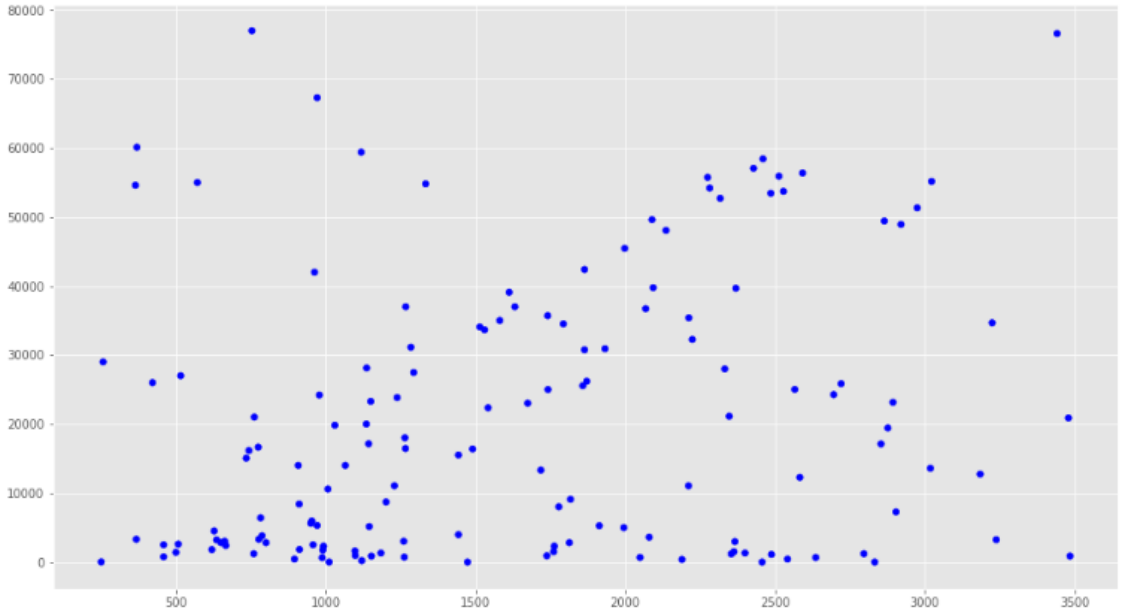
Observamos que la media de palabras en los artículos es de 1808. El artículo más corto tiene 250 palabras y el más extenso 8401.

Luego, eliminamos 3 features que no nos parecen de tanta importancia para predecir el #shares (igualmente lo recomendable es analizar las relación entre las variables y ver nuestra variable "y" en función de qué otra variable "sigue" una relación lineal mediante histogramas –o mediante matrices de correlación utilizando el coeficiente de Pearson-, y eliminar las que no siguen este tipo de relación lineal). Pero, a modo de ejemplo, simplemente eliminamos estas 3 para ver qué es lo que sucede con nuestro modelo. Ahora si, mediante la matriz de correlación observamos los coeficientes de Pearson para ver si es factible utilizar RL con estas variables:

	Word count	# of Links	# of comments	# Images video	# Shares
Word count	1.000000	0.346730	0.108730	0.463767	0.141736
# of Links	0.346730	1.000000	0.116415	0.021087	0.288325
# of comments	0.108730	0.116415	1.000000	0.018377	0.437413
# Images video	0.463767	0.021087	0.018377	1.000000	0.062170
# Shares	0.141736	0.288325	0.437413	0.062170	1.000000

No visualizamos ninguna correlación muy lineal entre nuestras variables e Y (#Shares). A modo de ejemplo **seleccionaremos como variable predictora a "Word Count"** (será nuestra X). Y trataremos de hacer predecir nuestro #Shares en base a esta X. Esto lo haremos únicamente a modo de ejemplo, si quisiéramos obtener un mejor modelo lineal hubiésemos elegido el # of comments como x, que tiene un coeficiente de Pearson de 0,43 con la variable a predecir.

Graficamos # Shares (Y) en función de Word Count (X) para corroborar gráficamente si existe una relación lineal entre ambas variables y así ver si es recomendable aplicar RL:



Comprobamos que, como vimos anteriormente en la matriz de Pearson, no hay una relación lineal. Pero como dijimos, seguiremos solo para ver qué ocurre con nuestro modelo de RLS.

Entonces, lo primero que hacemos es preparar la data para usar RLS. Para esto usamos a 'Word count' como X y '# Shares" como Y:

```
X =DF_Filtrado['Word count']
Y = DF_Filtrado['# Shares']
```

Ahora partimos la data en sets de entrenamiento y de test. Entrenaremos nuestro modelo con un 70% de ejemplos; y luego lo testearemos con el resto 20%. **Esto lo hacemos para evaluar el rendimiento del modelo en datos no vistos**. Para hacer esta partición usamos la función de scikit-learn "train_test_split":

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3, random_state=5)
```

Ahora entrenamos a nuestro modelo con el set de entrenamiento (mediante la función LinearRegression de scickit-learn):

```
RLS = linear_model.LinearRegression() #Defino el algoritmo a utilizar
RLS.fit(X_train, Y_train) # Entrenamos nuestro modelo / ajustamos el algoritmo.
```

Ahora evaluamos nuestro modelo... hacemos predicciones en base a nuestros set de PRUEBAS (TESTs). Dándonos los siguientes resultados de performance:

```
Coeficiente: 4.09
Termino independiente: 15921.26

Precision del modelo:  0.03
Score: -0.04
MSE:  348715217.39
```

Vemos que nos da un Score MUY malo (hasta negativo!, debería ser cercano a 1), y además tenemos un MSE eneorme. Esto era esperable, ya que como dijimos anteriormente, elegimos una variable que tenía un bajo coeficiente de Pearson respecto a nuestra Y.

3.1.4-Implementación del modelo de RLM.

Vamos a intentar mejorar el Modelo de RL que hicimos anteriormente, añadiendo una variable de entrada / dimensión más al modelo.

Esto le dará mayor poder al algoritmo de ML, ya que de esta manera podremos obtener predicciones más complejas basadas en más de 1 variable predictora / feature.

Ahora nuestra ecuación de la "recta" DEJA de ser una recta y pasa a ser un PLANO. En nuestro caso utilizaremos 2 variables predictivas para poder graficar un plano 3D (pero recordemos que podemos realizar mejores predicciones utilizando más de 2 entradas y prescindir del gráfico... ya que más de 3D no podemos graficar).

Nuestra 1ra variable de entrada seguirá siendo la cantidad de palabras ("Word COunt"). Y nuestra 2da variable serà la SUMA de 3 columnas / features de entrada:

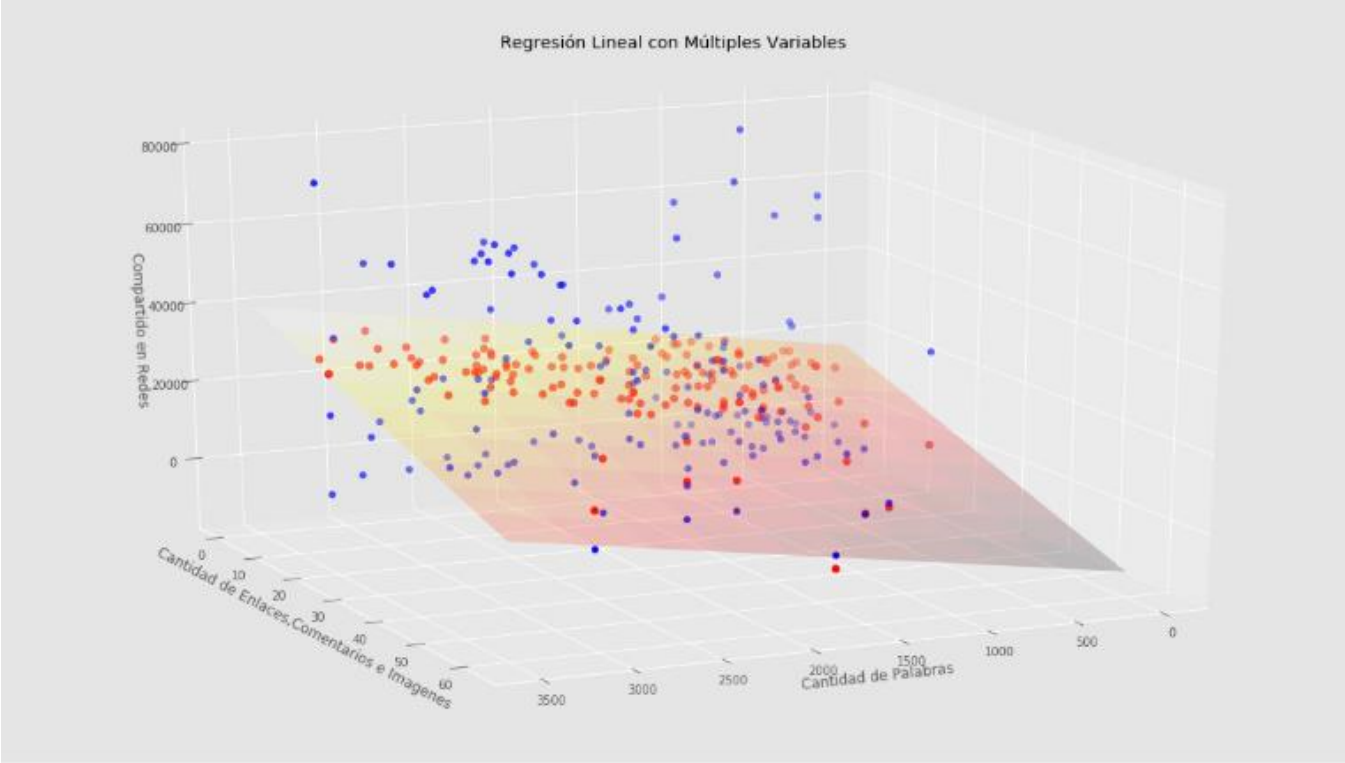
```
suma = (DF_Filtrado["# of Links"] + DF_Filtrado['# of comments'].fillna(0) + DF_Filtrado['# Images video'])

dataX2 =  pd.DataFrame()
#Ahora dataX2 serà un DF de 2 posiciones, dividimos en 2 para agregar las columnas a dataX2:
dataX2["Word count"] = DF_Filtrado["Word count"]
dataX2["suma"] = suma
```

Luego, como antes, separamos en train y test, y creamos nuestro modelo de regresión lineal múltiple con 2 variables predictoras, dándonos la siguiente información de performance:

```
Coeficientes b: [   6.63216324 -483.40753769]
MSE: 352122816.48
Score: 0.11
```

Como vemos, obtenemos 2 coeficientes (cada uno correspondiente a nuestras 2 variables predictivas), pues ahora lo que graficamos no será una línea si no, un plano en 3 Dimensiones. Además, nuestro MSE se mantiene muy alto, pero, nuestro Score mejoró un poco (aunque sigue siendo muy malo, muy lejos del 1). Por esto, nuestro modelo tampoco será fiable para realizar predicciones.

Por último realizamos un plot para obtener una representación gráfica de nuestro modelo lineal (que como tenemos 2 variables predictoras y 1 variable a predecir podemos generar nuestro plano en 3 dimensiones):



En azul están los puntos de entrenamiento y en rojo los puntos con los que llegamos mediante la predicción. Y comparando ambos puntos podríamos obtener los errores. Pero como vimos anteriormente, tenemos un modelo con bajo score, por lo que nuestras predicciones no serán fiables.

3.1.5-Resultados.

Como vimos, aplicar RLM envés de RLS en nuestro caso mejoró el modelo.

De esta manera, vimos cómo crear con SKLEARN en Python modelos de RL con 1 o múltiples variables. En este ejemplo NO tuvimos una gran confianza en las predicciones... por ejemplo en nuestro primer modelo de RLS, 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. Y para nuestro 2do modelo (RLM), donde agregamos otra dimensión más, mejoramos un poco el error y el score pero igualmente no fue muy alto, con lo que nuestro modelo no tendrá mucha confianza en sus predicciones (pero si más que la RLS que realizamos en el 1er modelo).

RLS y RLM no nos sirvieron para nuestros datos (aunque hubiésemos elegido otras variables, ya que como vimos en la matriz de correlación, el mayor coeficiente de Pearson que teníamos ni siquiera llegaba a 0,5). Pero, hay muchísimos Data Sets donde si tenemos datos con una correlación lineal con respecto a nuestra variable a predecir y, en ese caso, nuestra RL llegaría a predicciones acertadas.

3.2-Implementación Ejemplo 2 – Propiedades en Boston: RLM y Regularización de Ridge.

3.2.1-Data set.

En este ejemplo nuestro Data Set lo cargaremos directamente desde la librería de Python SKLearn. Es un DS de 506 filas x 13 columnas/variables features que contienen información de propiedades de Boston (más específicamente Casas).

Como características / feature de entrada (nuestras columnas) tendremos:

- CRIM: tasa de crimen per capita por ciudad.
- ZN: proporción de zona residencial dividida en zonas para lotes de más de 25,000 pies cuadrados.
- INDUS: proporción de negocios NO minoristas por ciudad.
- CHAS: variable ficticia (dummy) sobre el Rio "Charles". Si es =1 el tramo limita con el rio, si es =0 caso contrario.
- NOX: concentración de óxidos nítricos (medida por 10 millones)
- RM: número promedio de habitaciones por vivienda
- AGE: proporción de unidades ocupadas por sus propietarios construidas antes de 1940
- DIS: distancias ponderadas a cinco centros de empleo de Boston
- RAD: indice de accesibilidad a las carreteras radiales
- TAX: tasa de impuesto a la propiedad de valor total de $10.000.
- PTRATIO: relación alumno-profesor por localidad.
- B: 1000(Bk - 0.63)^2 → donde Bk es la proporción de población negra en la ciudad.
- LSTAT: menor STATUS de la población.

Y además, tendremos una variable "MEDV", la cual mide el precio de las casas. Más específicamente es el valor medio de las viviendas ocupadas por sus propietarios en $1000's (miles). Por ejemplo, si MEDV = 2 significa que el valor medio es de 2.000.Esta última variable (la 14) – MEDV será nuesta variable a predecir / nuestra variable "target".

Ejemplo de los 5 primeros registros con sus features:

```
In [182]: DFBoston=pd.DataFrame(data=VectorDataBoston, columns=VectorHeadersBoston) #Definimos el DF.
          DFBoston.head()
```

Out[182]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

3.2.2-¿Qué queremos predecir?

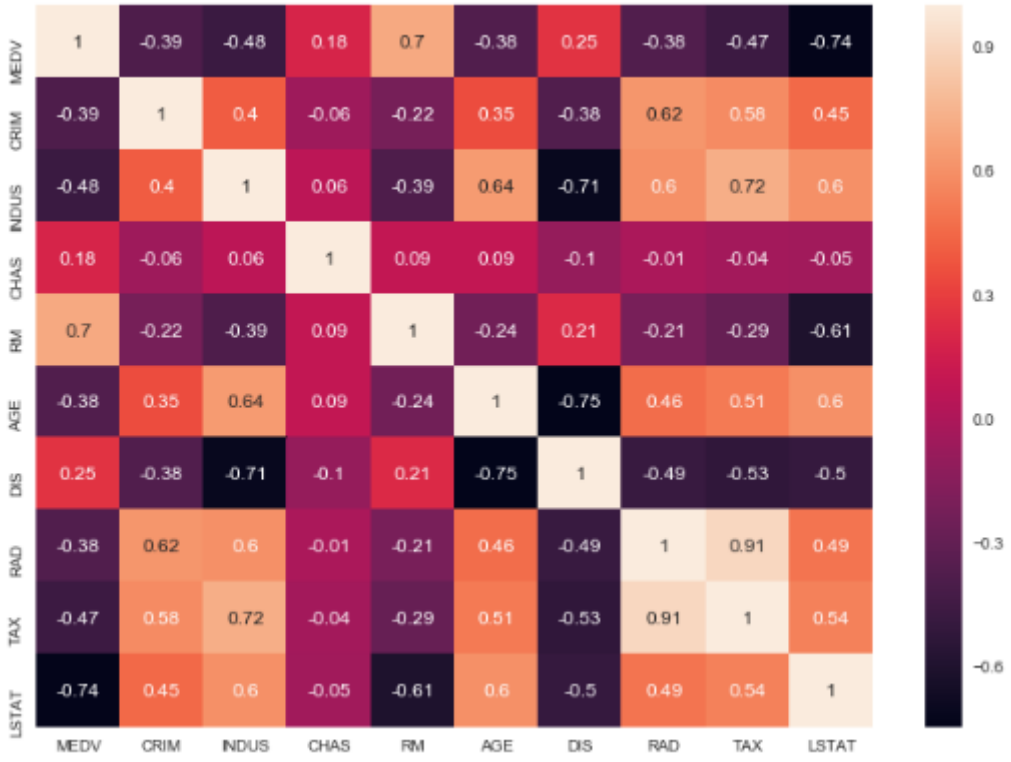
A partir de algunas de nuestras variables features intentaremos predecir el valor de los precios de las casas osea que nuestra variable a predecir “y” serà la variable MEDV) utilizando RLM con 3 variables y analizaremos los resultados: ver si se llegó o no a una buena predicción.

Aplicaremos RLM y Regresión de Ridge para evaluar nuestro modelo y obtener distintos rendimientos a modo de ejemplo.

3.2.3-Implementación RLM.

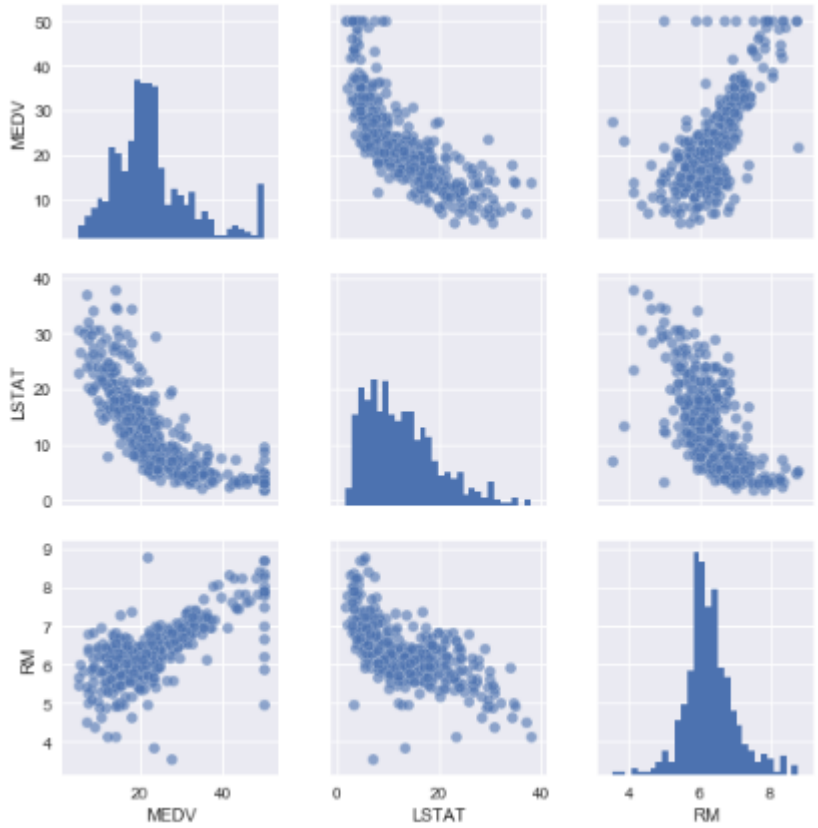
Antes de realizar cualquier algoritmo, primero analizamos nuestros datos. Por esto lo que hacemos es analizar la matriz de correlación para ver el coeficiente de Pearson / correlación entre las variables, y de esta manera ver si es posible utilizar un algoritmo de RL para las predicciones.

Obteniendo así:



Observamos en la columna (o fila) MEDV (nuestra variable a predecir) los valores para las relaciones con las demás features. Y vemos que tiene una correlación lineal positiva "alta" con RM (de 0,7) y una correlación lineal negativa "alta" con LSTAT (de -0,74).

Verificamos este factor de correlación viendo los histogramas para las relaciones entre MEDV, LSTAT y RM y verificamos que la matriz dé valores correctos y los gráficos se parescan a "algo lineal"):



Y observamos que, como nos marcó la matriz de correlación, hay una relación levemente lineal POSITIVA entre RM y MEDV.Y también vemos que hay una relación levemente lineal NEGATIVA entre MEDV y LSTAT. Podemos ver que los valores de RM incrementan linealmente (a mayor RM, mayor es MEDV); y que los precios tienden a decrementar cuando LSTAT incrementa.

Ahora sí, realizaremos un análisis predictivo mediante un modelo de Regresión Lineal. Como vimos anteriormente **utilizaremos las variables / features RM y LSTAT** (que tienen una relación "lineal" / mayor correlación con MEDV, nuestra variable target). Ya que para entrenar nuestro modelo de RL necesitamos feautures que tengan alta correlación con MEDV.

Para seleccionar las variables feature para un modelo de RLM debemos tener en cuenta también la MULTICOLINEALIDAD. La multicolinealidad hace referencia a la correlación lineal entre mis variables predictoras (en este caso LSTAT y RM). Si existe una relación lineal fuerte / un factor de correlación grande entre nuestras variables predictoras, nuestro modelo seguramente predecirá mal.

Pero, para nuestro caso NO se cumple esto, ya que tienen un factor de correlación de -0,61 (no es muy alto). De esta manera NO nos afectará la MULTICOLINEALIDAD. Distinto hubiera sido si hubiésemos visto que MEDV tiene una correlación grande con RAD y TAX. Ya que, en este caso si, como TAX y RAD entre si están fuertemente correlacionados (factor de 0,91) nos afectaría la MULTICOLINEALIDAD. También ocurriría lo mismo para DIG y AGE (correlación de -0,75).

Primero preparamos la data para usar RLM. Para esto Concatenamos las columnas LSTAT y RM usando np.c_ (de la libreria numpy):

```
X = pd.DataFrame(np.c_[DFBoston['LSTAT'], DFBoston['RM']], columns = ['LSTAT','RM'])
Y = DFBoston['MEDV']
```

Ahora partimos la data en sets de entrenamiento y de test. Entrenaremos nuestro modelo con un 80% de ejemplos; y luego lo testearemos con el resto 20%. Esto lo hacemos **para evaluar el rendimiento del modelo en datos no vistos**. Para hacer esta partición usamos la función de scikit-learn "train_test_split":

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
```

Luego entrenamos a nuestro modelo con el set de entrenamiento obviamente (mediante la función LinearRegression de scickit-learn):

```
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

Por último evaluamos nuestro modelo... hacemos predicciones en base a nuestros set de PRUEBAS (TESTs). Dandonos los siguientes resultados:

```
Performance del modelo:
MSE:  26.393
RMSE es: 5.137
R2 score es: 0.663
```

Estos valores que nos dieron NO son los mejores, pero es un buen camino.

Los coeficientes que obtuvimos para este Modelo de RLM fueron:

```
lin_model.coef_
```

```
array([-0.71722954,  4.58938833])
```

3.2.4-Implementación Regularización de Ridge.

Como sabemos, Ridge es una técnica de regularización que nos permite mejorar la performance del modelo por ej. cuando tenemos multicolinealidad (en nuestro caso no sucede esto: LSTAT y RM NO lo son), pero igualmente lo implementamos y comparamos ambas regresiones para ver a qué valores llegamos.

Creamos un nuevo modelo que utilizará para calcular los coeficientes, envés del método de mínimos cuadrados, Ridge:

```
lin_model_ridge = Ridge(alpha=0.5)
lin_model_ridge.fit(X_train, Y_train)  #entrenamos el modelo.
```

Los coeficientes nos dieron:

```
lin_model_ridge.coef_
```

```
array([-0.71822299,  4.57171055])
```

Evaluamos la performance del modelo:

```
Performance del modelo Método de Ridge:
MSE:  26.419
RMSE es: 5.140
R2 score es: 0.663
```

3.2.5-Resultados.

Vemos que tanto los coeficientes como los score y errores en el caso de aplicar Ridge nos dieron prácticamente idénticos al modelo anterior utilizando RLM con método de mínimos cuadrados. Esto ocurrió a causa de nuestros datos, los cuales NO presentan multicolinealidad. Entonces **para este caso Ridge no fue necesario utilizarlo**, ya que no necesitamos realizar una penalización.

Otro factor que tuvo incidencia en esto es la **elección del alpha**, podríamos utilizar el método de cross validation para seleccionar un alpha óptimo y mejorar nuestro modelo. Como como dijimos anteriormente, nuestro caso no presentaba multicolinealidad (las variables que utilizamos como predictoras (LSTAT y RM) NO tenían una correlación muy lineal entre si; entonces utilizar RLM mediante el método de mínimos cuadrados o utilizar la Regularización de Ridge no afectó mucho en el score de nuestro modelo.

Distinto hubiese sido si hubiésemos utilizado otras variables que si tengan correlación lineal entre sí: por ejemplo AGE y DIS (ya que al ver Matriz de correlación que presentan un coeficiente de Pearson de 0,75). Allí seguramente, en presencia de multicolinealidad, la Regularización de Ridge hubiese dado mejores resultados que RLM.

4-Conclusiones.

Esta práctica nos permitió observar que antes de aplicar RLS o RLM tenemos que tener muy en cuenta el **coeficiente de Pearson** entre nuestras variables y tener especial atención cuando tenemos un caso de la **multicolinealidad** en nuestras variables predictoras. Si nos ocurre esto, lo mejor sería utilizar la **Regularización de Ridge**.

En nuestro 2do ejemplo aplicar el método de Ridge nos dió prácticamente el mismo resultado que aplicando método de mínimos cuadrados con RLM, esto debido a, como dijimos anteriormente, la no presencia de multicolinealidad en las variables que elegimos como predictoras.

Además, se debería probar mediante el método de "Cross Validation" (Explicado teóricamente en la sección ***2.7-Regularización de Ridge***) elegir un ALFA apropiado, ya que anteriormente elegimos un alfa al "al azar" de 0.5 para Ridge.

5-Mejoras a desarrollar.

- Se debería evaluar algún Data Set que presente problemas de multicolinealidad entre sus variables predictoras para así aplicar la Regularización de Ridge y observar si el modelo presenta un mejor score (en teoría debería darlo) que utilizando RLM con el método de mínimos cuadrados.
- También se podría utilizar otra técnica de regularización, tal como la Regularización Lasso (L2) y Regularización Elastic Net (la cual combina L2 y L1).
- Se debería utilizar cross validation para elegir el parámetro ALFA apropiado al aplicar Regularización de Ridge.

6-Bibliografía.

- Regresión Lineal:

  https://iartificial.net/regresion-lineal-con-ejemplos-en-python/

  https://iartificial.net/error-cuadratico-medio-para-regresion/

- Ejemplo artículos ML:

  https://www.aprendemachinelearning.com/regresion-lineal-en-espanol-con-python/#more-5722

- Ejemplo Propiedades Boston:

  https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155

  http://prisoft.com/data-analysis-in-python-using-the-boston-housing-dataset/