**Tu primera regresión lineal con scikit-learnClase 1 / 18**

**Lecturas recomendadas**

[Curso de Matemáticas para Data Science: Probabilidad - Platzi](https://platzi.com/cursos/ds-probabilidad/" \t "_blank)

[Curso de Estadística Descriptiva - Platzi](https://platzi.com/cursos/estadistica-descriptiva/" \t "_blank)

[Curso de Introducción a Machine Learning por MindsDB - Platzi](https://platzi.com/cursos/machine-learning/" \t "_blank)

[Curso de Análisis Exploratorio de Datos - Platzi](https://platzi.com/cursos/analisis-exploratorio-datos/" \t "_blank)

[scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/" \t "_blank)

[Index of /ml/machine-learning-databases/housing](https://archive.ics.uci.edu/ml/machine-learning-databases/housing/" \t "_blank)

[https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data](https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data" \t "_blank)

[Curso de Fundamentos de Álgebra Lineal con Python - Platzi](https://platzi.com/cursos/algebra-lineal/" \t "_blank)

[Primera\_regresión\_lineal.ipynb - Google Drive](https://drive.google.com/file/d/1wh1T5AE1AgmgDcTaiziaeKYSuovVFQPk/view?usp=sharing" \t "_blank)

[Primera\_regresión\_lineal\_Template.ipynb - Google Drive](https://drive.google.com/file/d/1NkcOQg-BTEAD0ttGDcZcAZwYpD0jXPJd/view?usp=sharing" \t "_blank)

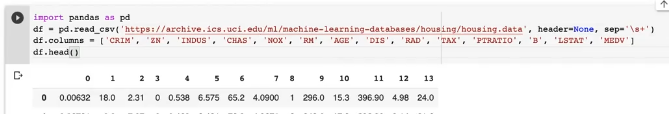
[Curso de Funciones Matemáticas para Data Science e Inteligencia Artificial - Platzi](https://platzi.com/cursos/funciones-matematicas/" \t "_blank)

[Platzi Master: aumenta tus ingresos con un mejor trabajo en tecnología.](https://platzi.com/master/" \t "_blank)

Primero los datos a trabajar

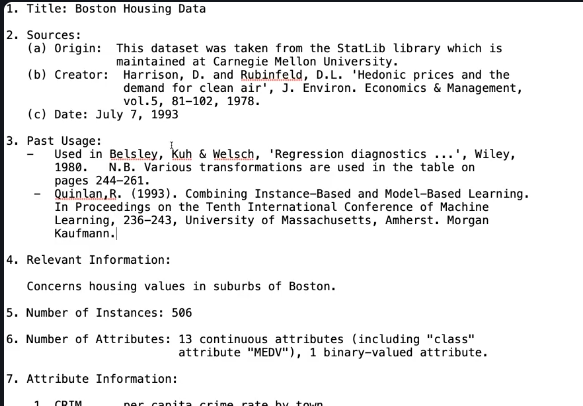
<https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data>

Luego se abre colab:



sep"\s+: existe separación por espacios

descargar archivo descriptivo



**Update 2023**

El link del dataset se cayó, entonces yo lo añadi desde local, ej:

[Descargar dataset (housing.data)](https://static.platzi.com/media/public/uploads/housing_edf82b23-51e2-4f57-887d-6be67da34d27.data)

import pandas as pd

df = pd.read\_csv(

'./housing.data',

header=None, # especificamos que el dataset no tiene headers.

sep='\s+') # especificamos que la separacion esta hecha por espacios y no por comas.

df.columns = ["CRIM","ZN","INDUS" , "CHAS" ,"NOX","RM" ,"AGE","DIS", "RAD","TAX", "PTRATIO", "B" , "LSTAT" , "MEDV"]

df.head()

Quienes hagan el curso a partir de 2023 sepan que el enlace que usar para descargar el Dataset dejo de existir, aquí les dejo el Dataset en Kaggel para que lo descarguen

[The Boston Housing Dataset | Kaggle](https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset)

[Boston House Prices (kaggle.com)](https://www.kaggle.com/datasets/vikrishnan/boston-house-prices)

Si descargan los datos y los suben a su carpeta de google Drive en Collab usen esta linea de código:

from google.colab import drivedrive.mount('/content/drive')

Luego copien la dirección donde está su Dataset en la línea de importación de datos de pandas, tal como lo muestro.

import pandas as pd

pd.read\_csv('copiar direccion aquí')

**Análisis de datos para tu primera regresión linealClase 2 / 18**

**Archivos de la clase**

[primera-regresion-lineal-template.ipynb](https://static.platzi.com/media/public/uploads/primera_regresion_lineal_template_3963ca26-1363-48a2-80ed-5b4b18d4afda.ipynb" \t "_blank)

[primera-regresion-lineal.ipynb](https://static.platzi.com/media/public/uploads/primera_regresion_lineal_2bf0c62c-c174-4577-abd9-a81b1f3c2f57.ipynb" \t "_blank)

[housing.data](https://static.platzi.com/media/public/uploads/housing_fa62d090-9515-4031-a808-cfee2b6ae8cd.data" \t "_blank)

**Lecturas recomendadas**

[Primera\_regresión\_lineal\_Template.ipynb - Google Drive](https://drive.google.com/file/d/1NkcOQg-BTEAD0ttGDcZcAZwYpD0jXPJd/view?usp=sharing" \t "_blank)

[Primera\_regresión\_lineal.ipynb - Google Drive](https://drive.google.com/file/d/1wh1T5AE1AgmgDcTaiziaeKYSuovVFQPk/view?usp=sharing" \t "_blank)

Excelente aporte; aquí les dejo las líneas de código:

sns.set(font\_scale = 1.5)

sns.heatmap(df[cols].corr(), cbar = True, annot=True, yticklabels=cols, xticklabels=cols);

se colocan las librerias y crea graficos

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

cols = ['DIS','INDUS','CRIM','RM','MEDV']

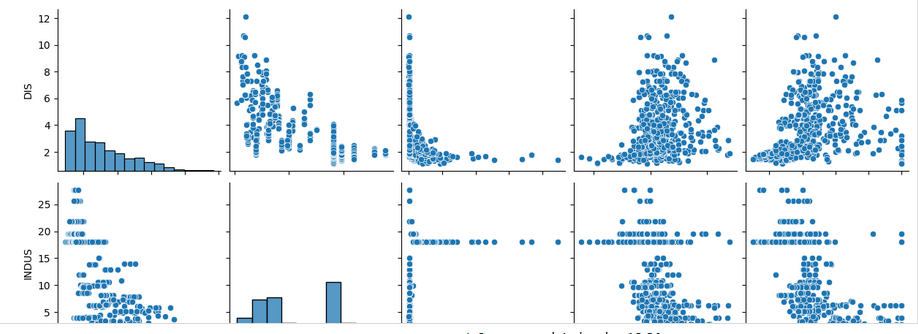
df= pd.read\_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data",header=None, sep='\s+')

df.columns = ["CRIM","ZN","INDUS" , "CHAS" ,"NOX","RM" ,"AGE","DIS", "RAD","TAX", "PTRATIO", "B" , "LSTAT" , "MEDV"]

df.head()

sns.pairplot(df[cols], height=2.5)

plt.show()



Variable a predecir: mediana de los precios, MEDV

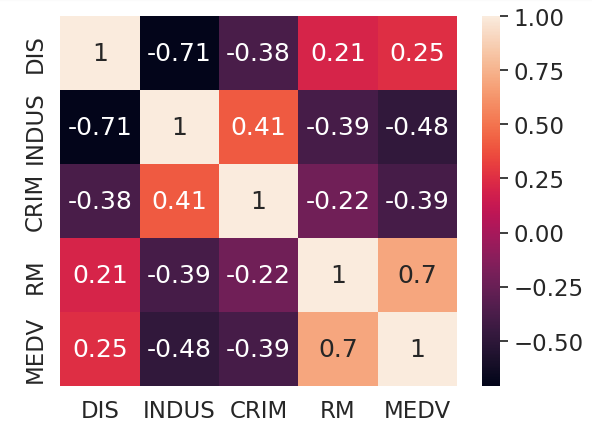
import numpy as np

cm=np.corrcoef(df[cols].values.T)#coeficiente de correlación, se la un arreglo de datos con valores transpuestos

sns.set(font\_scale=1.5)

sns.heatmap(cm, cbar=True,annot=True,yticklabels=cols,xticklabels=cols)

#cbar coloque barra al lado de información, poner anotaciones, y coloque la información de las columnas que se utilizan



1. CRIM - tasa de criminalidad per cápita por ciudad
2. ZN: proporción de suelo residencial zonificado para lotes de más de 25.000 pies cuadrados.
3. INDUS: proporción de acres de negocios no minoristas por ciudad.
4. CHAS - Variable ficticia de Charles River (1 si el tramo limita con el río; 0 en caso contrario)
5. NOX - concentración de óxidos nítricos (partes por 10 millones)
6. RM - número medio de habitaciones por vivienda
7. EDAD: proporción de unidades ocupadas por sus propietarios construidas antes de 1940
8. DIS: distancias ponderadas a cinco centros de empleo de Boston
9. RAD - índice de accesibilidad a carreteras radiales
10. IMPUESTO: tasa de impuesto a la propiedad de valor total por $10,000
11. PTRATIO - ratio alumnos-maestro por ciudad
12. B - 1000(Bk - 0,63)^2 donde Bk es la proporción de negros por ciudad
13. LSTAT - % de estatus inferior de la población
14. MEDV - Valor medio de viviendas ocupadas por sus propietarios en miles de dólares
15. **Entrenando un modelo de regresión lineal con scikit-learnClase 3 / 18**

**Archivos de la clase**

[primera-regresion-lineal.ipynb](https://static.platzi.com/media/public/uploads/primera_regresion_lineal_a0b1dfed-7f52-4e78-b1c3-d5a409bff8fc.ipynb" \t "_blank)

[primera-regresion-lineal-template.ipynb](https://static.platzi.com/media/public/uploads/primera_regresion_lineal_template_1fab1b4e-d890-4567-b99e-1c3b479673ee.ipynb" \t "_blank)

**Lecturas recomendadas**

[scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/" \t "_blank)

[sklearn.linear\_model.LinearRegression — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \t "_blank)

[sklearn.preprocessing.StandardScaler — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html" \t "_blank)

[Primera\_regresión\_lineal.ipynb - Google Drive](https://drive.google.com/file/d/1wh1T5AE1AgmgDcTaiziaeKYSuovVFQPk/view?usp=sharing" \t "_blank)

[Primera\_regresión\_lineal\_Template.ipynb - Google Drive](https://drive.google.com/file/d/1NkcOQg-BTEAD0ttGDcZcAZwYpD0jXPJd/view?usp=sharing" \t "_blank)

Creando modelo con sklearn

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

x = df['RM'].values.reshape(-1, 1)

y = df['MEDV'].values.reshape(-1, 1)

sc\_x = StandardScaler()

sc\_y = StandardScaler()

X\_std = sc\_x.fit\_transform(x)

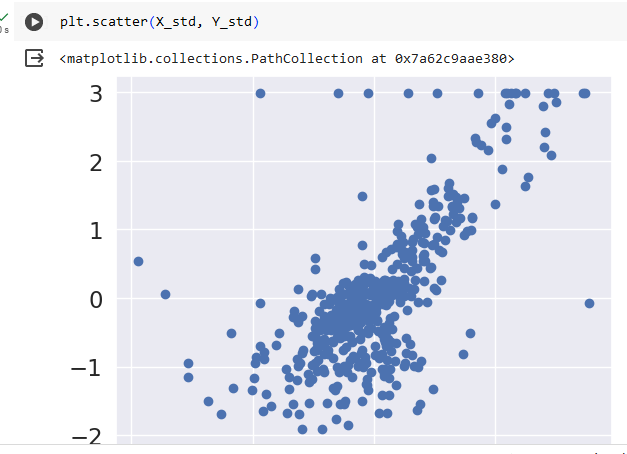
Y\_std = sc\_y.fit\_transform(y)

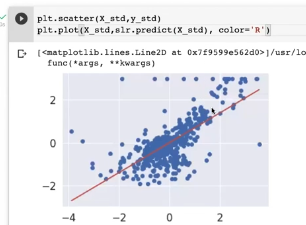
slr = LinearRegression()

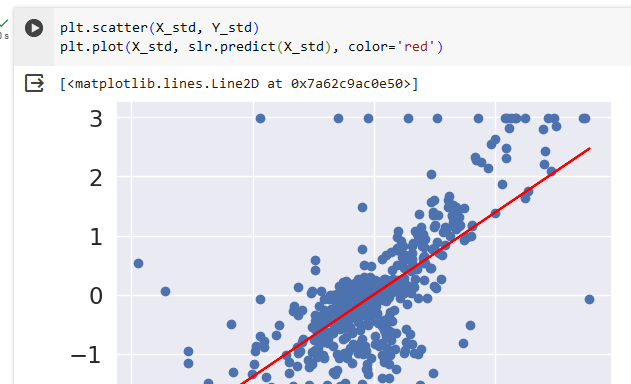
slr.fit(X\_std, Y\_std)

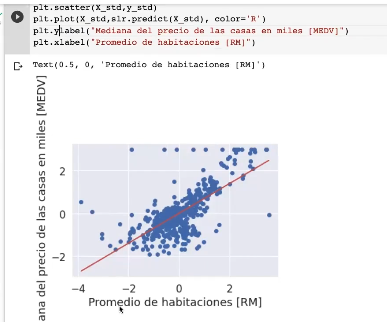
LinearRegression()

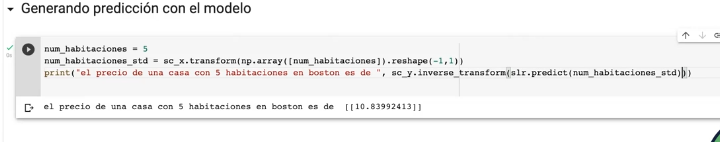
Quizas sirva de aclaracion para alguien, con el **-1** en **reshape(-1,1)** le decimos a numpy que 'decifre' la nueva dimension basado en la longitud de la dimension previa, es decir si teniamos una dimension de **(4,)** ahora tendremos una dimension igual a **(4,1)**, si hubieramos puesto **reshape(1,-1)** tendríamos una dimensión igual a **(1,4)** | Fuente: <https://stackoverflow.com/questions/18691084/what-does-1-mean-in-numpy-reshape>











Nota: se aplica la inversa transformada

numero\_habitaciones = 5

numero\_habitaciones\_std=sc\_x.transform(np.array([numero\_habitaciones]).reshape(-1,1))

print('El precio de una casa con cinco habitaciones en boston es de', sc\_y.inverse\_transform(slr.predict(numero\_habitaciones\_std)))

La estandarización se usa por varios motivos. Entre ellos cuando las variables tienen diferentes unidades. Por ejemplo, si quisiéramos analizar el peso y estatura de individuos, tendríamos variables muy distintas. Una persona puede medir 182 cm y pesar 72 kg. De 72 a 182 hay una diferencia de 110. Entonces, al estandarizar, se pretende minimizar ese efecto.

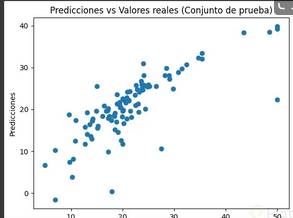
Para machine learning, previo a la creación de los modelos, hay un paso que se llama **feature engineering** que trata sobre todos estos procedimientos de estandarización y normalización.

**Tambien se pueden dividir**

**Se divide en variables predictoras (X) y la variable objetivo (y). Luego, se utiliza la función train\_test\_split de Scikit-Learn para dividir los datos en conjuntos de entrenamiento y prueba.**

* A continuación, se crea el modelo de regresión lineal utilizando la clase LinearRegression de Scikit-Learn, y se entrena el modelo utilizando los datos de entrenamiento llamando al método fit.
* Una vez entrenado el modelo, se utilizan los datos de prueba (X\_test) para hacer predicciones llamando al método predict, y se almacenan en la variable y\_pred. Luego, se calcula el error cuadrático medio (MSE) de las predicciones utilizando la función mean\_squared\_error de Scikit-Learn.

Finalmente, se imprime el valor del MSE como una medida del rendimiento del modelo. Un valor de MSE más bajo indica un mejor ajuste del modelo a los datos de prueba.



Recuerda ajustar la ruta del archivo "housing.data" según la ubicación real del archivo en tu sistema. Además, puedes realizar ajustes adicionales según tus necesidades, como realizar una validación cruzada o aplicar técnicas de regularización.

**¿Qué es la regresión lineal?Clase 4 / 18**

La regresión lineal es una técnica que nos ayuda a hacer comparaciones entre los puntos en el plano cartesiano. es decir, que para un valor de X existe un valor de Y, esto se empieza a graficar sobre el plano como un grafico de dispersión

para esos datos, hay que ajustar una linea recta que mejor los entienda, ósea el modelo trata de predecir el mejor ajuste para la linea recta en esos datos.

La función matemática correspondiente a este problema es Y=w0 + w1x

* X es un valor que existe sobre el plano
* W0 es el punto que se esta cortando sobre el plano, es decir, cuando X = 0 cual es el salto que existe el punto de X = 0 y el plano en el eje Y.
* W1 es cuando se agrega un valor a la X cuanto salto sobre Y

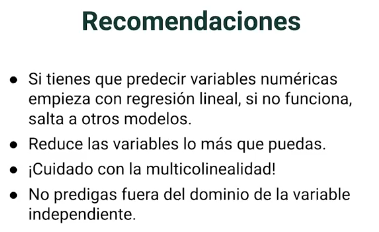
Básicamente esto se conoce como w0 el Intercepto y w1 la pendiente de la linea recta.

durante el ajuste del modelo de regresión lineal se conocen como los pesos del modelo, estos son los valores que va a estar intentando aprender el modelo para entender cual va a ser la mejor linea recta que se va a ajustar a los datos.

**Cuándo utilizar un modelo de regresión linealClase 5 / 18**

3 grandes preguntas para saber cuando utilizar una regresion lineal:

1. ¿Tengo que predecir una variable numerica? la respuesta debe ser afirmativa.
2. ¿Las variables independientes con las que cuento son primordialmente numericas? debe ser afirmativa
3. Cuento con una gran cantidad de avriables categoricas? o poicas variables con muchos niveles? no es aconsejable utilizr la regresion lineal en caso de este numerals er afirmativo.



**Ejemplos de cuándo utilizar la regresión lineal**

Puede usar la regresión lineal cuando intente aprender más sobre la relación entre diferentes variables de datos. A continuación, se muestran algunos ejemplos específicos de escenarios en los que se puede utilizar este proceso de análisis estadístico:

**Ejemplo 1:** compromiso de marketing

Personalized Health Now, un servicio virtual de atención de médico a paciente, está interesado en comprender la relación entre los niveles de participación de sus nuevos esfuerzos de marketing y la cantidad de pacientes nuevos que han recibido desde entonces. Usan sus datos de participación de marketing para la variable independiente y los nuevos números de pacientes para la variable dependiente. Luego, consulte sus tablas de datos organizadas en los ejes yy x para terminar de resolver el cálculo. El resultado puede proporcionar a Personalized Health Now información sobre si sus nuevos esfuerzos de marketing los están ayudando a obtener nuevos pacientes y, de ser así, la empresa puede optar por continuar o avanzar en esos esfuerzos.

Relacionado: Análisis de regresión múltiple: definición y cómo calcular

**Ejemplo 2: análisis deportivo**

Peyton Craft, un presentador de podcasts de analistas deportivos, determina las predicciones para un próximo campeonato deportivo y las comparte con los oyentes. Comienza este proceso investigando las ganancias y pérdidas estadísticamente. El número de victorias representa la variable independiente y el número de pérdidas representa la variable dependiente en su análisis de regresión lineal. Para cada equipo, el Sr. Craft completa el cálculo con su tabla de datos y luego compara los resultados. Esta comparación puede proporcionarle la información que desea compartir con sus oyentes de podcasts antes del evento deportivo. Ejemplo 3: Procesos ambientales

Conservation Clam, una organización ambientalista, está interesada en comprender los efectos de la contaminación en los niveles de sequía. La organización utiliza sus datos de nivel de contaminación como variable independiente y los datos de sequía como variable dependiente. Luego, los analistas e investigadores ambientales organizan las tablas de datos para terminar de resolver el cálculo. El resultado de Conservation Clam puede proporcionar a los equipos la información que necesitan para comprender los efectos de la contaminación en los niveles de sequía. Los equipos ambientales dentro de la organización también pueden usar estos datos para un estudio de caso para involucrar inversionistas, asegurar financiamiento y comunicarse con el público sobre los esfuerzos de conservación en curso.

[fuente](https://www.iccbram.com/desarrollo-de-carrera/comprender-cuando-utilizar-la-regresion-lineal-con-ejemplos.html)

**El modelo de regresión lineal es útil en varias situaciones en las que deseamos comprender y predecir la relación entre una variable dependiente y una o más variables independientes. A continuación, se presentan algunos ejemplos detallados de cuándo se puede utilizar un modelo de regresión lineal:**

**Predicción de precios de viviendas:**

* Si tenemos datos sobre características de viviendas (como el número de habitaciones, el área, la ubicación, etc.) y queremos predecir los precios de las viviendas, podemos utilizar un modelo de regresión lineal. El modelo puede aprender la relación entre las variables independientes y el precio de las viviendas, lo que nos permite hacer predicciones sobre el precio de una vivienda basándonos en sus características.

**Análisis de ventas:**

* Si tenemos datos sobre variables como el gasto en publicidad, el tamaño del mercado y las ventas de productos, podemos utilizar un modelo de regresión lineal para analizar cómo estas variables influyen en las ventas. Podemos determinar el impacto relativo de cada variable y utilizar el modelo para predecir las ventas en diferentes escenarios.

**Estimación de ingresos:**

* Si tenemos datos sobre características demográficas, como la edad, la educación, la experiencia laboral, etc., y queremos estimar los ingresos de las personas, un modelo de regresión lineal puede ayudarnos. El modelo puede aprender la relación entre las variables independientes y los ingresos y proporcionar estimaciones de ingresos basadas en las características de las personas.

**Recomendaciones para utilizar un modelo de regresión lineal:**

**Verificar supuestos:**

* Antes de aplicar un modelo de regresión lineal, es importante verificar los supuestos del modelo, como la linealidad de la relación, la independencia de los residuos y la homocedasticidad. Si los supuestos no se cumplen, es posible que el modelo no sea apropiado y se requiera una técnica diferente.

**Evaluar la calidad del ajuste:**

* Es importante evaluar la calidad del ajuste del modelo de regresión lineal. Esto se puede hacer mediante la evaluación de las métricas de rendimiento, como el error cuadrático medio (MSE) o el coeficiente de determinación (R²). Una buena calidad del ajuste indica que el modelo se ajusta bien a los datos y puede hacer predicciones precisas.

**Considerar la interpretación de los coeficientes:**

* Los coeficientes de regresión en un modelo lineal tienen interpretaciones específicas.

**Evaluar la calidad del ajuste:**

* Es importante comprender cómo afecta cada variable independiente al valor de la variable dependiente y verificar si las relaciones son lógicas y coherentes con el dominio del problema.

**Explorar relaciones no lineales:**

* Si la relación entre las variables no parece ser lineal, se pueden explorar transformaciones de variables o técnicas de regresión no lineal para capturar mejor la relación subyacente.

**Considerar variables adicionales:**

* Si el modelo inicial no proporciona un ajuste satisfactorio, se pueden considerar variables adicionales o interacciones entre variables para mejorar la capacidad predictiva del modelo.

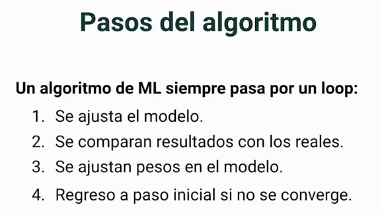
**Recuerda que el modelo de regresión lineal es una herramienta poderosa, pero también tiene limitaciones y suposiciones específicas. Es importante comprender estas limitaciones y considerar alternativas más avanzadas si el modelo lineal no es adecuado para los datos o el problema en cuestión.**

**Función de pérdida y optimización: mínimos cuadradosClase 6 / 18**

#### Lecturas recomendadas

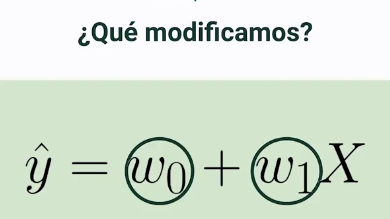
[Regresión Lineal | Aprende Machine Learning](https://www.aprendemachinelearning.com/tag/regresion-lineal/" \t "_blank)

[¿Qué es el descenso del gradiente? - Platzi](https://platzi.com/clases/2155-calculo-data-science/35480-que-es-el-descenso-del-gradiente/" \t "_blank)

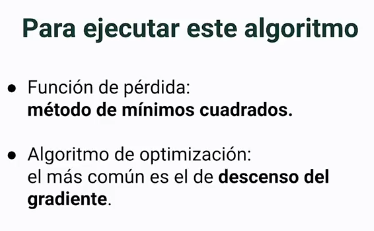


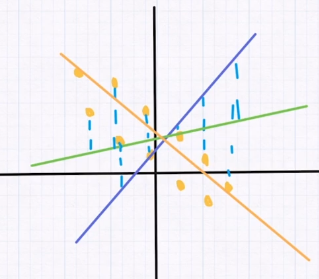


Se empieza con la línea roja y esta se ajusta a la línea azul que es el mejor modelo



Se ajustan los intersectos Wo y W1.





Se inicia con la línea morada, luego se ajusta a la verde y se finaliza con la anaranjanda como la que mejor se ajusta a los datos con el método de mínimos cuadrados y del gradiente.

# Evaluando el modelo: R^2 y MSEClase 7 / 18

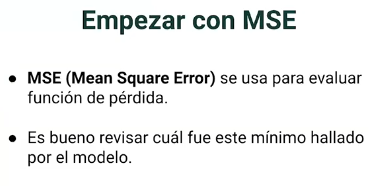
#### Lecturas recomendadas

[3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/modules/model_evaluation.html" \t "_blank)

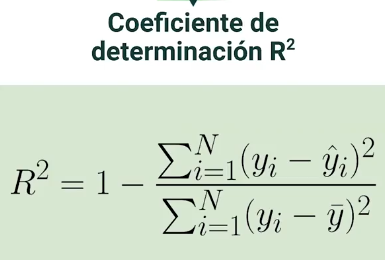
[Difference between Adjusted R-squared and R-squared](https://www.listendata.com/2014/08/adjusted-r-squared.html" \t "_blank)

[Interpreting Residual Plots to Improve Your Regression](https://www.qualtrics.com/support/stats-iq/analyses/regression-guides/interpreting-residual-plots-improve-regression/" \t "_blank)

Como evaluar modelo de regresión







Muestra que también se ajusta el modelo con base en la varianza de los datos (su variabilidad real)

#### R-cuadrado ajustada (R-squared adjusted):

* El coeficiente R-cuadrado ajustado es una medida que indica la proporción de la varianza de la variable dependiente que es explicada por el modelo, teniendo en cuenta el número de variables independientes en el modelo y el tamaño de la muestra. Mientras más cercano a 1 sea el valor del R-cuadrado ajustado, mejor es el ajuste del modelo a los datos.

#### Error máximo (Maximum Error):

* El error máximo es simplemente el valor absoluto máximo de las diferencias entre los valores reales y los valores predichos por el modelo. Esta métrica proporciona una medida de la mayor discrepancia entre las predicciones y los valores reales.

#### Error absoluto promedio (Mean Absolute Error - MAE):

* El error absoluto promedio es la media de las diferencias absolutas entre los valores reales y los valores predichos por el modelo. Esta métrica mide el promedio de las magnitudes de los errores sin considerar su dirección. Un valor de MAE más bajo indica una mejor capacidad predictiva del modelo.

#### Mediana de los errores absolutos (Median Absolute Error):

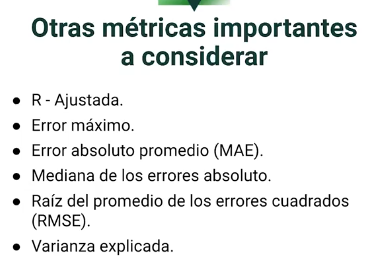
* La mediana de los errores absolutos es el valor que se encuentra en el centro de los errores absolutos, ordenados de menor a mayor. A diferencia del MAE, la mediana no se ve afectada por valores atípicos y proporciona una medida robusta de los errores.

#### Raíz del promedio de los errores cuadrados (Root Mean Squared Error - RMSE):

* El RMSE es la raíz cuadrada del promedio de los errores al cuadrado. Esta métrica penaliza los errores más grandes de manera más significativa que el MAE. Al elevar los errores al cuadrado, los errores negativos y positivos no se cancelan entre sí. El RMSE es una medida comúnmente utilizada para evaluar la precisión de un modelo de regresión y se expresa en las mismas unidades que la variable dependiente.

#### Varianza explicada:

* La varianza explicada, también conocida como coeficiente de determinación (R-squared), indica la proporción de la varianza total de la variable dependiente que es explicada por el modelo. R-squared varía entre 0 y 1, donde 1 indica una ajuste perfecto del modelo a los datos.



Utilizar la función lineal y ponerla en 90 grados y mirar que tan dispersos están los datos.



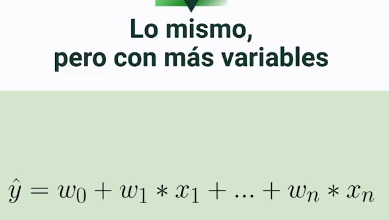
# Regresión lineal multivariableClase 8 / 18

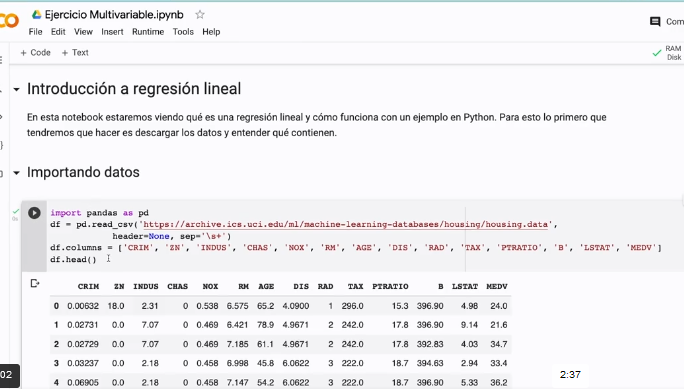
#### Archivos de la clase

[ejercicio-multivariable.ipynb](https://static.platzi.com/media/public/uploads/ejercicio_multivariable_32b1d1b6-256d-48ce-9bcc-10bb8e108079.ipynb" \t "_blank)

#### Lecturas recomendadas

[Ejercicio\_Multivariable.ipynb - Google Drive](https://drive.google.com/file/d/1ONeY0MvSBuSml6zp_g1un_febp1g4NgL/view?usp=sharing" \t "_blank)





Se incorpora la variable INDUS en el modelo



Muestra que se creo el modelo de regresión lineal

La regresión lineal multivariable es una técnica estadística utilizada para modelar la relación entre una variable dependiente y múltiples variables independientes. A diferencia de la regresión lineal simple, que involucra una única variable independiente, la regresión lineal multivariable considera varios predictores para predecir el valor de la variable dependiente.

**++En términos matemáticos, la regresión lineal multivariable se puede expresar como:++**

y = β0 + β1x1 + β2x2 + ... + βn\*xn + ε

Donde:

y es la variable dependiente que queremos predecir.

x1, x2, ..., xn son las variables independientes o predictores.

β0 es el término constante o intercepto.

β1, β2, ..., βn son los coeficientes que representan las pendientes de cada predictor.

ε es el término de error o residual, que captura la variabilidad no explicada por el modelo.

* El objetivo de la regresión lineal multivariable es encontrar los coeficientes β0, β1, β2, ..., βn que minimicen la suma de los cuadrados de los errores (método de mínimos cuadrados). Una vez que se han estimado los coeficientes, podemos utilizar el modelo para realizar

La regresión lineal multivariable es una técnica estadística utilizada para modelar la relación entre una variable dependiente y múltiples variables independientes. A diferencia de la regresión lineal simple, que involucra una única variable independiente, la regresión lineal multivariable considera varios predictores para predecir el valor de la variable dependiente.

**++En términos matemáticos, la regresión lineal multivariable se puede expresar como:++**

y = β0 + β1x1 + β2x2 + ... + βn\*xn + ε

Donde:

y es la variable dependiente que queremos predecir.

x1, x2, ..., xn son las variables independientes o predictores.

β0 es el término constante o intercepto.

β1, β2, ..., βn son los coeficientes que representan las pendientes de cada predictor.

ε es el término de error o residual, que captura la variabilidad no explicada por el modelo.

* El objetivo de la regresión lineal multivariable es encontrar los coeficientes β0, β1, β2, ..., βn que minimicen la suma de los cuadrados de los errores (método de mínimos cuadrados). Una vez que se han estimado los coeficientes, podemos utilizar el modelo para realizar

# Análisis de regresión multivariableClase 9 / 18

#### Archivos de la clase

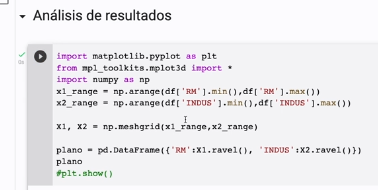
[ejercicio-multivariable.ipynb](https://static.platzi.com/media/public/uploads/ejercicio_multivariable_b5a702ed-290f-4753-814b-eb823e643117.ipynb" \t "_blank)

#### Lecturas recomendadas

[Ejercicio\_Multivariable.ipynb - Google Drive](https://drive.google.com/file/d/1ONeY0MvSBuSml6zp_g1un_febp1g4NgL/view?usp=sharing" \t "_blank)

Hacer un grafico en tres dimensiones con un plano

Ravel, pone el array en una dimensión





Para cada valor del promedio de la casa, predice un valor de la industria

A la predicción se le dice que tenga la misma forma de la variable x1







el codigo es el siguiente: Nota: no realice el paso de escalamiento (para mas facilidad)

#LIBs

import plotly.express as px

import plotly.graph\_objects as go

# TRAINING MODEL

X = df[['RM', 'INDUS']].values

y = df['MEDV'].values.reshape(-1, 1)

slr = LinearRegression()

slr.fit(X, y)

# PLOTTING

mesh\_size = .02

margin = 0

# Create a mesh grid on which we will run our model

x\_min, x\_max = X[:,0].min() - margin, X[:,0].max() + margin

y\_min, y\_max = X[:,1].min() - margin, X[:,1].max() + margin

xrange = np.arange(x\_min, x\_max, mesh\_size)

yrange = np.arange(y\_min, y\_max, mesh\_size)

xx, yy = np.meshgrid(xrange, yrange)

# Run model

pred = slr.predict(np.c\_[xx.ravel(), yy.ravel()])

pred = pred.reshape(xx.shape)

# Generate the plot

fig = px.scatter\_3d(df, x='RM', y='INDUS', z='MEDV')

fig.update\_traces(marker=dict(size=5))

fig.add\_traces(go.Surface(x=xrange, y=yrange, z=pred, name='pred\_surface'))

fig.show()

referencia: <https://plotly.com/python/ml-regression/>

----->Análisis de regresión multivariable<----

σ Objetivo: Reacomodar las lineas de codigo anteriores para poder usar 1 . Importar un herramienta que se le llama toolkits. Usando from mpl\_toolkits.mplot3d import\* + Lo cual nos ayuda a basicamente graficar en 3d + Luis uso " \* " al final, y menciono que significa "everything" (interesante 🤔) 2.Procede a importar numpy usando algo ya visto anteriormente: " import numpy as np" -----------> Explica que se necesita crear un plano, para generar una malla en 3era dimension, y no solo un plano cartesiano. 3. Lo siguiente es usar las variables anteriores " (RM) " " (INDUS) ". Usando " x\_range = np.arrange (df [ 'RM' ].min (), df [ 'RM' ].Max() ) " + Notese que se hizo un " arreglo a la funcion minima de ' RM ', asi como para su funcion Maxima . " (Algo asi como delimitar el alcance minimo y el maxio que tiene la funcion dentro del plano). 4. Continua haciendo un copy / paste del codigo anterior y cambia el nombre de la variable " x1\_range " a " x2\_range " +Esto ya que ahora estamos usando una segunda variable. 5. Interesantemente esta es la funcion que nos permite crear la malla del plano, y esta es la siguiente. " X1, X2 = np.meshgrid(x1\_range, x2\_range) ". +Debido a que igual no estoy entendiendo al 100% lo que esta pasando. Es importante recalcar la maenra tan peculiar en la ques esta escrito el dodigo, la cual es la estructura de la funcion para crear la malla. + Con esto se crea una malla con todas las variables.

6. Todo lo anterior ya definido, ahora solo resta crear el plano, para ello se crea la variable " plano " y se aplica lo siguiente: " plano = pd.DataFrame({‘RM’:X1.ravel(), ‘INDUS’:X2.ravel()}) "

+ se esta defininedo como un Data Frame.

+ Que a su vez tiene como primera variable el promedio de las casas (RM).

+ Notese que despues de ' RM ' se utliza " : " el cual es para darle paso a la definicion que es ," X1. ravel() ".

+" X1.ravel ()" , el cual a su vez esta poniendo (o invocando el arreglo de numpy que se hizo anteriormente), haciendolo mas sencillo y mejor de graficar.

+Por ultimo, Luis agrega solo la siguiente variable que es " ' INDUS ' "y desues se usan dos ( "." ) puntos para deifinir que tiene " X2.ravel () "

+Notese que se esta usando llaves ( " { } " )en donde estan las variables. ...interesante 🤔

σ Una vez definido el plano con el que vamos a trabajar. 1. " pred = slr.predict(plano).reshape(X1.shape)" +Se crea una variable, la cual basicamente se le dice a la prediccion que tenga la misa forma de x1 y que todas las variables tengan las mismas caracteristicas. ( Asi se vuelva mas facil de entender para que maplotlib cree el grafico en 3era dimension ).

2.pred = sc\_y.inverse\_transform(pred)

+Pondresmos la inversa para que muestre los resultados en comparacion con los precios.

+ Luis hace nota de que se esta utilizando " sc\_y " debido a que se esta usando codigo de anteriores ejercicios. Mas que en casos mas como ahora estamos usando 3 variables, esta bien podria llamarse " sc\_z ".

3. Ahora cuando ya tienes la grafica pero ahora sigue definirla como fig = plt.figure()

4. A lo que Luis de inmediatamente despues explica que un axis sobre esta figura de tal manera de que sepa que es una figura en 3 dimensiones. A lo que se usa.

" ax = fig.gca (projection='3d') " +Lo que se le esta diciendo a Matplotlib, es que tiene que hace una grafica en 3 dimensiones. 7. Lo siguiente es crear la superficie, la malla que se va a usar. Para ello ya se tiene la funcion apropiara, la cual es: " ax.plot\_surface (X1,X2,pred, alpha=0.4) " +Diciendo asi que basicamente se va hacer una superficie con la variable" x1", "x2" y la variable " predi " (o prediccion, ya establecida anteriormente). +Asi mismo, agregando al final (y antes de cerrar parentesis) , la cacteristica " alpha = 0.4 ". Lo cual agrega solo un valor de transparencia a la figura, lo que permitira que sea mas legible. ax.plot.dca (projection = '3d') " 8.Ya con esto definido lo ultimo que se tiene que hacer es invocar la version scatter de las variables que ya definiste. Como se muestra a continuacion : " ax.scatter3D( df ['RM'], df ['INDUS'], df ['MEDV'], color = 'R', marker = '.' ) " ---------------->Pasaron muchas cosas. + notese que ax. scatter, ya no es "ax.plotscatter" sino "ax.scatter3D" y esto por que sencillamente ahora estamos usando una grafica 3D y asi que tamb se tomen en cuentan las 3 variables que lo crean. + Y que al final las unicas cosas a destacar tambien, son el que se le pide que el color a usar sea Rojo ( 'R' ) y tamb un marcador que tenga forma de un punto ( '.' ). 9.Lo siguiente que se hizo fue cambiar el punto de vista desde el cual se mostraba la grafica 3D. Para cambiarla se escribio la linea: "ax.view\_init (elev = 10, azim =5) + Asiendo asi el que el modelo se vea desde una perspectiva mas adecuada. y se aprecien mejor los detalles del mismo. 10. Por ultimo se escribe "plt.show" Para mostrar el resultado.

##Plotly

Este código utiliza la biblioteca Plotly para crear una figura 3D con una superficie basada en las predicciones del modelo y puntos dispersos representando los datos reales. Puedes personalizar la apariencia de la gráfica según tus preferencias modificando los parámetros en el código.



# Regresión lineal para predecir los gastos médicos de pacientesClase 10 / 18

#### Lecturas recomendadas

[Medical Cost Personal Datasets | Kaggle](https://www.kaggle.com/mirichoi0218/insurance" \t "_blank)





Función para detectar **valores atípicos** en las variables numéricas continuas del dataset:

def detect\_otliers(x):

Q3 = Datos[x].quantile(0.75)

Q1 = Datos[x].quantile(0.25)

IQR = Q3 - Q1

superior = Q3 + (1.5 \* IQR)

inferior = Q1 - (1.5 \* IQR)

out\_sup = Datos[Datos[x] > superior].index

out\_inf = Datos[Datos[x] < inferior].index

outliers = []

for i in out\_sup:

outliers.append(i)

for j in out\_inf:

outliers.append(j)

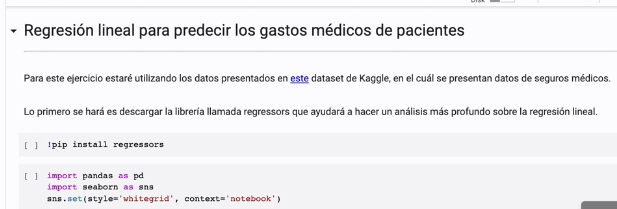
size = len(outliers)

return (f'Hay {size} valores atípicos en la variable {x}, y corresponden a los índices: {outliers}')

detect\_otliers('bmi')

'Hay 8 valores atípicos en la variable bmi, y corresponden a los índices: [115, 285, 400, 843, 856, 1043, 1084, 1310]'

# Exploración y preparación de datosClase 11 / 18



Para el que tenga problemas a la hora de instalar regressors es posible que sea por la versión de setuptools que tienen.

sugiero lo siguiente

pip install --upgrade pip setuptools==57.5.0

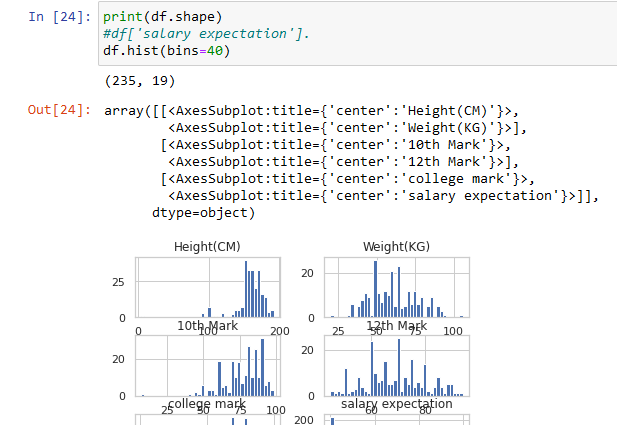
y despues ya instalar regressors

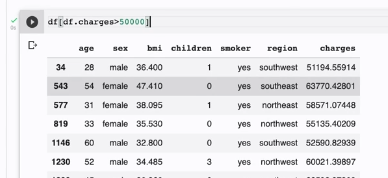
pip install regressors

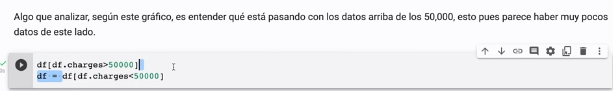




En el ejemplo con estudiantes se cambio la instrucción charges que daba error:









# Análisis de correlación de los datosClase 12 / 18

#### Archivos de la clase

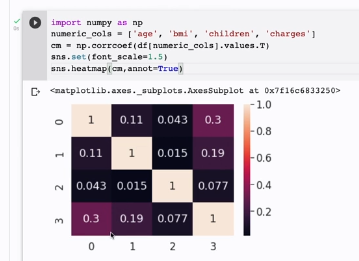
[proyecto-final-template.ipynb](https://static.platzi.com/media/public/uploads/proyecto_final_template_796a7515-cf10-4e9b-b1a1-ad16f2c32db1.ipynb" \t "_blank)

[proyecto-final.ipynb](https://static.platzi.com/media/public/uploads/proyecto_final_a0257c97-4ded-4262-a437-7de7ae4427e5.ipynb" \t "_blank)

#### Lecturas recomendadas

[Proyecto\_Final\_Template.ipynb - Google Drive](https://drive.google.com/file/d/1PrSPTAVz6lXKAqNcTlUvWc_de8TRILOx/view?usp=sharing" \t "_blank)

[Proyecto\_Final.ipynb - Google Drive](https://drive.google.com/file/d/1UDV8y9eyZuKEDSIRKY7w93Bdo7EBqz5Z/view?usp=sharing" \t "_blank)



Ejemplo de filtrado

df[‘campo’].filter(lambda x:x.sum()>2400)

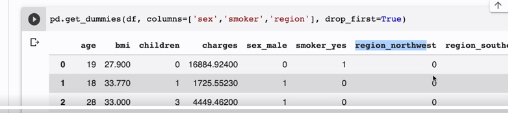


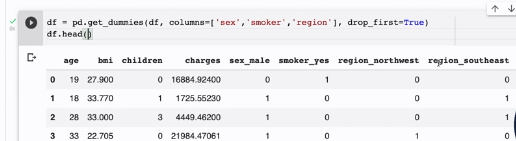
One-hot-codding se requiere porque los analises requieren variables numéricas





Elimina una variable (de las categóricas)





Hago un aporte, utilice estas dos lineas de código para poder sacar nuestras variables numéricas y categoricas por separado pensado en el futuro cuando nos encontremos con dataset que tengas muchas variables, espero que les sirva a todos:

* creo una variable ++**cols\_numerica**++ y en ella almaceno una lista comprimida que evaluara si mi columna es numerica o flotante
* creo una variable ++**cols\_categorica**++ y en ella almaceno una lista comprimida que evaluara si mi columna es de tipo object

cols\_numericas = [colname for colname in df.columns if df[colname].dtype in ['int64', 'float64']]

cols\_categoricas = [colname for colname in df.columns if df[colname].dtype in ['object']]

Otra forma de obtener las variables categóricas convertidas con scikit\_learn.

categorical\_cols = ['sex','smoker','region']

ohe = OneHotEncoder(drop='first')

ohe.fit(df[categorical\_cols])

new = ohe.transform(df[categorical\_cols])

df\_one\_hot\_encoded = df.drop(columns=['sex', 'smoker', 'region'])

df\_one\_hot\_encoded[ohe.get\_feature\_names\_out()] = new.toarray()

df\_one\_hot\_encoded.head(5)

|  | **age** | **bmi** | **children** | **charges** | **sex\_male** | **smoker\_yes** | **region\_northwest** | **region\_southeast** | **region\_southwest** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 19 | 27.9 | 0 | 16884.9 | 0 | 1 | 0 | 0 | 1 |
| 1 | 18 | 33.77 | 1 | 1725.55 | 1 | 0 | 0 | 1 | 0 |
| 2 | 28 | 33 | 3 | 4449.46 | 1 | 0 | 0 | 1 | 0 |
| 3 | 33 | 22.705 | 0 | 21984.5 | 1 | 0 | 1 | 0 | 0 |
| 4 | 32 | 28.88 | 0 | 3866.86 | 1 | 0 |  |  |  |

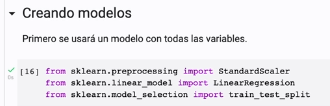
# Entrenamiento del modeloClase 13 / 18

#### Lecturas recomendadas

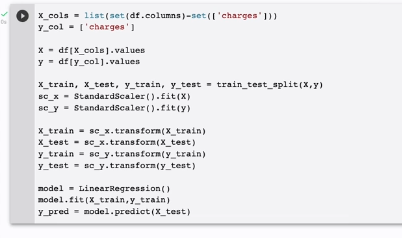
[sklearn.linear\_model.LinearRegression — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \t "_blank)

[Proyecto\_Final.ipynb - Google Drive](https://drive.google.com/file/d/1UDV8y9eyZuKEDSIRKY7w93Bdo7EBqz5Z/view?usp=sharing" \t "_blank)

[Proyecto\_Final\_Template.ipynb - Google Drive](https://drive.google.com/file/d/1PrSPTAVz6lXKAqNcTlUvWc_de8TRILOx/view?usp=sharing" \t "_blank)



Modelo predictor



Predijo datos, y una predicción (solo hay una variable predictora)



Se crearon dos set de datos:

* + Entrenamiento: sirve para ajustar modelo
  + Prueba: evalua modelo y que tan bueno es

Este capítulo me dejó unas dudas que resolví investigando un poco.

1. Al no colocar un test\_size, se asume que toma un 75% para train y 25% para test
2. El .fit lo que haría es obtener la media y la desviación estandar para luego aplicarlo al usar la transformación.
3. Al hacer un fit de X, y es para sacar los valores que describo en 2) y aplicarlos al X\_train, X\_test, y\_train, y\_test
4. No siempre es necesario aplicar un fit\_transform al target
5. En caso uses el método que coloco abajo, debes saber que .fit\_transform se aplica a X\_train, y\_train, y .transform solo se aplica a X\_test, y\_test

Yo lo hice de esta manera, es otra opción:

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

X = data.drop(columns = ['charges'])

y = data[['charges']]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25)

sc\_x = StandardScaler()

X\_train = sc\_x.fit\_transform(X\_train)

X\_test = sc\_x.transform(x\_test)

sc\_y = StandardScaler()

y\_train = sc\_y.fit\_transform(y\_train)

y\_test = sc\_y.transform(y\_test)

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Evaluando el modeloClase 14 / 18

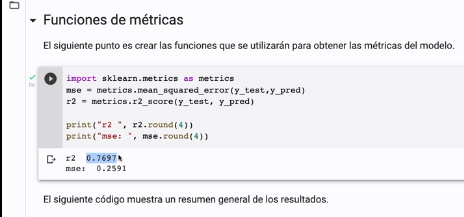
#### Lecturas recomendadas

[3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/modules/model_evaluation.html" \t "_blank)

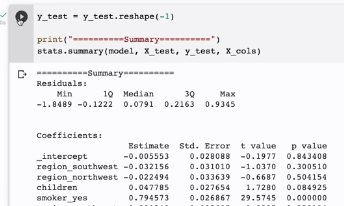
[Proyecto\_Final\_Template.ipynb - Google Drive](https://drive.google.com/file/d/1PrSPTAVz6lXKAqNcTlUvWc_de8TRILOx/view?usp=sharing" \t "_blank)

[Proyecto\_Final.ipynb - Google Drive](https://drive.google.com/file/d/1UDV8y9eyZuKEDSIRKY7w93Bdo7EBqz5Z/view?usp=sharing" \t "_blank)

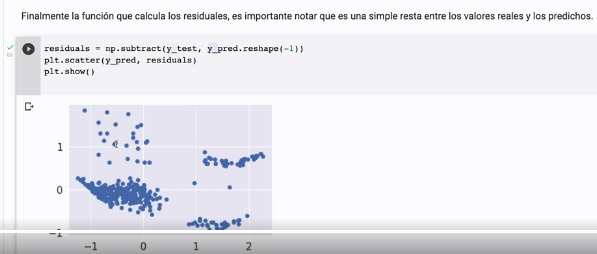
Evalua los valores reales contra los valores predichos







Pvalue es el dato a analizar si vale la pena utilizar esa variable



Se hizo un reshape a “y” en razón a que se había realizado previamente para ajustar el tamaño

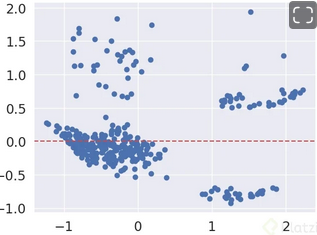
el valor predicho esta sobre la línea cero y los valor reales se ven arriba.

residuals = np.subtract(y\_test, y\_pred.reshape(-1))

plt.scatter(y\_pred, residuals)

plt.axhline(y=0, color='r', linestyle='--') # Agregar la línea en el valor cero

plt.show()



##Error cuadrático medio (MSE, por sus siglas en inglés):

* Es la media de los errores al cuadrado entre los valores predichos y los valores reales. Cuanto menor sea el valor del MSE, mejor será el modelo.

### Raíz del error cuadrático medio (RMSE):

* Es la raíz cuadrada del MSE. Al utilizar la raíz cuadrada, se obtiene una métrica en la misma escala que la variable objetivo. También se utiliza para evaluar la precisión del modelo, donde un valor más bajo indica un mejor ajuste.

### Coeficiente de determinación (R^2):

* Es una medida que indica qué tan bien se ajustan los datos al modelo. R^2 varía entre 0 y 1, donde 1 indica un ajuste perfecto y 0 indica que el modelo no explica la variabilidad de los datos. Un valor cercano a 1 es deseable.

### Error absoluto promedio (MAE, por sus siglas en inglés):

* Es la media de los errores absolutos entre los valores predichos y los valores reales. Proporciona una medida promedio del tamaño de los errores cometidos por el modelo.

**Estas métricas pueden ser calculadas utilizando funciones de evaluación de modelos de la biblioteca scikit-learn. Aquí tienes un ejemplo de cómo calcular estas métricas:**

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

# Calculando las métricas

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

# Imprimiendo los resultados

print("Error cuadrático medio (MSE):", mse)

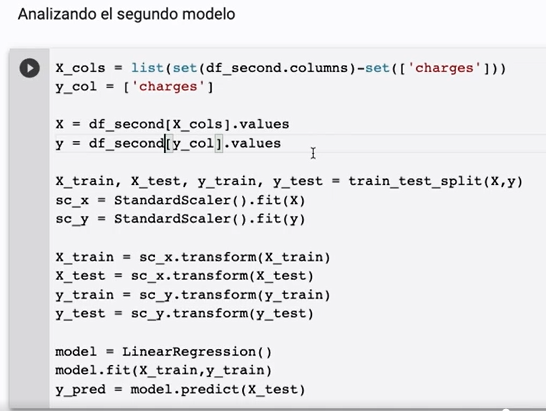
print("Raíz del error cuadrático medio (RMSE):", rmse)

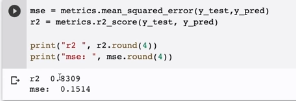
print("Error absoluto promedio (MAE):", mae)

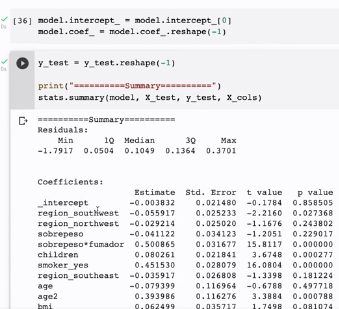
print("Coeficiente de determinación (R^2):", r2)

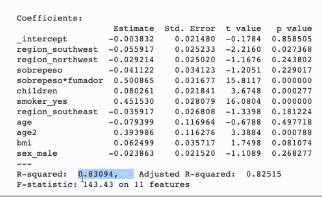
# Mejorando el modeloClase 15 / 18

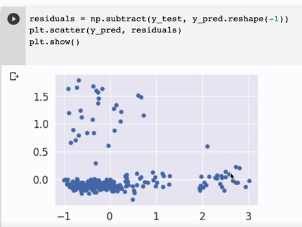




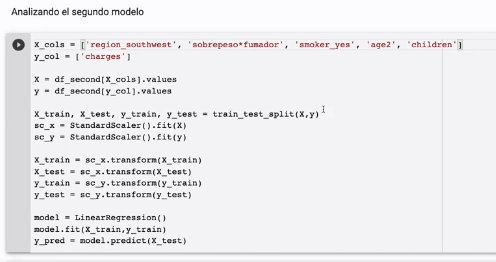




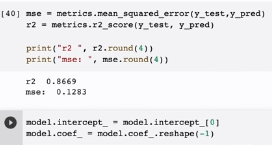


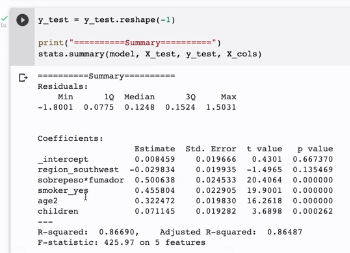


Eliminar las variables con p mayor a 0.5

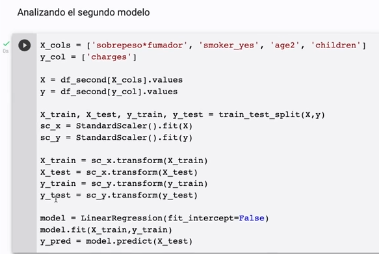


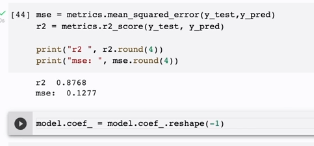
Se vuelven a correr las variables

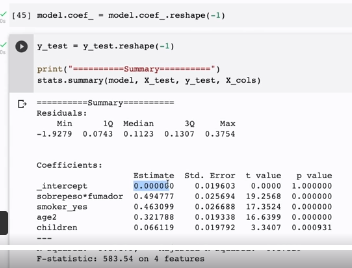




Se eliminan variables y se vuelven a correr







Reto: ir eliminando variables para ver como afecta al modelo

**Para mejorar el modelo de regresión lineal, puedes considerar las siguientes estrategias:**

### Feature Engineering:

* Analiza las variables existentes y busca formas de crear nuevas variables que puedan tener una mejor relación con la variable objetivo. Esto puede incluir transformaciones, combinación de variables o creación de variables dummy.

### Selección de características:

* Evalúa la importancia y relevancia de cada variable en el modelo. Puedes utilizar técnicas como la correlación, la importancia de características o la eliminación recursiva de características para seleccionar las variables más significativas.

### Regularización:

* Considera el uso de técnicas de regularización como la regresión ridge o la regresión lasso para reducir la complejidad del modelo y evitar el sobreajuste. Estas técnicas penalizan los coeficientes de las variables menos relevantes.

### Manejo de valores atípicos:

* Identifica y maneja los valores atípicos en tus datos. Los valores atípicos pueden afectar negativamente el rendimiento del modelo y distorsionar los resultados. Puedes eliminarlos, imputarlos o utilizar técnicas robustas para minimizar su impacto.

### Validación cruzada:

* Utiliza la validación cruzada para evaluar el rendimiento del modelo de manera más robusta. Esto te permite obtener estimaciones más precisas del rendimiento y asegurarte de que el modelo generaliza bien a nuevos datos.

### Regularización de hiperparámetros:

* Ajusta los hiperparámetros del modelo, como la tasa de aprendizaje o el parámetro de regularización, utilizando técnicas como la búsqueda en cuadrícula o la optimización bayesiana. Esto te permite encontrar la configuración óptima que maximice el rendimiento del modelo.

# Qué hay más allá de la linealidad?Clase 16 / 18

#### Lecturas recomendadas

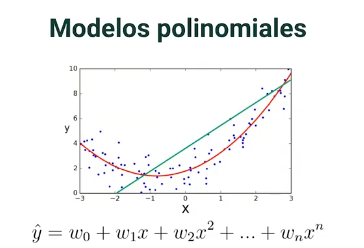
[Ridge and Lasso Regression: L1 and L2 Regularization | by Saptashwa Bhattacharyya | Towards Data Science](https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b" \t "_blank)

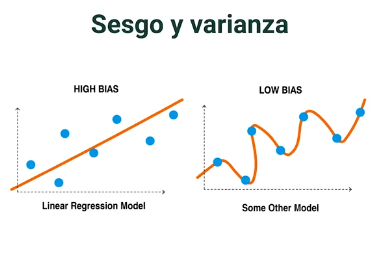
[sklearn.linear\_model.Ridge — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html" \t "_blank)

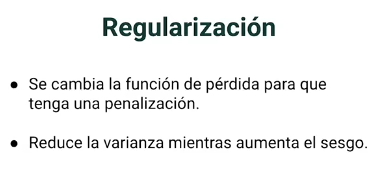
[sklearn.linear\_model.Lasso — scikit-learn 1.0.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html" \t "_blank)

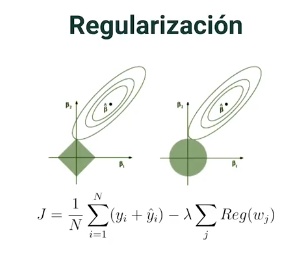
que pasa cuando no hay modelo lineal

Cuando variables elevados a un exponente:









Añadir a la función de costos una penalización para regularizar