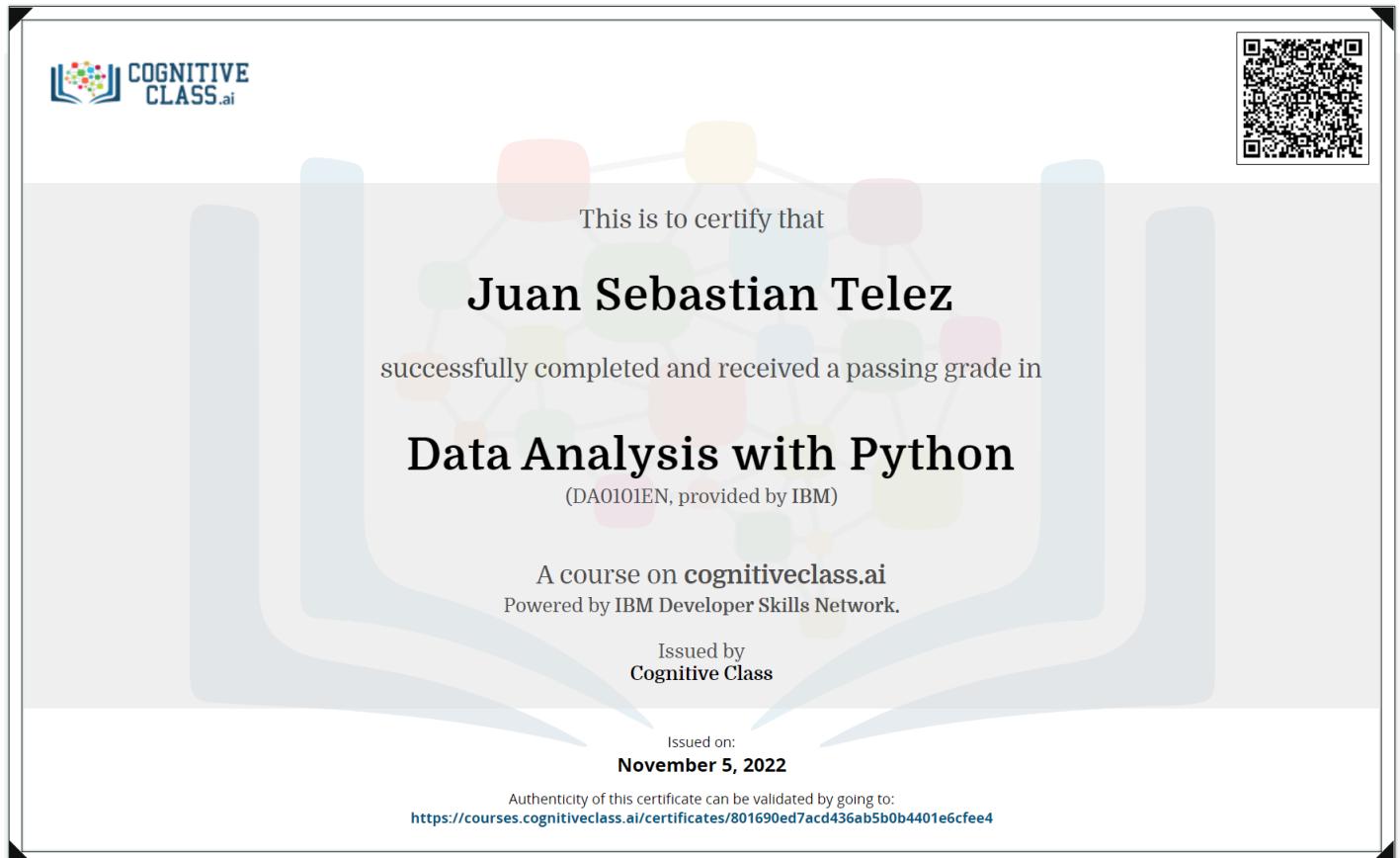


▼ Juan Sebastián Téllez López. A01793859.

Adicionalmente en el archivo "Python by Jtellez" comparto ejercicios realizados por mi cuenta y una guia de aprendizaje de Python.



Course Progress for 'Jtellez624' (A01793859@tec.mx)



▼ Module 1 - Introduction

▼ Introduction to Data Analysis with Python

En este primer modulo se aprendera sobre

- Un problema que requiera análisis de datos.
- Un dataset que será analizado con Python.
- Mirada general a las librerías que podemos utilizar con Python.
- Importar y exportar datos con Python.
- Obtener información relevante del dataset.

Algunas preguntas que podríamos responder serían.

- Podemos estimar el precio de un vehículo usado de acuerdo a sus características?

▼ The Problem

- Es importante reconocer la importancia del análisis de datos, ya que los datos podemos encontrarlos en cualquier lado.
- Los datos no son solo información, sino los hallazgos que podamos realizar con ellos.

Preguntas al problema - Tom quiere vender su vehículo.

- El quiere elegir un precio razonable que represente el valor del carro.
- ¿Cómo podemos ayudarle a determinar el mejor precio?
- En los datos tenemos precios de otros vehículos y sus características (Marca, color, precio)?

▼ Understanding the Data

El dataset que utilizaremos es el siguiente:

Dataset – Used Automobiles (CSV)

```
3?,alfa-romero,gas,std,two,convertible,rwd,front,88.60,168.80,64.10,48.80,2548,dohc,four,130,mpfi,3.47,2.68,9.00,111,5000,21,27,13495
3?,alfa-romero,gas,std,two,convertible,rwd,front,88.60,168.80,64.10,48.80,2548,dohc,four,130,mpfi,3.47,2.68,9.00,111,5000,21,27,16500
1?,alfa-romero,gas,std,two,hatchback,rwd,front,94.50,171.20,65.50,52.40,2823,ohcv,six,152,mpfi,2.68,3.47,9.00,154,5000,19,26,16500
2,164,audi,gas,std,four,sedan,fwd,front,99.80,176.60,66.20,54.30,2337,ohc,four,109,mpfi,3.19,3.40,10.00,102,5500,24,30,13950
2,164,audi,gas,std,four,sedan,4wd,front,99.40,176.60,66.40,54.30,2824,ohc,five,136,mpfi,3.19,3.40,8.00,115,5500,18,22,17450
2?,audi,gas,std,two,sedan,fwd,front,99.80,177.30,66.30,53.10,2507,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,15250
1,158,audi,gas,std,four,sedan,fwd,front,105.80,192.70,71.40,55.70,2844,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,17710
1?,audi,gas,std,four,wagon,fwd,front,105.80,192.70,71.40,55.70,2954,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,18920
1,158,audi,gas,turbo,four,sedan,fwd,front,105.80,192.70,71.40,55.90,3086,ohc,five,131,mpfi,3.13,3.40,8.30,140,5500,17,20,23875
0?,audi,gas,turbo,two,hatchback,4wd,front,99.50,178.20,67.90,52.00,3053,ohc,five,131,mpfi,3.13,3.40,7.00,160,5500,16,22,?
2,192,bmw,gas,std,two,sedan,rwd,front,101.20,176.80,64.80,54.30,2395,ohc,four,108,mpfi,3.50,2.80,8.80,101,5800,23,29,16430
0,192,bmw,gas,std,four,sedan,rwd,front,101.20,176.80,64.80,54.30,2395,ohc,four,108,mpfi,3.50,2.80,8.80,101,5800,23,29,16925
0,188,bmw,gas,std,two,sedan,rwd,front,101.20,176.80,64.80,54.30,2710,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,21,28,20970
0,188,bmw,gas,std,four,sedan,rwd,front,101.20,176.80,64.80,54.30,2765,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,21,28,21105
1?,bmw,gas,std,four,sedan,rwd,front,103.50,189.00,66.90,55.70,3055,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,20,25,24565
0?,bmw,gas,std,four,sedan,rwd,front,103.50,189.00,66.90,55.70,3230,ohc,six,209,mpfi,3.62,3.39,8.00,182,5400,16,22,30760
0?,bmw,gas,std,two,sedan,rwd,front,103.50,193.80,67.90,53.70,3380,ohc,six,209,mpfi,3.62,3.39,8.00,182,5400,16,22,41315
0?,bmw,gas,std,four,sedan,rwd,front,110.00,197.00,70.90,56.30,3505,ohc,six,209,mpfi,3.62,3.39,8.00,182,5400,15,20,36880
2,121,chevrolet,gas,std,two,hatchback,fwd,front,88.40,141.10,60.30,53.20,1488,1,three,61,2bb1,2.91,3.03,9.50,48,5100,47,53,5151
```

Y es importante conocer lo siguiente:

- El dataset está en formato CSV separado por comas.
- Tenemos facilidad de importarlo en diversas herramientas o aplicaciones.
- Cada línea representa una fila.
- No tenemos encabezados, pero se describen a continuación.

No.	Attribute name	attribute range	No.	Attribute name	attribute range
1	symboling	-3, -2, -1, 0, 1, 2, 3.	14	curb-weight	continuous from 1488 to 4066.
2	normalized-losses	continuous from 65 to 256.	15	engine-type	dohc, dohcv, l, ohc, ohcf, ohcv, rotor.
3	make	audi, bmw, etc.	16	num-of-cylinders	eight, five, four, six, three, twelve, two.
4	fuel-type	diesel, gas.	17	engine-size	continuous from 61 to 326.
5	aspiration	std, turbo.	18	fuel-system	1bbl, 2bbl, 4bbl, idl, mfi, mpfi, spdi, spfi.
6	num-of-doors	four, two.	19	bore	continuous from 2.54 to 3.94.
7	body-style	hardtop, wagon, etc.	20	stroke	continuous from 2.07 to 4.17.
8	drive-wheels	4wd, fwd, rwd.	21	compression-ratio	continuous from 7 to 23.
9	engine-location	front, rear.	22	horsepower	continuous from 48 to 288.
10	wheel-base	continuous from 86.6 120.9.	23	peak-rpm	continuous from 4150 to 6600.
11	length	continuous from 141.1 to 208.1.	24	city-mpg	continuous from 13 to 49.
12	width	continuous from 60.3 to 72.3.	25	highway-mpg	continuous from 16 to 54.
13	height	continuous from 47.8 to 59.8.	26	price	continuous from 5118 to 45400.

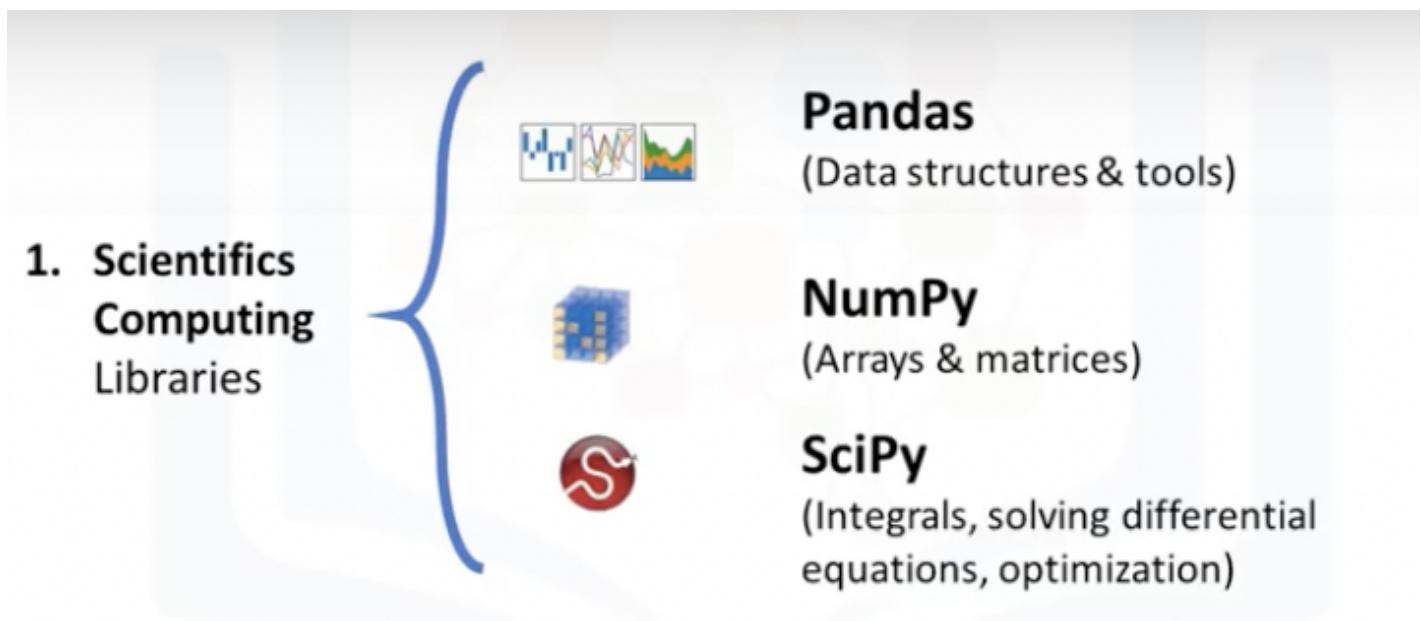
- **Symboling:** Corresponde al nivel de riesgo del vehiculo, donde -3 es muy seguro y +3 inseguro.
- **Normalized - losses:** se refiere a cuanto disminuye el valor del vehiculo con los años.
- **Make:** Es la marca del vehiculo.
- **Fuel -type:** Tipo de combustible.
- **Aspiration:** Se refiere al tipo de combustión.
- **Num-of-doors:** Cantidad de puertas.
- **Body-style:** Tipo de vehiculo.
- **Drive - wheels:** Tipo de tracción.
- **Engine - location:** Ubicación del motor.
- **Wheel - base:** Distancia de las llantas delanteras de las traseras.
- **Lenght:** Longitud.
- **Width:** Ancho.
- **Height:** Altura.
- **Curb-weight:** Peso total del vehiculo con accesorios estandar y tanque de combustible lleno.
- **Engine - type:** Tipo de motor.
- **Num-of-Cylinders:** Cantidad de cilindros.
- **Engine-size:** Tamaño del motor.
- **Fuel system:** Sistema de combustible.
- **Bore:** Tamaño del piston del motor.
- **Stroke:** Movimiento del piston.
- **Compression-ratio:** La relación entre el volumen del cilindro con el pistón en la posición inferior.
- **Horsepower:** Caballos de fuerza.
- **Peak-rpm:** Pico de revoluciones por minuto.
- **City-mpg:** la calificación de mpg más baja para un vehículo principalmente debido a los frecuentes arranques, paradas y marcha en vacío.

- **Highway-mpg:** El promedio que obtendrá un automóvil mientras conduce en un tramo abierto de la carretera sin detenerse ni arrancar, generalmente a una velocidad más alta.
- **Price:** Precio.

▼ Python Packages for Data Science

- Estas librerías permiten ejecutar modulos pre-construidos con diferentes funcionalidades.

El primer grupo son librerías de computación científica.



Pandas: Ofrece una efectiva manipulación de los datos y su análisis, donde su instrumento principal es una tabla bidimensional que consiste en filas y columnas, lo cual conocemos como DataFrame.

Numpy: Utiliza arrays para sus entradas y salidas, también permite el trabajo con matrices con un alto nivel de procesamiento.

SciPy: Ofrece funciones matemáticas avanzadas, para análisis más profundos.

Por otro lado, tenemos las librerías de visualización, donde encontramos las siguientes:

2. Visualization Libraries



Matplotlib

(plots & graphs, most popular)

Seaborn

(plots : heat maps, time series, violin plots)

Matplotlib: Es de las mas conocidas para realizar graficos con un alto nivel de personalización.

Seaborn: Basada en Matplotlib, permite generar distintos tipos de graficos, como mapas de calor, series de tiempo entre otros.

En cuanto a la tercera categoria, tenemos las librerias para algoritmos de machine learning:

3. Algorithmic libraries



Scikit-learn

(Machine Learning : regression, classification,...)

Statsmodels

(Explore data, estimate statistical models, and perform statistical tests.)

Scikit-learn: Permite realizar modelamiento estadistico como regresión, clasificación y clustering.

StatsModels: Permite realizar estimaciones estadisticas y pruebas.

▼ Importing and Exporting Data in Python

- Es el proceso de cargar y leer datos en Python desde diversas fuentes.
- Debemos tener en cuenta dos factores, formato y ruta del archivo. Teniendo en cuenta que formato es como viene codificado (csv,json,xlsx). Y en cuanto la ruta del archivo es si lo

tenemos de manera local o en algun lugar de internet.

- Cada fila es un punto de dato. Debido que las propiedades indican separación por comas, podemos intuir que el formato es csv (Delimitado por comas).
- En pandas "read_csv()" permite leer archivos con separación de comas.

Para leer datos en pandas podemos hacer lo siguiente:

```
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data"

df = pd.read_csv(url, header = None)
```

Teniendo en cuenta que para leer los datos podemos hacer lo siguiente:

- `df` imprime el dataframe (No es recomendado para grandes cantidades de datos).
- `df.head(n)` Muestra las primeras n filas del dataframe.
- `df.tail(n)` Muestra las ultimas n filas del dataframe.

A continuación observamos que los encabezados se ponen automaticamente en una lista de numeros enteros, ya que es dificil trabajar con dataframes que no tienen nombres de columnas.

0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25	
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

También podemos asignar nombres a las columnas de la siguiente manera:

- Replace default header (by `df.columns = headers`)

```
headers = ["symboling","normalized-losses","make","fuel-type","aspiration","num-of-doors","body-style",
"drive-wheels","engine-location","wheel-base","length","width","height","curb-weight","engine-type",
"num-of-cylinders","engine-size","fuel-system","bore","stroke","compression-ratio","horsepower","peak-
rpm","city-mpg","highway-mpg","price"]
```

```
df.columns=headers
```

- Para exportar nuestro dataframe a un archivo csv podemos utilizar "to_csv()", y si quisieramos ser especificos con el nombre utilizariamos `df.to_csv("automobile.csv")`.

A continuación tenemos otros metodos para guardar información a otros formatos.

Data Format	Read	Save
csv	pd.read_csv()	df.to_csv()
json	pd.read_json()	df.to_json()
Excel	pd.read_excel()	df.to_excel()
sql	pd.read_sql()	df.to_sql()

Getting Started Analyzing Data in Python

Es importante tener en cuenta lo siguiente:

- Tipos de datos.
- Distribución de los datos.
- Ubicar dificultades con los datos.

Algunos tipos de datos son los siguientes:

Pandas Type	Native Python Type	Description
object	string	numbers and strings
int64	int	Numeric characters
float64	float	Numeric characters with decimals
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	time data.

Algunos aspectos a tener en cuenta:

- Incongruencia con la información.
- Cambios en tipos de datos específicos a una columna.
- Debemos tener en cuenta que algunas funciones matemáticas deben ser solo aplicadas a datos numéricos. O sino obtendremos errores de tipo.
- Cuando el "dtype" es aplicado al dataset de cada columna retorna una serie.

```

symboling      int64
normalized-losses   object
make          object
fuel-type      object
aspiration     object
num-of-doors    object
body-style      object
drive-wheels    object
engine-location  object
wheel-base     float64
length         float64
width          float64
height          float64
curb-weight     int64
engine-type     object
num-of-cylinders  object
engine-size      int64
fuel-system     object
bore          object
stroke         object
compression-ratio float64
horsepower      object
peak-rpm        object
city-mpg        int64
highway-mpg     int64
price           object
dtype: object

```

- Es importante ser minuciosos al revisar nuestros datos antes de empezar con cualquier tipo de análisis.
- Si realizamos una correcta revisión podemos evitar valores extremos o desviaciones.
- Si utilizamos "df.describe()" obtendremos una breve descripción de algunas métricas estadísticas que permitirán un análisis rápido de la información.

• Returns a statistical summary

`df.describe()`

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

- Por defecto al usar "dataframe.describe()" omite filas que no contienen números, lo cual podemos solucionar incluyendo el argumento "include = all" dentro de los parentesis.

*Este resumen permite encontrar tendencias de como se comportan los datos.

*Con "df.info()" tendremos una descripción general del dataframe.

▼ Lab 1

Question #1:

Check the bottom 10 rows of data frame "df".

```
[11]: # Write your code below and press Shift+Enter to execute
print("The last 10 rows of the dataframe\n")
df.head(10)
```

The last 10 rows of the dataframe

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	15250
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	17710
7	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	18920
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3	140	5500	17	20	23875
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0	160	5500	16	22	?

10 rows × 26 columns

Question #2:

Find the name of the columns of the dataframe.

```
[16]: # Write your code below and press Shift+Enter to execute
print(df.columns)

Index(['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-type',
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price'],
      dtype='object')
```

Question #3:

You can select the columns of a dataframe by indicating the name of each column. For example, you can select the three columns as follows:

```
dataframe[[' column 1 ',column 2', ' column 3']]
```

Where "column" is the name of the column, you can apply the method ".describe()" to get the statistics of those columns as follows:

```
dataframe[[' column 1 ',column 2', ' column 3']] .describe()
```

Apply the method to ".describe()" to the columns 'length' and 'compression-ratio'.

```
[1]: # Write your code below and press Shift+Enter to execute
df[['length', 'compression-ratio']].describe()
```

```
1]:
length compression-ratio
count    201.000000        201.000000
mean     174.200995        10.164279
std      12.322175        4.004965
min      141.100000        7.000000
25%     166.800000        8.600000
50%     173.200000        9.000000
75%     183.500000        9.400000
max      208.100000        23.000000
```

▼ Graded Review Questions 1

Question 1

1/1 point (graded)

What does CSV stand for?



Comma-separated values



Car sold values



Car state values



None of the above



Sav

Submit

You have used 1 of 2 attempts



Correct (1/1 point)

Question 2

1/1 point (graded)

In the data set, which of the following represents an attribute or feature?

Row

Column

Each element in the dataset



Sav

Submit

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 3

1/1 point (graded)

What is the name of what we want to predict?

Target

Feature

Dataframe



Sa

Submit

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 4

1/1 point (graded)

What is the command to display the first five rows of a dataframe `df`?



`df.head()`



`df.tail()`



Submit

You have used 1 of 1 attempt

Correct (1/1 point)

Question 5

1/1 point (graded)

What command do you use to get the data type of each row of the dataframe `df` ?



`df.dtypes`



`df.head()`



`df.tail()`



Sav

Submit

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 6

1/1 point (graded)

How do you get a statistical summary of a dataframe `df` ?



`df.describe()`



`df.head()`



`df.tail()`



Sav

Submit

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 7

1/1 point (graded)

If you use the method `describe()` without changing any of the arguments, you will get statistical summary of all the columns of type "object".

False

True



Submit

You have used 1 of 1 attempt

✓ Correct (1/1 point)

Module 2 - Data Wrangling

Pre-processing Data in Python

- El proceso de convertir datos sin procesar, de un formato a otro.
- También es conocido como limpieza de datos.
- En este modulo se identificaran y trabajaran con datos perdidos.
- Formato de datos.
- Normalización de datos.
- Agrupación de datos.
- Pasar de valores categoricos a numericos.
- Operaciones simples entre Dataframes.
- Series en Pandas.

Dealing with Missing Values in Python

- Los valores perdidos ocurren cuando el valor de un dato no está almacenado en donde corresponde.
- Puede ser representado por "?", "N/A", 0, o una celda en blanco.
- La primera posibilidad es revisar si la fuente de los datos tiene la información completa.
- Otra posibilidad es eliminar los datos perdidos (Podemos eliminar solo el campo perdido o toda la variable).
- Si removemos datos tenemos que detectar el impacto que puede tener.
- Reemplazar los datos es siempre la mejor opción, podemos utilizar el promedio, la media, mediana.
- Para las variables categóricas podemos utilizar la moda.
- Solo en algunos casos podemos simplemente dejar los datos perdidos.
- Para eliminar datos con valores perdidos en Python podemos hacer lo siguiente.

• Use `dataframes.dropna()`:

highway-mpg	price
...	...
20	23875
22	NaN
29	16430
...	...

Es importante especificar lo siguiente:

- "`axis=0`" Elimina las filas.
- "`axis=1`" Elimina las columnas.



The diagram illustrates the difference between `axis=0` and `axis=1` when using `dropna()`. On the left, a DataFrame has a row with a `NaN` value in the 'price' column highlighted. A red dashed line indicates the row is being considered for deletion. An arrow points to the right, where the same DataFrame is shown without that row, labeled as "axis=0 drops the entire row". Below the arrow, another label "axis=1 drops the entire column" is present, though it is visually aligned with the arrow.

highway-mpg	price
...	...
20	23875
22	NaN
29	16430
...	...

axis=0 drops the entire row
axis=1 drops the entire column

En este caso si queremos eliminar toda la columna de precios podemos hacer lo siguiente:

```
df.dropna(subset=["price"], axis=0, inplace = True)
```

Al poner el argumento "Inplace = True", permitimos la modificación de dataset directamente.

- Si queremos reemplazar valores con NaNs podemos utilizar el siguiente metodo.

Use dataframe.replace(missing_value, new_value):

Para este caso lo utilizaremos reemplazando el valor NaN de la columna Normalized-losses por el promedio de los datos que contiene.

normalized-losses	make
...	...
164	audi
164	audi
NaN	audi
158	audi
...	...

→

normalized-losses	make
...	...
164	audi
164	audi
162	audi
158	audi
...	...

```
mean = df["normalized-losses"].mean()
df["normalized-losses"].replace(np.nan, mean)
```

También podríamos cambiar mean por median, o abs. Dependiendo el tipo de modificación que queramos hacer a los datos perdidos.

▼ Data Formatting in Python

- Los datos son recolectados de diferentes lugares y guardados en diferentes formatos.
- Llevar los datos a una expresión estandar, permite a los usuarios compararlos mas facilmente.

Non-formatted:

- confusing
- hard to aggregate
- hard to compare

City
NY
New York
N.Y
N.Y



City
New York
New York
New York
New York

Formatted:

- more clear
- easy to aggregate
- easy to compare

- Si observamos al no usar abreviaciones, podemos entender los datos de una manera mas sencilla.

• Convert "mpg" to "L/100km" in Car dataset.

city-mpg
21
21
19
...



city-L/100km
11.2
11.2
12.4
...

```
df["city-mpg"] = 235/df["city-mpg"]
```

- En este ejemplo hacemos una conversión de datos, y con la siguiente instrucción renombramos la nueva columna.

```
df.rename(columns={"city_mpg": "city-L/100km"}, inplace=True)
```

- En algunas ocasiones tenemos datos con tipos que no corresponden, debido que fueron incorrectamente establecidos.
- Por ejemplo, observamos que el tipo de dato a la columna precio es de tipo "Objeto", cuando esperamos un tipo "Flotante o entero".

```
df[“price”].tail(5)

200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, dtype: object
```

- Tenemos diversos tipos de datos en pandas.

- There are many data types in pandas
- Objects : “A”, “Hello”..
- Int64 : 1,3,5
- Float64 : 2.123, 632.31,0.12

- Con los siguientes métodos podemos encontrar los tipos de datos y hacer su respectiva conversión

To identify data types:

- Use `dataframe.dtypes()` to identify data type.

To convert data types:

- Use `dataframe.astype()` to convert data type.

- Para el caso de nuestro ejemplo convertiremos el tipo de dato de la columna precio a entero de la siguiente manera.

Example: convert data type to integer in column “price”

```
df[“price”] = df[“price”].astype(“int”)
```

▼ Data Normalization in Python

- En este caso observamos diferencias de los rangos para width y height, y aquí es donde la normalización entra para hacer mas consistentes los rangos entre variables y poder compararlos.

• Uniform the features value with different range.

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
171.2	65.5	52.4
176.6	66.2	54.3
176.6	66.4	54.3
177.3	66.3	53.1
192.7	71.4	55.7
192.7	71.4	55.7
192.7	71.4	55.9

scale	[150,250]	[50,100]	[50,100]
impact	large	small	small

- Debemos tener en cuenta que al hacer algun cambio los datos sigan teniendo el mismo impacto.
- En el siguiente ejemplo observaremos las columnas "age" e "income" dnde para la primera tenemos rangos de 0 a 100, mientras que para la segunda de 0 a más de 20.000.

age	income
20	100000
30	20000
40	500000

Not-normalized

- “age” and “income” are in different range.
- hard to compare
- “income” will influence the result more

- Esto nos daria a entender que "income" sea 1000 veces mayor que "age", lo cual indica que estan en diferentes rangos.

- Si hicieramos un análisis de regresión lineal el atributo "income", podría influenciar el resultado, pero partiendo de la lógica es claro que este valor sea proporcionalmente diferente.
- Aquí un ejemplo de como podríamos normalizar estos valores.

age	income
0.2	0.2
0.3	0.04
0.4	1

Normalized

- similar value range.
- similar intrinsic influence on analytical model.

Algunas de las técnicas de normalización son las siguientes:

- Escalar variables: Esto divide cada valor por el máximo valor de la columna.
- Minimos y Maximos: Hace los valores en un rango de 0 a 1.
- Puntaje estandar: Cada valor le restamos el promedio de los datos de la columna, y luego dividimos por la desviación estandar. Donde los valores resultantes, usualmente están en un rango de -3 y +3.

Several approaches for normalization:

①

$$x_{new} = \frac{x_{old}}{x_{max}}$$

Simple Feature scaling

②

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

Min-Max

③

$$x_{new} = \frac{x_{old} - \mu}{\sigma}$$

Z-score

Para realizar este proceso con pandas, debemos hacer lo siguiente:

With Pandas:

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
180.0	65.5	52.4
...



length	width	height
0.81	64.1	48.8
0.81	64.1	48.8
0.87	65.5	52.4
...

```
df["length"] = df["length"]/df["length"].max()
```

- De esa manera aplicamos el escalamiento con el maximo. Ahora, si queremos con el minimo, harémos lo siguiente:

With Pandas:

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
180.0	65.5	52.4
...



length	width	height
0.41	64.1	48.8
0.41	64.1	48.8
0.58	65.5	52.4
...

```
df["length"] = (df["length"]-df["length"].min()) /  
                (df["length"].max() - df["length"].min())
```

- Finalmente para el metodo Z-score, aplicaremos la media, la cual devuelve el valor promedio de los datos y std la desviación estandar.

With Pandas:

The diagram illustrates a data transformation using Pandas. On the left, there is a data frame with columns 'length', 'width', and 'height'. The first three rows have values 168.8, 168.8, and 180.0 respectively, while the rest of the row values are '...'. On the right, the same data frame is shown after applying a standardization operation. The first three rows now have values -0.034, -0.034, and 0.039 respectively, while the rest of the row values are '...'. The transformation is indicated by a large black arrow pointing from the left frame to the right frame.

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
180.0	65.5	52.4
...

length	width	height
-0.034	64.1	48.8
-0.034	64.1	48.8
0.039	65.5	52.4
...

```
df["length"] = (df["length"]-df["length"].mean()) / df["length"].std()
```

En general, no todos los métodos encajan perfectamente en las transformaciones y debemos saber cuál utilizar para cada caso.

▼ Binning in Python

- Esta técnica permite agrupar valores en grupos o categorías por ejemplo "age" puede estar entre [0 a 5], [6 a 10] y así sucesivamente. Ocasionalmente esto permite mejorar la precisión de los modelos predictivos.
- Adicionalmente al hacer esto con números pequeños podríamos tener un mejor entendimiento de la distribución de los datos.
- En el siguiente ejemplo podemos categorizar el precio en 3 diferentes bins, "Bajo", "Medio" y "Alto", como lo veremos a continuación:

- **Binning: Grouping of values into "bins"**
- **Converts numeric into categorical variables**
- **Group a set of numerical values into a set of "bins"**
- **"price" is a feature range from 5,000 to 45,500 (in order to have a better representation of price)**

price: 5000, 10000, 12000, 12000, 30000, 31000, 39000, 44000, 44500

bins:

low	Mid	High
-----	-----	------

En python podemos hacerlo de la siguiente manera, teniendo en cuenta:

- Necesitamos 4 numeros para dividirlos, por eso utilizamos linspace para retornar los bins que contienen 4 numeros iguales espaciados en un intervalo, posteriormente creamos una lista "group_names", que tendra los diferentes nombres de las categorias.

The diagram illustrates the process of binning data. On the left, a table shows a single column 'price' with several numerical entries. An arrow points to the right, where another table shows the same data with an additional column 'price-binned'. The 'price-binned' column uses categorical labels ('Low', 'Medium', 'High') to group the original price values.

price	price-binned
13495	Low
16500	Low
18920	Medium
41315	High
5151	Low
6295	Low
...	...

```
bins = np.linspace(min(df["price"]), max(df["price"]), 4)
```

```
group_names = ["Low", "Medium", "High"]
```

Con la siguiente función en pandas podemos cortar el segmento y organizar los valores en sus correspondientes categorias.

```
df["price-binned"] = pd.cut(df["price"], bins, labels=group_names, include_lowest=True )
```

También podemos utilizar histogramas para visualizar mejor los datos que hemos dividido. Con lo cual observamos que la mayoria de los precios de vehiculos son bajos en comparación a los pocos vehiculos que tienen un precio alto.



Turning categorical variables into quantitative variables in Python

- Varios modelos estadisticos, no pueden tomar datos tipo objetos o strings como entrada y para el entrenamiento.
- En el ejemplo del dataset "Fuel type" es un valor categorico con dos valores gas - diesel que son tipo string.
- Podemos añadir nuevas columnas correspondientes a un valor unico para cada elemento, como por ejemplo 1 si tiene o 0 si no, tanto para gas como diesel, de la siguiente manera:

Solution:

- Add dummy variables for each unique category
- Assign 0 or 1 in each category

Car	Fuel	...	gas	diesel
A	gas	...	1	0
B	diesel	...	0	1
C	gas	...	1	0
D	gas	...	1	0

“One-hot encoding”

- Cuando un valor ocurre en la columna original, ponemos su resultado correspondiente en una de las dos nuevas, y la otra la dejamos en 0.
- En pandas podemos utilizar "get_dummies()" para convertir variables categoricas a variables indicadoras.

- Use pandas.get_dummies() method.
- Convert categorical variables to dummy variables (0 or 1)

fuel	
gas	
diesel	
gas	
gas	

gas	diesel
1	0
0	1
1	0
1	0

```
pd.get_dummies(df['fuel'])
```

▼ Lab 2

Question #1:

Based on the example above, replace NaN in "stroke" column with the mean value.

```
[17]: # Write your code below and press Shift+Enter to execute
#Calculate the mean vaule for "stroke" column
avg_stroke = df["stroke"].astype("float").mean(axis = 0)
print("Average of stroke:", avg_stroke)

# replace NaN by mean value in "stroke" column
df["stroke"].replace(np.nan, avg_stroke, inplace = True)
```

Average of stroke: 3.255422885572139

Question #2:

According to the example above, transform mpg to L/100km in the column of "highway-mpg" and change the name of column to "highway-L/100km".

```
[32]: # Write your code below and press Shift+Enter to execute
# transform mpg to L/100km by mathematical operation (235 divided by mpg)
df["highway-mpg"] = 235/df["highway-mpg"]

# rename column name from "highway-mpg" to "highway-L/100km"
df.rename(columns={'highway-mpg':'highway-L/100km'}, inplace=True)

# check your transformed data
df.head()
```

symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-L/100km	price	city-L/100km	
0	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	5000.0	21	8.703704	13495.0	11.190476
1	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	5000.0	21	8.703704	16500.0	11.190476
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	mpfi	2.68	3.47	9.0	154	5000.0	19	9.038462	16500.0	12.368421
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	mpfi	3.19	3.40	10.0	102	5500.0	24	7.833333	13950.0	9.791667
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	mpfi	3.19	3.40	8.0	115	5500.0	18	10.681818	17450.0	13.055556

5 rows x 27 columns

Question #3:

According to the example above, normalize the column "height".

```
[35]: # Write your code below and press Shift+Enter to execute
df['height'] = df['height']/df['height'].max()

# show the scaled columns
df[["length","width","height"]].head()
```

	length	width	height
0	0.811148	0.890278	0.816054
1	0.811148	0.890278	0.816054
2	0.822681	0.909722	0.876254
3	0.848630	0.919444	0.908027
4	0.848630	0.922222	0.908027

Question #4:

Similar to before, create an indicator variable for the column "aspiration"

```
[49]: # Write your code below and press Shift+Enter to execute
# get indicator variables of aspiration and assign it to data frame "dummy_variable_2"
dummy_variable_2 = pd.get_dummies(df['aspiration'])

# change column names for clarity
dummy_variable_2.rename(columns={'std':'aspiration-std', 'turbo': 'aspiration-turbo'}, inplace=True)

# show first 5 instances of data frame "dummy_variable_1"
dummy_variable_2.head()
```

	aspiration-std	aspiration-turbo
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0

Question #5:

Merge the new dataframe to the original dataframe, then drop the column 'aspiration'.

```
[50]: # Write your code below and press Shift+Enter to execute
# merge the new dataframe to the original dataframe
df = pd.concat([df, dummy_variable_2], axis=1)

# drop original column "aspiration" from "df"
df.drop('aspiration', axis = 1, inplace=True)
```

▼ Graded Review Questions 2

Question 1

1/1 point (graded)

Consider the dataframe `df`. What is the result of the following operation:

```
df[ 'symbolling' ] = df[ 'symbolling' ] + 1 ?
```



Every element in the column "symbolling" will increase by one.



Every element in the row "symbolling" will increase by one.



Every element in the dataframe will increase by one.



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 2

1/1 point (graded)

Consider the dataframe `df`. What does the command `df.rename(columns={'a': 'b'})` change about the dataframe `df`?

Renames column "a" of the dataframe to "b".

Renames row "a" to "b".

Nothing. You must set the parameter "inplace = True".



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 3

1/1 point (graded)

Consider the dataframe "df". What is the result of the following operation

```
df['price'] = df['price'].astype(int) ?
```

Convert or cast the row 'price' to an integer value.

Convert or cast the column 'price' to an integer value.

Convert or cast the entire dataframe to an integer value.



[Save](#)

[Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 4

0/1 point (graded)

Consider the column of the dataframe `df['a']`. The column has been standardized. What is the standard deviation of the values as a result of applying the following operation: `df['a'].std()`?

1 ✓

0

3

Explanation

The Z-score has unit standard deviation as it is simply the original data divided by its own standard deviation.

[Show answer](#)

[Submit](#)

You have used 2 of 2 attempts

Question 5 a)

1/1 point (graded)

Consider the column of the dataframe, df['Fuel'], with two values: 'gas' and 'diesel'. What will be the name of the new columns pd.get_dummies(df['Fuel']) ?

1 and 0

Just 'diesel'

Just 'gas'

'gas' and 'diesel'



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

 Correct (1/1 point)

Question 5 b)

1/1 point (graded)

What are the values of the new columns from part 5a)?

1 and 0

Just 'diesel'

Just 'gas'

'gas' and 'diesel'



[Save](#) | [Show a](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

▼ Module 3 - Exploratory Data Analysis

▼ Exploratory Data Analysis

En este modulo se cubren los siguientes temas:

- Resumir caracteristicas principales de los datos.
- Obtener un mejor entendimiento de los datos.
- Encontrar relaciones entre variables.
- Extraer importantes variables.

- Se responderá la pregunta ¿Cuáles son las características que mayor impacto tienen en el precio del vehículo?
- Se aprenderá sobre estadística descriptiva, agrupar datos mediante "group by", ANOVA (análisis de varianza, correlación y observación)

▼ Descriptive Statistics

- Nos permite describir características básicas de los datos.
- Permite generar pequeños resúmenes sobre muestras de datos.
- en pandas con "df.describe()" tenemos un resumen estadístico de las variables numéricas de la siguiente manera:

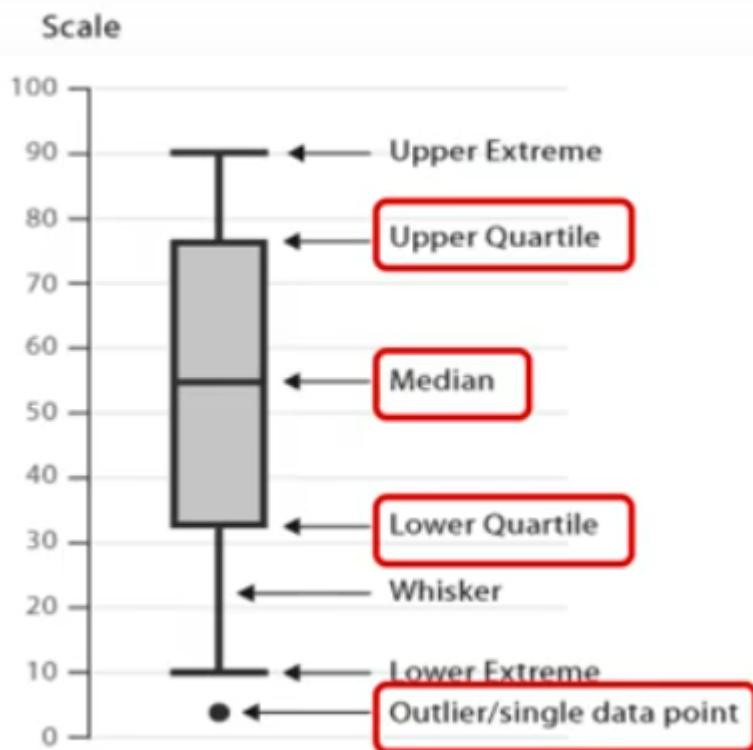
```
df.describe()
```

	Unnamed: 0	symboling	normalized- losses	wheel- base	length	width	height	curb-weight	engine- size	bore	stroke
count	201.000000	201.000000	164.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000
mean	100.000000	0.840796	122.000000	98.797015	174.200995	65.889055	53.766667	2555.666667	126.875622	3.319154	3.256766
std	58.167861	1.254802	35.442168	6.066366	12.322175	2.101471	2.447822	517.296727	41.546834	0.280130	0.316049
min	0.000000	-2.000000	65.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000
25%	50.000000	0.000000	NaN	94.500000	166.800000	64.100000	52.000000	2169.000000	98.000000	3.150000	3.110000
50%	100.000000	1.000000	NaN	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000
75%	150.000000	2.000000	NaN	102.400000	183.500000	66.600000	55.500000	2926.000000	141.000000	3.580000	3.410000
max	200.000000	3.000000	256.000000	120.900000	208.100000	72.000000	59.800000	4066.000000	326.000000	3.940000	4.170000

- Para sumar las variables categóricas podemos utilizar "value_counts()" lo cual aplica para grupos o que tengan valores discretos, como por ejemplo en nuestro dataset tenemos "forward-wheel drive", "rear-wheel" y una forma de resumirlas sería de esta forma:

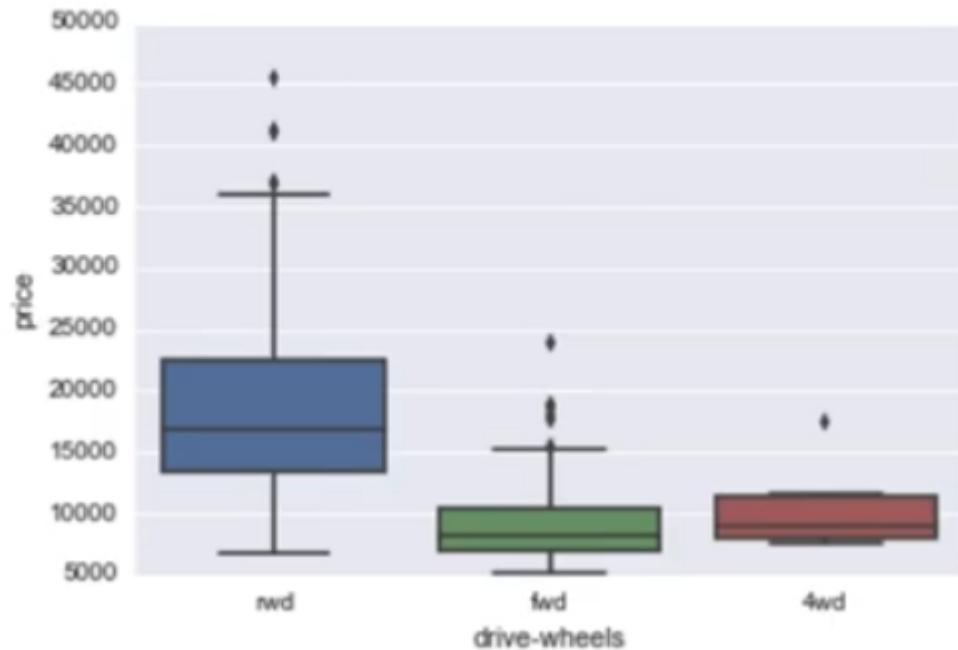
```
drive_wheels_counts=df[ "drive-wheels" ].value_counts()
```

- Mediante los diagramas de caja o Boxplot, podemos visualizar las distribuciones de los datos



- Si queremos definir un boxplot en Python podemos hacerlo de la siguiente manera utilizando la libreria "Seaborn".

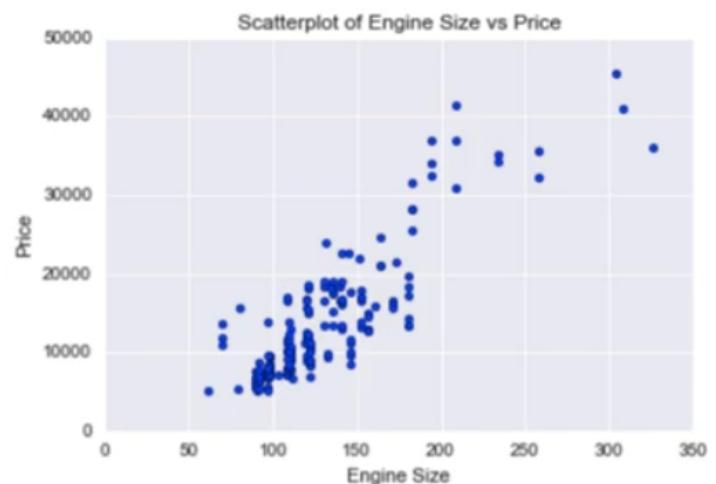
```
sns.boxplot(x= "drive-wheels", y= "price", data=df)
```



- Podemos observar las diferentes distribuciones entre las categorías que tenemos.
- Algunas veces vemos variables continuas en nuestros datos, estos puntos son representados como numeros, que son contenidos en algun rango.
- Si quisieramos saber si el tamaño del motor y el precio tienen alguna relación para predecir el precio, podemos utilizar el siguiente grafico "Scatter plot", donde la variable predictora o independiente esta en el eje X que para este ejemplo es el tamaño del motor, mientras que la variable objetivo es lo que intentamos predecir, para este caso es el precio y esta representada en el eje Y.

```
y=df[ "engine-size"]
x=df[ "price"]
plt.scatter(x,y)

plt.title("Scatterplot of Engine Size vs Price")
plt.xlabel("Engine Size")
plt.xlabel("Price")
```



- Es muy importante darle nombre a los ejes para entender mejor la relación.
- Lo que podemos observar de este gráfico es que entre más grande el motor el precio aumenta, lo cual es un indicador inicial de que tenemos una relación lineal positiva entre estas dos variables.

▼ GroupBy in Python

- Podríamos asumir que hay una diferencia entre los diferentes tipos de sistema de conducción o respecto la tracción con el precio, podríamos agruparlos en diferentes categorías y comparar los resultados entre sí.
- Con el método "dataframe.groupby()" podemos agrupar variables categóricas, de manera individual o múltiples, de esta forma.

```
df_test = df[['drive-wheels', 'body-style', 'price']]  
df_grp = df_test.groupby(['drive-wheels', 'body-style'], as_index=False).mean()  
df_grp
```



	drive-wheels	body-style	price
0	4wd	hatchback	7603.000000
1	4wd	sedan	12647.333333
2	4wd	wagon	9095.750000
3	fwd	convertible	11595.000000
4	fwd	hardtop	8249.000000
5	fwd	hatchback	8396.387755
6	fwd	sedan	9811.800000
7	fwd	wagon	9997.333333
8	rwd	convertible	23949.600000
9	rwd	hardtop	24202.714286
10	rwd	hatchback	14337.777778
11	rwd	sedan	21711.833333
12	rwd	wagon	16994.222222



- Elegimos tres columnas de datos que nos interesen.
- Luego agrupamos los datos reducidos de acuerdo a "drive-wheels" y "body style".
- Si estamos interesados en saber como el precio promedio difiere alrededor de los diferentes datos podemos añadir la media de cada grupo y añadirla.
- Podemos ver que de acuerdo a los datos "rear wheel drive convertibles" y "rear wheel drive hardtops" tienen los valores mas altos, mientras que "four wheel drive hatchbacks" tienen el valor menor.

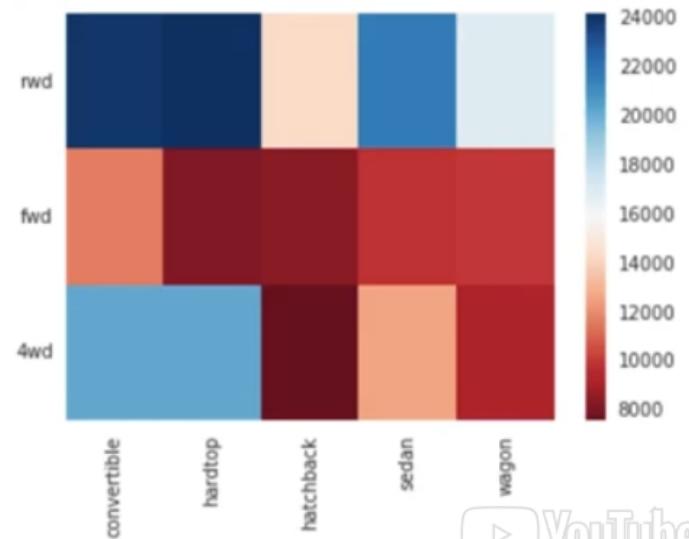
- Para una mejor visualización de la tabla podemos utilizar el metodo "pivot", que permite mostrar una variable junto con sus columnas y los otros valores son mostrados como filas.

```
df_pivot = df_grp.pivot(index= 'drive-wheels', columns='body-style')
```

	price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	20239.229524	20239.229524	7603.000000	12647.333333	9095.750000
fwd	11595.000000	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.600000	24202.714286	14337.777778	21711.833333	16994.222222

- Logrando tener una tabla rectangular, muy similar a como lo haríamos en excel.
 - Otra alternativa para representar nuestros datos sería utilizando mapas de calor o "Heatmap", que es un grafico el cual le asigna un color a los datos dependiendo la intensidad de estos, es una gran opción para mostrar la variable objetivo sobre multiples variables a traves.
- Permitiendo una clara visualización de la relación que estas tienen.

```
plt.pcolor(df_pivot, cmap= 'RdBu')
plt.colorbar()
plt.show()
```

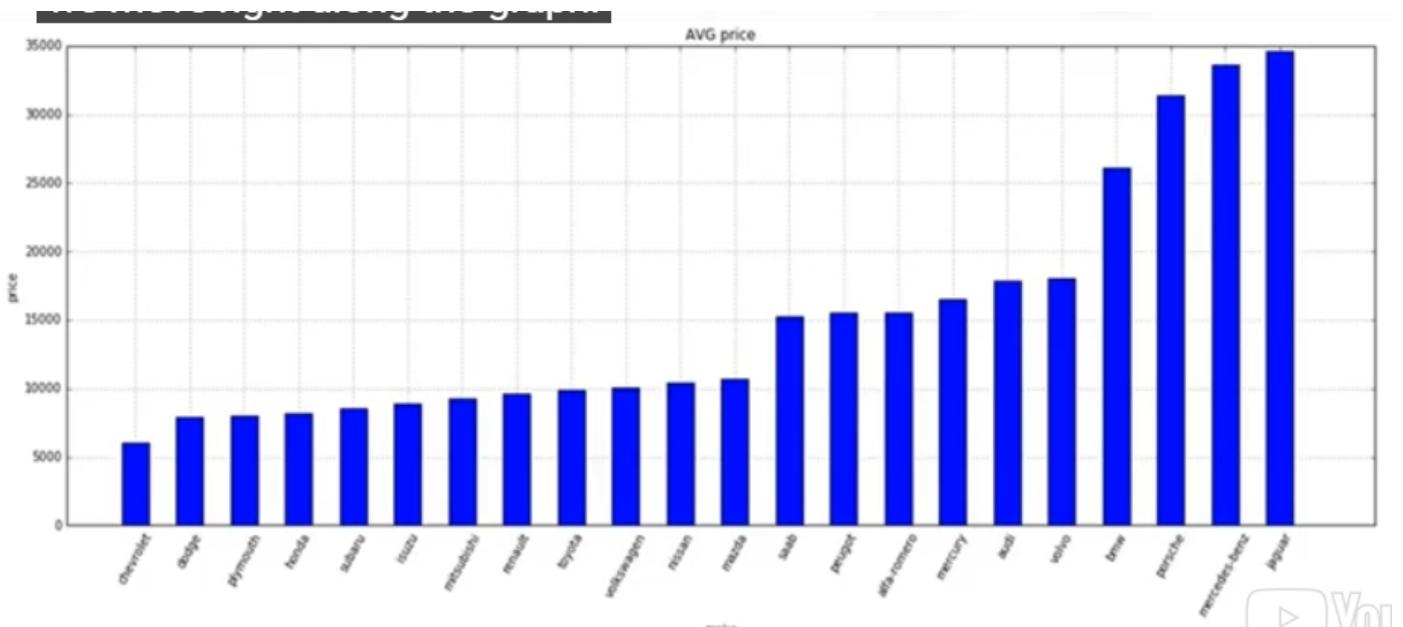


- Podemos observar que valores azules tienen valores mas altos, mientras que los rojos mas bajos.

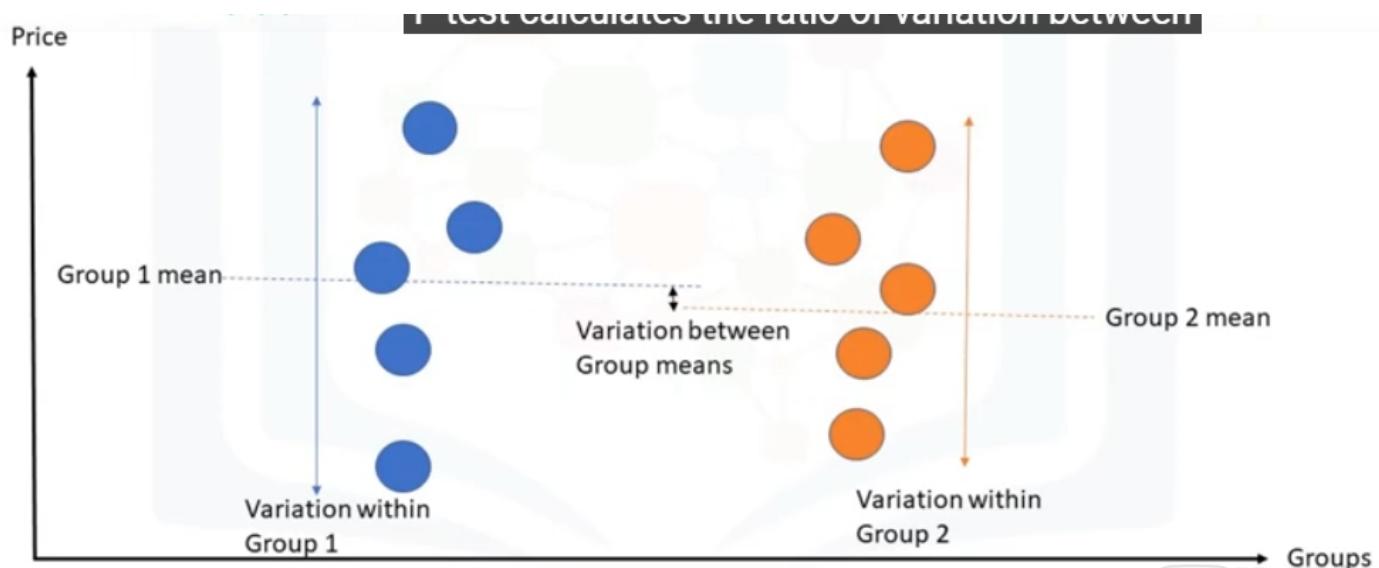
Analysis of Variance ANOVA

- Asumamos que queremos analizar variables categoricas y observar la relación que tienen.

- Por ejemplo si tomamos nuestro dataset podemos preguntarnos ¿Cuantas categorias diferentes de una caracteristica tienen impacto en el precio?
- Podemos ver una tendencia al incremento de precios a lo largo del grafico.



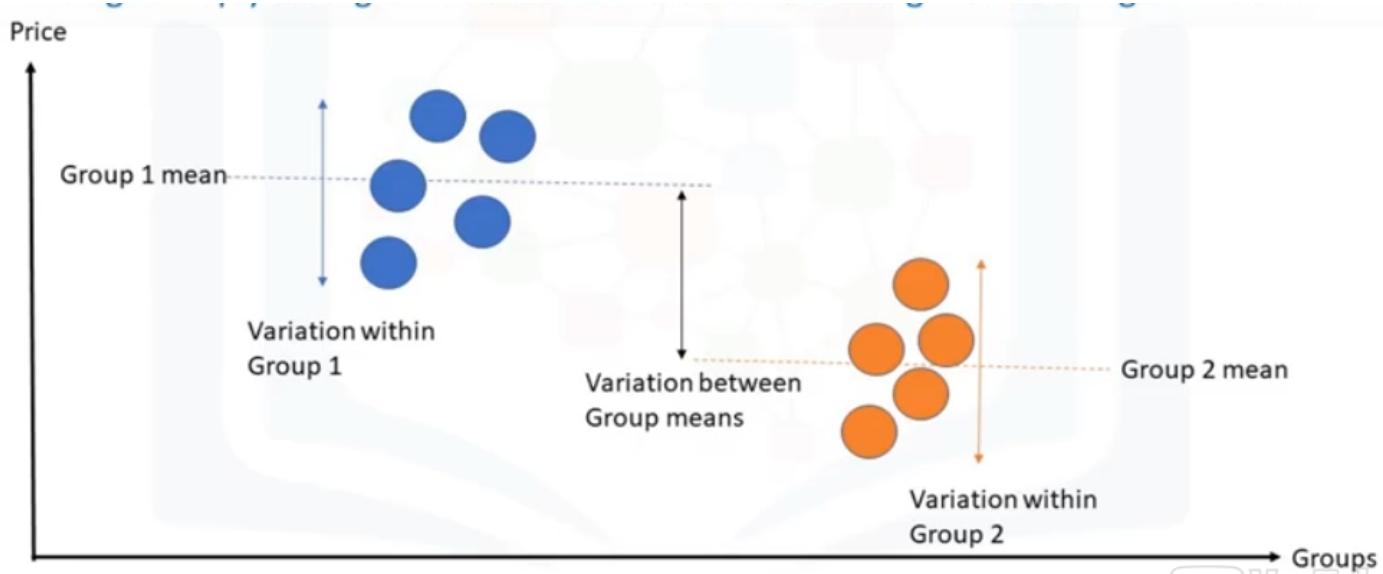
- Pero, ¿Que categoria tiene el mayor/menor impacto en la predicción del precio del vehiculo?
- Para realizar este analisis recurrimos al metodo "ANOVA", el cual es un analisis de la varianza, que permite encontrar la correlaccion entre diferentes grupos de una variable categorica.
- Podemos hacer un ejemplo con las marcas "Subaru" y "Honda".
- Con ANOVA obtenemos dos valores **F-Test score**: Variación entre la media del grupo de muestra dividida la variación del conjunto de muestra. mientras que **p-value**: Grado de certeza.



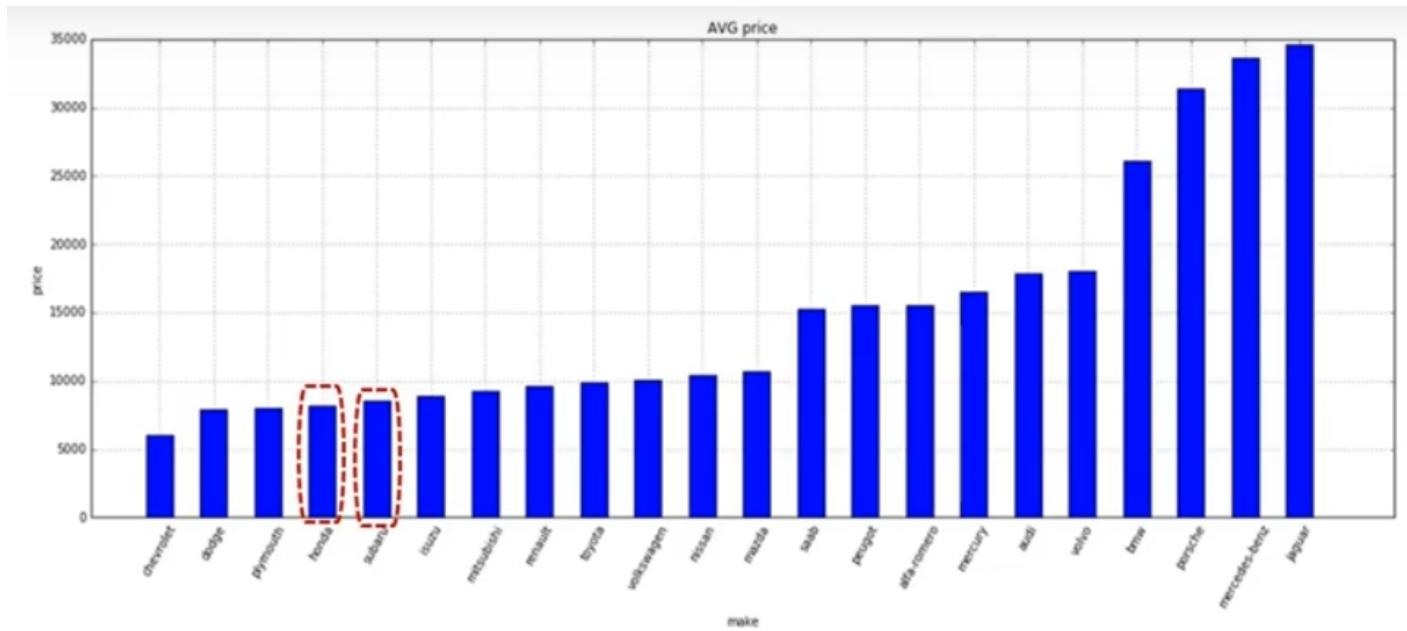
- En el siguiente grafico observamos la variación entre cada uno de los grupos de muestra, donde vemos que el F-test es pequeño, debido a que la variación de los precios en cada

grupo es mayor que las diferencias del promedio de los valores en cada grupo.

- Asumiendo que el primer grupo es Honda y el segundo es Subaru, donde la correlación del precio y la variable objetivo es débil.



- En este caso observamos el valor de F-test score es mayor, asumiendo que el primer grupo es Jaguar y el segundo Honda, la correlación es fuerte.



- Volviendo al primer grafico nos damos cuenta que el F-score entre "Honda" y "Subaru" es una pequeña diferencia, a comparación de "honda" y "Jaguar", debido a que las diferencias de precio son muy significantes.
- A continuación haremos el proceso con Python, de la siguiente manera:

• ANOVA between “Honda” and “Subaru”

```
df_anova=df[["make", "price"]]  
grouped_anova=df_anova.groupby([["make"]])
```

```
anova_results_1=stats.f_oneway(grouped_anova.get_group("honda") ["price"], grouped_anova.get_group("subaru") ["price"])  
ANOVA results: F=0.19744031275, p=F_onewayResult(statistic=0.1974430127), pvalue=0.660947824
```

• ANOVA between “Honda” and “Jaguar”

```
anova_results_1=stats.f_oneway(grouped_anova.get_group("honda") ["price"], grouped_anova.get_group("jaguar") ["price"])  
ANOVA results: F=0.19744031275, p=F_onewayResult(statistic=400.92589), pvalue=1.05861
```

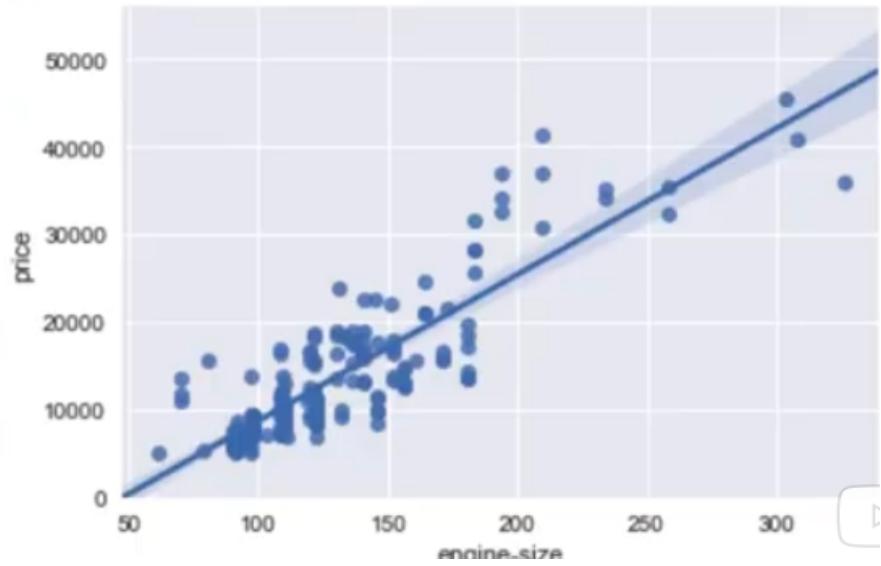
- Observamos las diferencias son equivalentes a como las veiamos graficamente. De esta forma podemos hacer test para variables categoricas y observar su relación.

▼ Correlation

- Nos permite medir como se extienden diferentes variables y estas son interdependientes, o como el efecto en una afecta la otra.
- Por ejemplo fumar esta correlacionado con el cancer de pulmon, es decir tendras mas probabilidad de tener cancer de pulmon si fumas.
- O la correlación entre sombrillas y lluvia, entre mas llueva mas personas utilizaran sombrillas.
- Para este ejemplo veremos la correlación entre el tamaño del motor y el precio.

- Correlation between two features (engine-size and price).

```
sns.regplot(x="engine-size", y="prices", data=df)  
plt.ylim(0, )
```



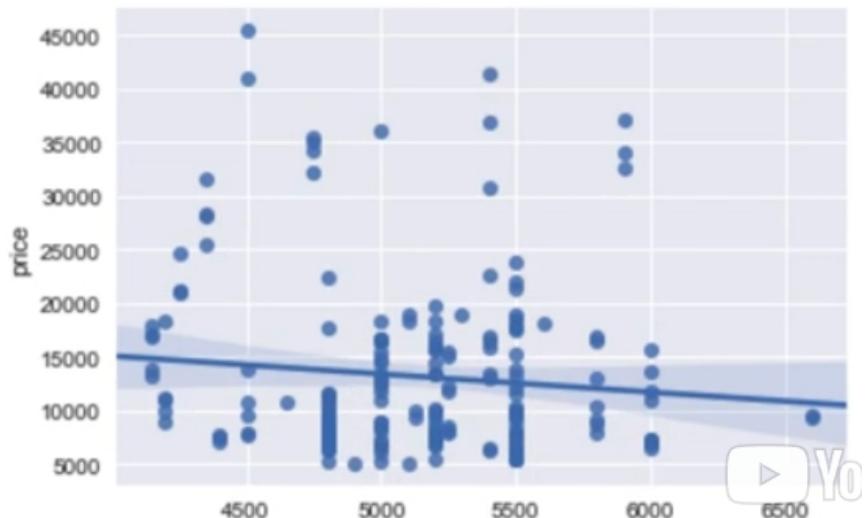
- Utilizamos un scatterplot y agregamos una linea la cual conocemos como regresión lineal, que indica la relación entre estas dos variables.
- El objetivo principal es entender si el tamaño del motor tiene impacto en el precio, lo cual indica una relación positiva lineal.
- No olvidemos que para hacer el scatter plot utilizamos el metodo "regplot()" de seaborn.

- Correlation between two features (highway-mpg and price)

- En este otro ejemplo observaremos millas por galon vs el precio. entre mayor es el precio menor son las millas por galon, lo cual es una correlación negativa.

- Weak correlation between two features (peak-rpm and price).

```
sns.regplot(x="peak-rpm", y="prices", data=df)  
plt.ylim(0, )
```



- Para este ultimo ejemplo donde tomamos el pico de revoluciones vs el precio, obtenemos una correlación debil. Lo cual indica que esta variable no seria un buen indicador para predecir el precio del vehiculo.

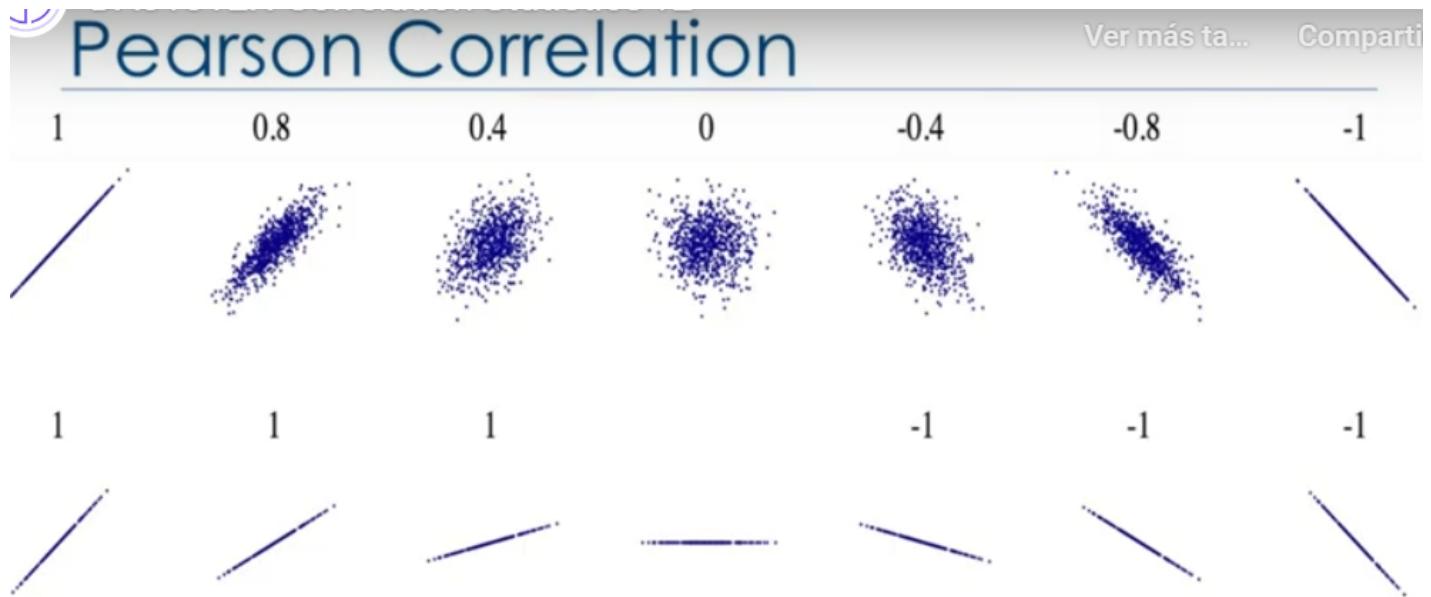
▼ Correlation - Statistics

- Para medir la fuerza de la correlación entre dos variables numericas utilizamos el metodo de correlación de Pearson, el cual nos devuelve dos valores el coeficiente de correlación y p-value.
- Para interpretar estos valores +1 correlación positiva, -1 correlación negativa, 0 no correlación.
- Mientras que con P-value:

- **Correlation coefficient**

- **P-value**

- una correlación fuerte es cuando el coeficiente esta entre -1 y 1 y p-value menor que 0.001.
- Aquí observamos datos con diferentes valores de correlación.

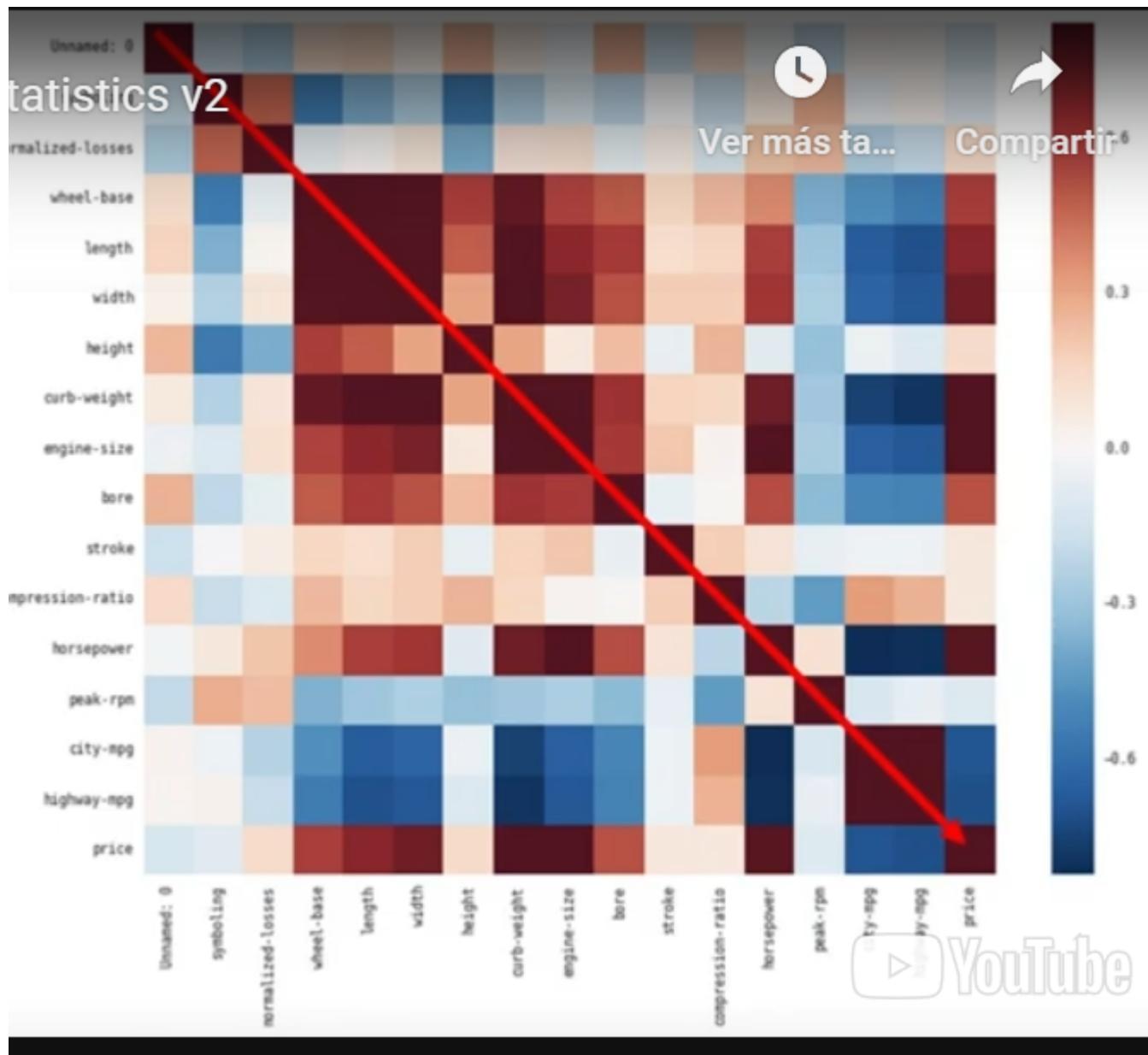


- Para este ejemplo veremos la correlación entre los caballos de fuerza y el precio, donde observamos una correlación positiva entre estas.

```
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
```

- **Pearson correlation:** 0.81

- **P-value :** 9.35 e-48



- Si utilizamos un mapa de calor con todas las variables, podremos observar la correlación que estas tienen respecto a otras, y como estan relacionadas con el precio.

▼ Lab 3

Question #1:

What is the data type of the column "peak-rpm"?

```
[10]: # Write your code below and press Shift+Enter to execute  
df["peak-rpm"].dtypes
```

```
[10]: dtype('float64')
```

Question #2:

Find the correlation between the following columns: bore, stroke, compression-ratio, and horsepower.

Hint: if you would like to select those columns, use the following syntax: df[['bore','stroke','compression-ratio','horsepower']]

```
[12]: # Write your code below and press Shift+Enter to execute  
df[['bore','stroke','compression-ratio','horsepower']].corr()
```

	bore	stroke	compression-ratio	horsepower
bore	1.000000	-0.055390	0.001263	0.566936
stroke	-0.055390	1.000000	0.187923	0.098462
compression-ratio	0.001263	0.187923	1.000000	-0.214514
horsepower	0.566936	0.098462	-0.214514	1.000000

Question 3 a):

Find the correlation between x="stroke" and y="price".

Hint: if you would like to select those columns, use the following syntax: df[["stroke","price"]].

```
[19]: # Write your code below and press Shift+Enter to execute  
df[["stroke","price"]].corr()
```

```
[19]:      stroke    price  
stroke  1.00000  0.08231  
price   0.08231  1.00000
```

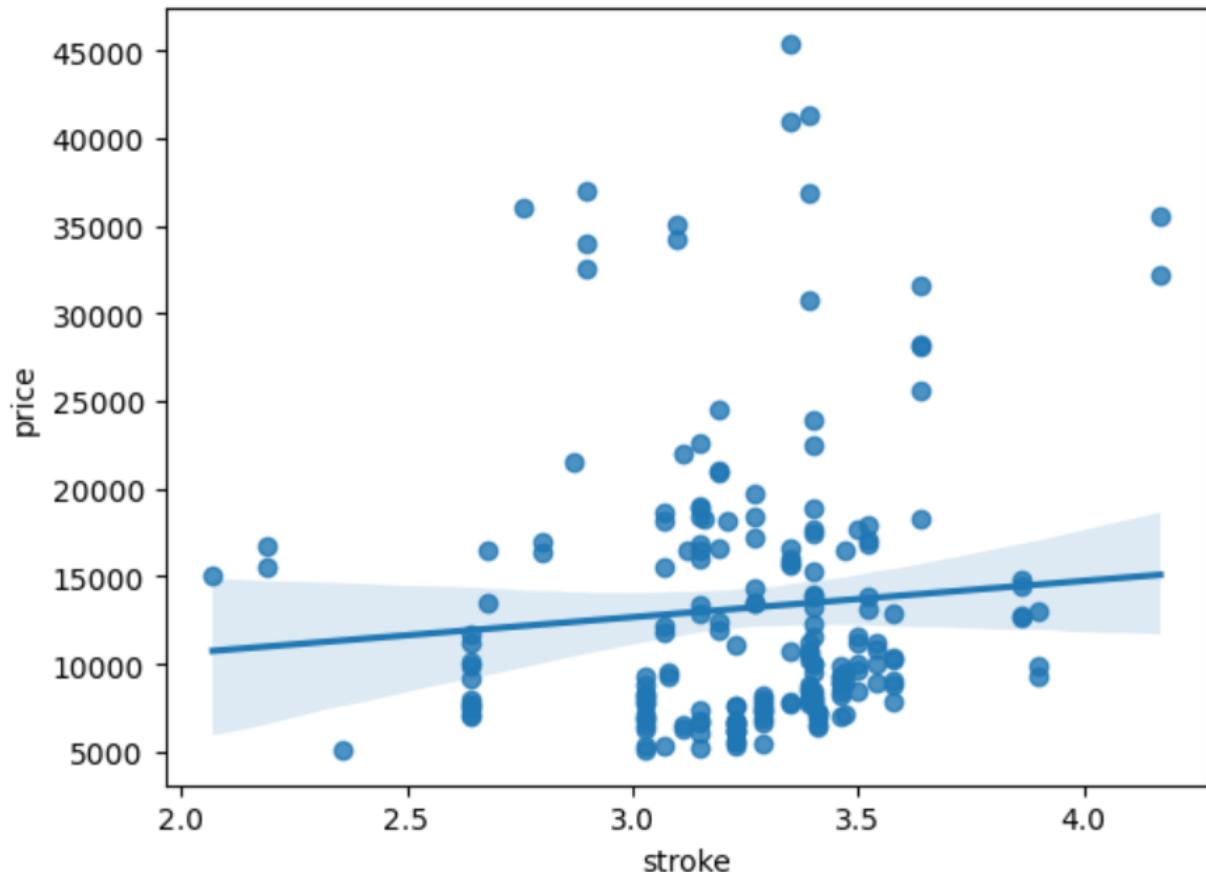
Question 3 b):

Given the correlation results between "price" and "stroke", do you expect a linear relationship?

Verify your results using the function "regplot()".

```
[20]: # Write your code below and press Shift+Enter to execute  
sns.regplot(x="stroke", y="price", data=df)
```

```
[20]: <AxesSubplot:xlabel='stroke', ylabel='price'>
```



Question 4:

Use the "groupby" function to find the average "price" of each car based on "body-style".

```
[37]: # Write your code below and press Shift+Enter to execute
# grouping results
df_gptest2 = df[['body-style','price']]
grouped_test_bodystyle = df_gptest2.groupby(['body-style'],as_index= False).mean()
grouped_test_bodystyle
```

```
[37]:   body-style      price
0   convertible  21890.500000
1     hardtop   22208.500000
2   hatchback    9957.441176
3      sedan    14459.755319
4      wagon   12371.960000
```

▼ Graded Review Questions 3

Question 1

1/1 point (graded)

Consider the dataframe "df". What method provides the summary statistics?

df.describe()

df.head()

df.tail()



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 2

1/1 point (graded)

Consider the following dataframe:

```
df_test = df[['body-style', 'price']]
```

The following operation is applied:

```
df_grp = df_test.groupby(['body-style'], as_index=False).mean()
```

What are resulting values of `df_grp['price']` ?

The average price for each body style.

The average price.

The average body style.



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 3

1/1 point (graded)

Correlation implies causation:

False

True



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 4

1/1 point (graded)

What is the minimum possible value of Pearson's Correlation?

 1 -100 -1

[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 5

1/1 point (graded)

What is the Pearson correlation between variables X and Y if X=Y:

 -1 1 0

[Show answer](#)

[Submit](#)

You have used 2 of 2 attempts

Correct (1/1 point)

▼ Module 4 - Model Development

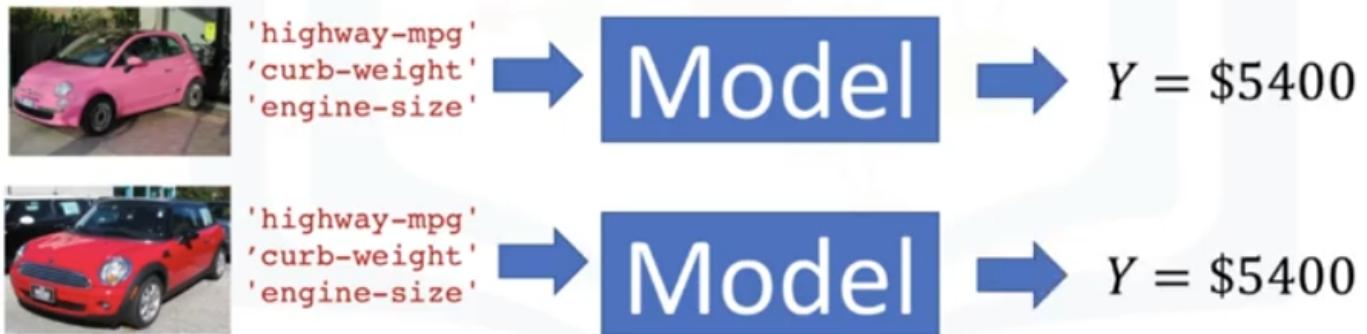
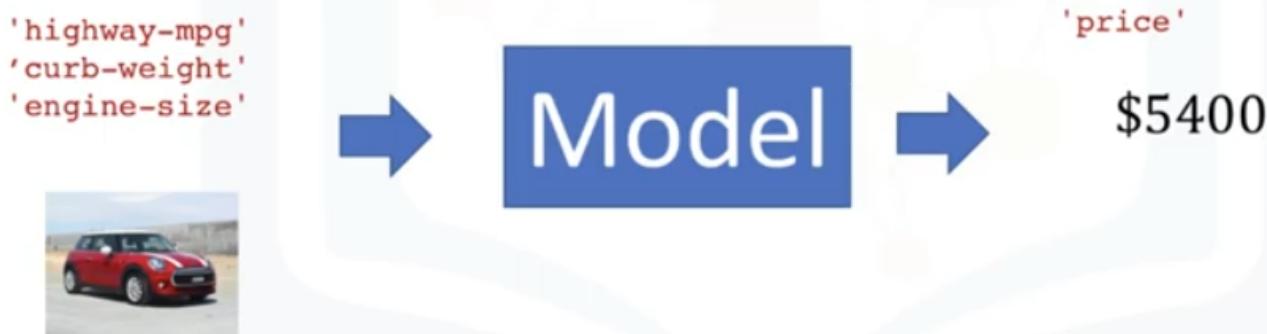
▼ Model Development

En este modulo aprenderemos sobre:

- Simple y multiple regresión lineal.
- Evaluación del modelo usando visualización.

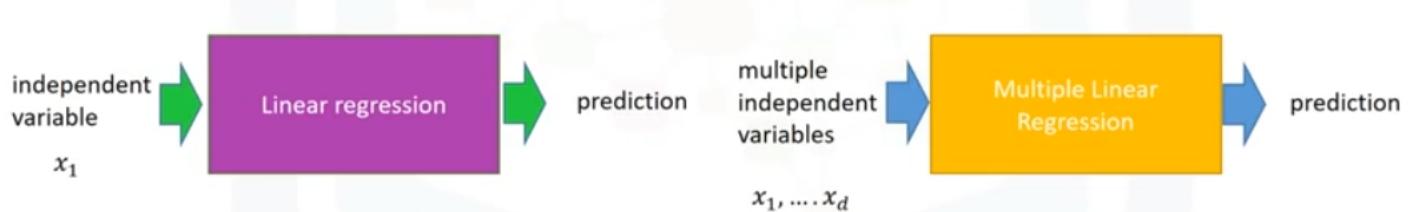
- Regresión polinomica y pipelines.
- R-squared y MSE para evaluación.
- Responderemos a la pregunta **¿Como determinar un valor justo para un vehiculo usado?**

- Usually the more relevant data you have the more accurate your model is



- Debemos tener precaución en agregar los datos suficientes para describir apropiadamente nuestros datos, como en este caso donde si no agregamos la caracteristica del color, nuestro modelo va a predecir el mismo resultado.

▼ Linear Regression and Multiple Linear Regression



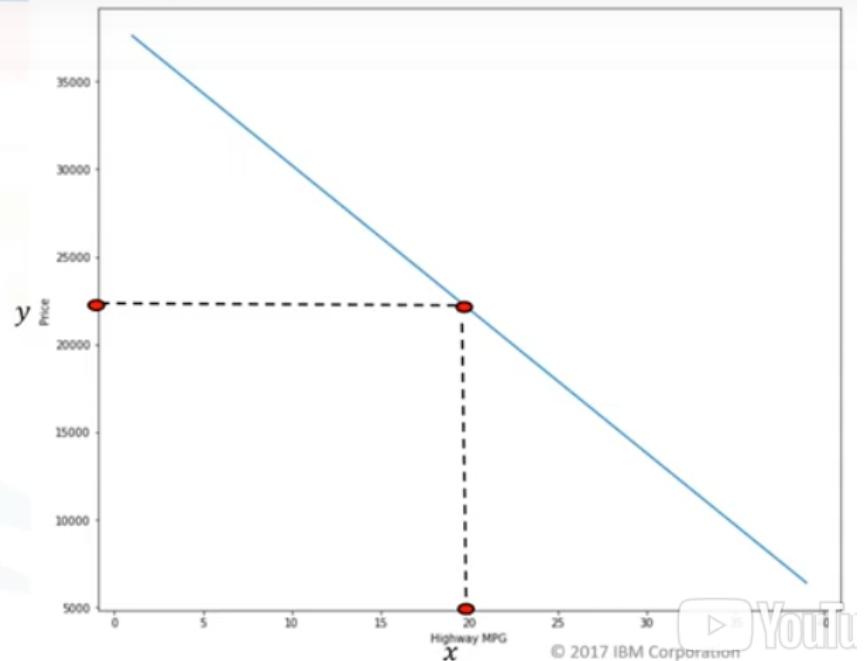
- Regresión lineal se refiere a una variable independiente para hacer una predicción.
- Regresión lineal multiple se refiere a varias variables independientes para hacer predicciones.
- Estos métodos nos ayudan a entender la relación entre variables la predictora (independiente x), y el objetivo (dependiente y).

1. The predictor (independent) variable - x
2. The target (dependent) variable - y

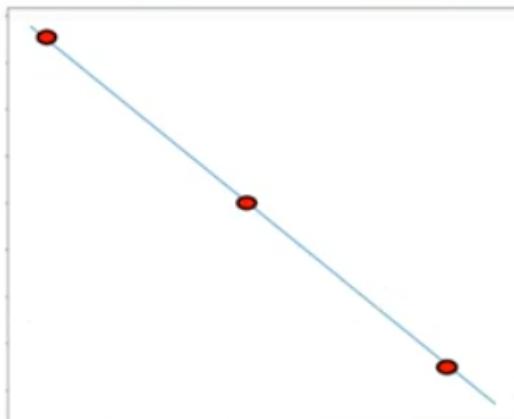
$$y = b_0 + b_1 x$$

- b_0 : the intercept
- b_1 : the slope

$$\begin{aligned}
 y &= 38423 - 821x \\
 &= 38423 - 821(20) \\
 &= 22\,003
 \end{aligned}$$

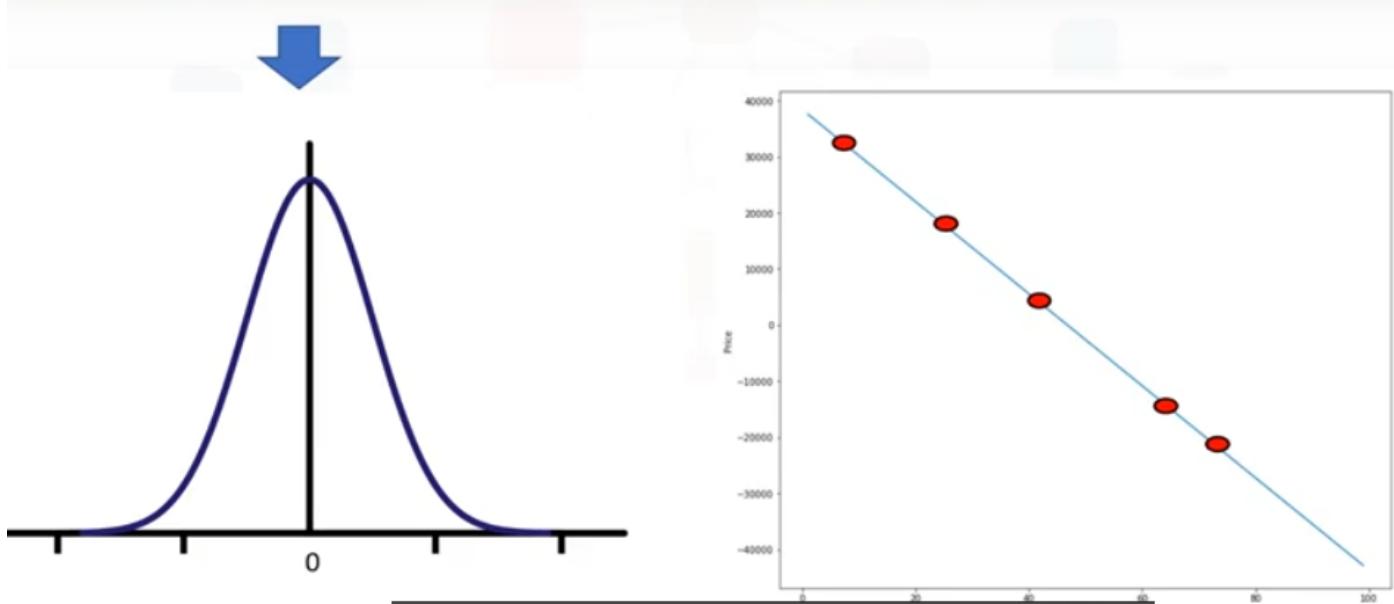


- Para este caso si asumimos que las millas por galon es 20 al reemplazar en la ecuación tendremos una predicción de \$20.000 en el precio.
- Para determinar la linea tomamos los puntos de datos y los utilizaremos para entrenar el modelo, esos resultados son los parametros, los cuales usualmente guardamos en dos dataframe o arrays de Numpy.
- El objetivo ira al array Y, mientras que la variable dependiente en array x.

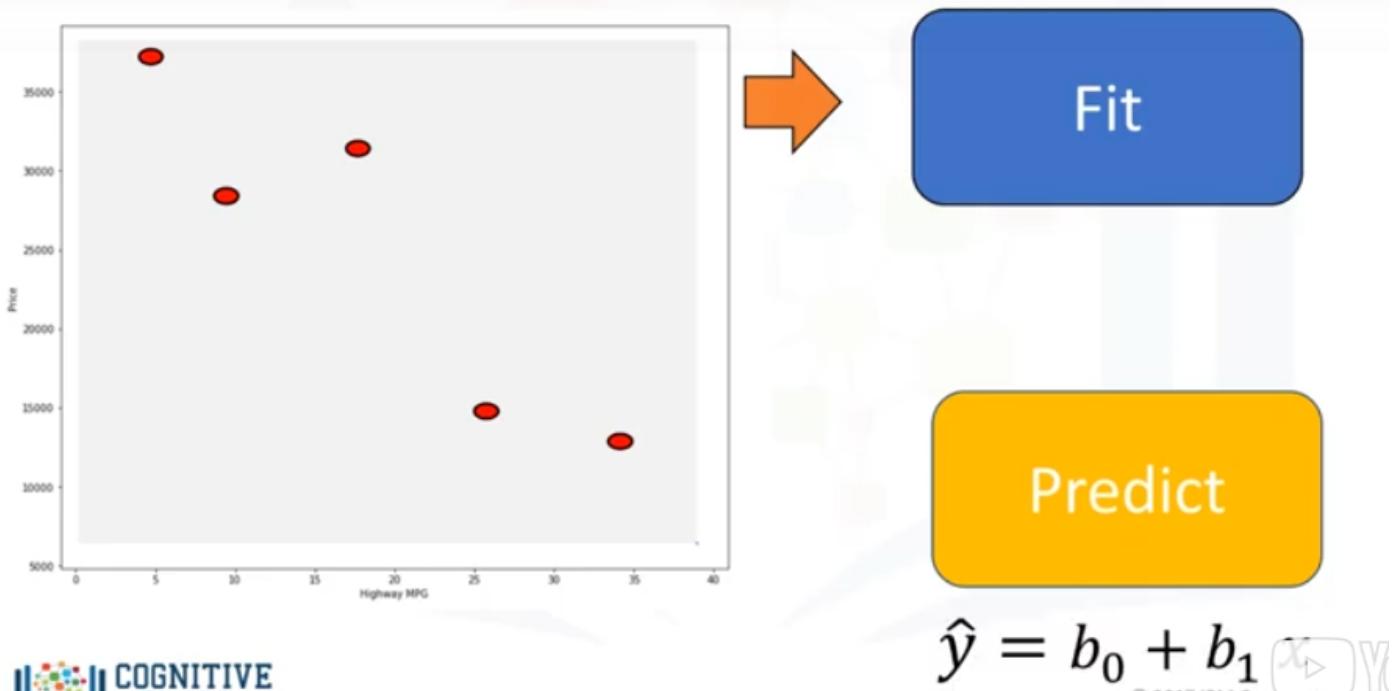


$$X = \begin{bmatrix} 0 \\ 20 \\ 40 \end{bmatrix} \quad Y = \begin{bmatrix} 38423 \\ 22003 \\ 5583 \end{bmatrix}$$

- En cada dataframe o array varios factores influencian cuanto una persona pagaría por un vehiculo, por ejemplo con la marca, modelo.



- El ruido como se muestra en la imagen de la distribución, el eje vertical muestra el valor añadido y el eje horizontal muestra la probabilidad de que el valor sea añadido, positivo o negativo, algunas veces son muchos valores pero usualmente son agregados cerca de 0, podemos resumirlo de la siguiente manera:



- De esta forma podemos predecir valores que no se han visto.
- Cuando encima de la variable hay un sombrero indica que sera el valor que se va a estimar.
- Para nuestro ejemplo no tenemos un vehiculo con 20 millas por galon, y podemos usar nuestro modelo para predecir el precio de este vehiculo, pero no olvidemos que el modelo no

siempre es correcto, lo cual podemos hacer comparando el valor predecido vs el valor actual.

- Por ejemplo tenemos una muestra para 10 millas por galon, pero los valores predecidos no concuerdan con el valor actual, esto puede ocurrir por el ruido, pero hay otras razones, para hacerlo en Python tenemos lo siguiente:

- X :Predictor variable
- Y: Target variable

1. Import linear_model from scikit-learn

```
from sklearn.linear_model import LinearRegression
```

2. Create a Linear Regression Object using the constructor :

```
lm=LinearRegression()
```

- importamos la libreria y definimos la regresión lineal con el constructor.

• We define the predictor variable and target variable

```
X = df[['highway-mpg']]  
Y = df['price']
```

• Then use lm.fit (X, Y) to fit the model , i.e fine the parameters b_0 and b_1

```
lm.fit(X, Y)
```

• We can obtain a prediction

```
Yhat=lm.predict(X)
```

The output is an array, the array has the same number of samples as the input X

Yhat	X
2	5
:	
3	4



- La salida sera un array que tiene el mismo numero de muestras que la entrada X, el intercepto b0 es un atributo del objeto lm, mientras que la pendiente b1 es un atributo también de lm.
- La relación entre el precio y highwayMPG es dada por la siguiente ecuación:

- We can view the intercept (b_0): `lm.intercept_`
38423.305858

- We can also view the slope (b_1): `lm.coef_`
-821.73337832

- The Relationship between Price and Highway MPG is given by:
- Price = 38423.31 - 821.73 * highway-mpg

- La regresión lineal multiple es usada para explicar la relación entre un objetivo continuo (Y) y dos o mas predictores (X), por ejemplo:

$$\hat{Y} = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4$$

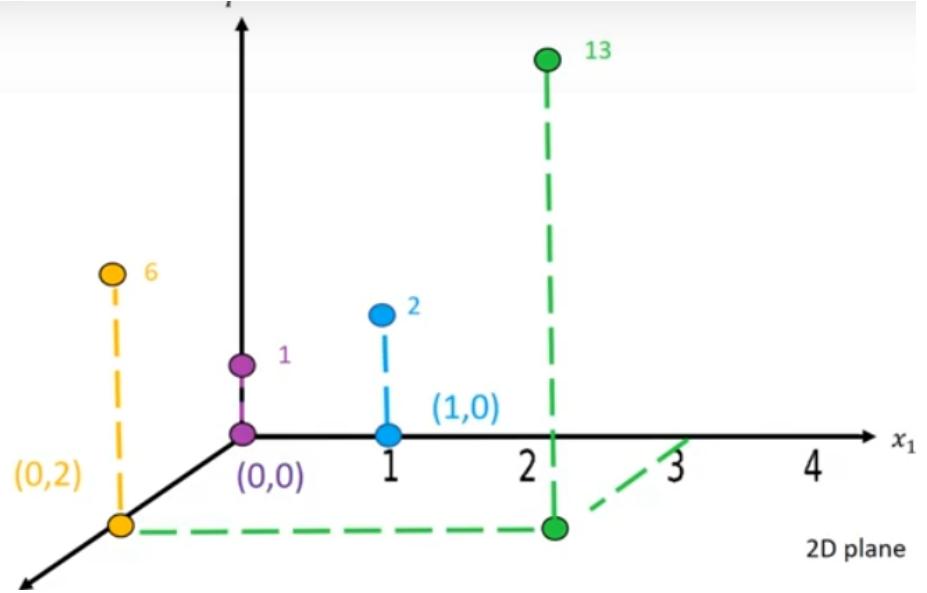
- b_0 : intercept (X=0)
- b_1 : the coefficient or parameter of x_1
- b_2 : the coefficient of parameter x_2 and so on..

- Para este ejemplo visualizamos la siguiente regresión multiple, donde en la dirección vertical con el peso proporcional al valor de yhat que toma.

$$\hat{Y} = 1 + 2x_1 + 3x_2$$

n	x_1	x_2
1	0	0
2	0	2
3	1	0
4	3	2

	\hat{Y}
1	1
2	6
3	2
4	13



- Por otro lado, podemos extraer las 4 variables predictoras y almacenarlas en la variable Z, el modelo de entrenamiento utiliza las variables dependientes y los objetivos.

1. We can extract the for 4 predictor variables and store them in the variable Z

```
Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

2. Then train the model as before:

```
lm.fit(Z, df['price'])
```

3. We can also obtain a prediction

```
Yhat=lm.predict(X)
```

x_1	x_2	x_3	x_4
3	5	-4	3
:	:	:	:
2	4	2	-4

- En este caso la entrada es un dataframe con 4 columnas, el numero de filas corresponde al numero de muestras.
- La salida es un array con el mismo numero de elementos que el numero de muestras, el intercepto es un atributo del objeto y los coeficientes tambien, es bueno visualizar esta ecuación reemplazando los valores dependientes de las variables con sus nombres actuales.

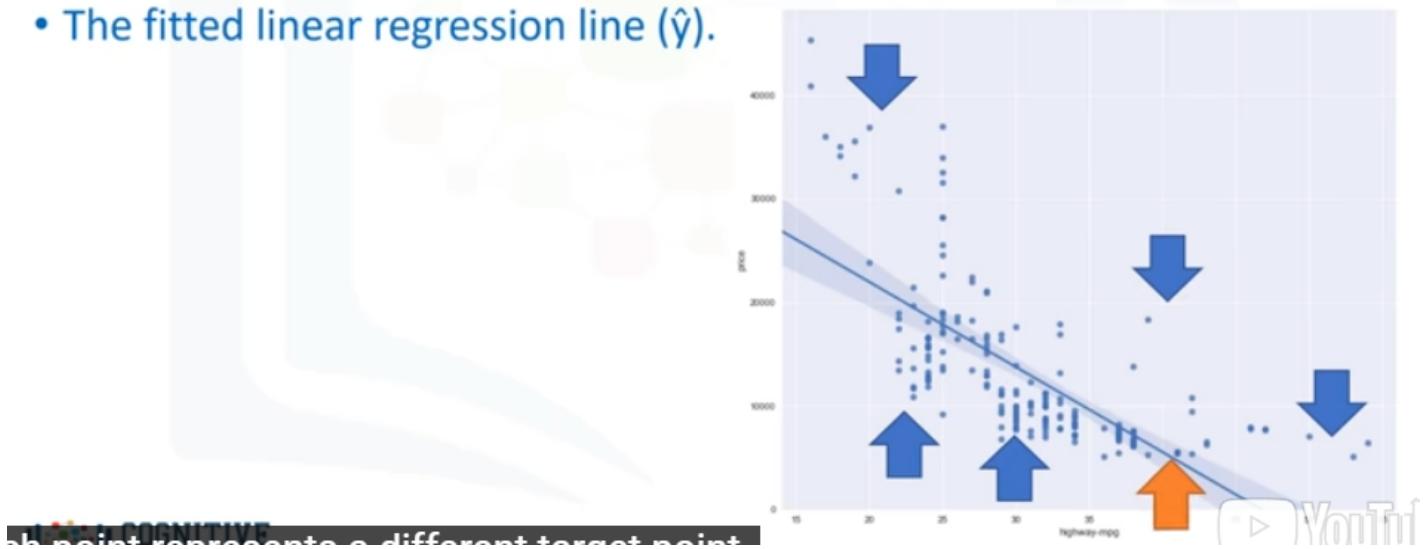
▼ Model Evaluation using Visualization

La grafica de regresión nos permite un estimado de:

- Relación entre dos variables.
- Fuerza de correlación.
- Dirección de la relación (positiva-negativa)

Regression Plot shows us a combination of:

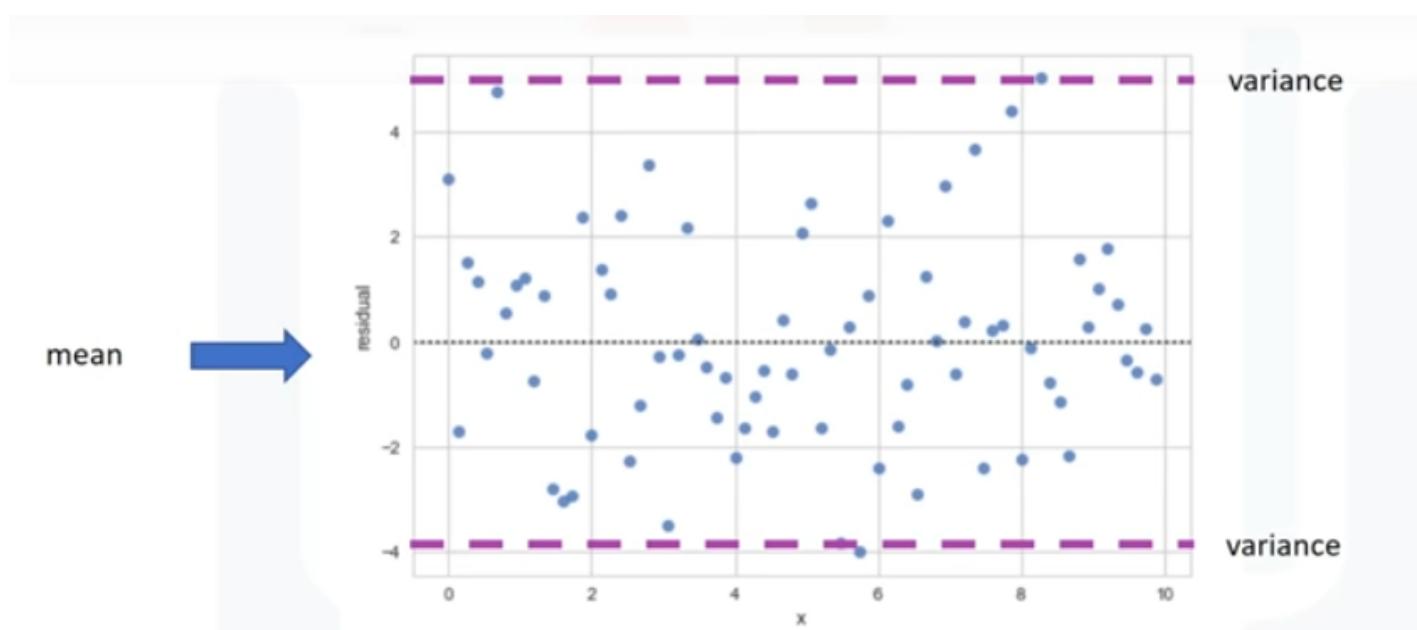
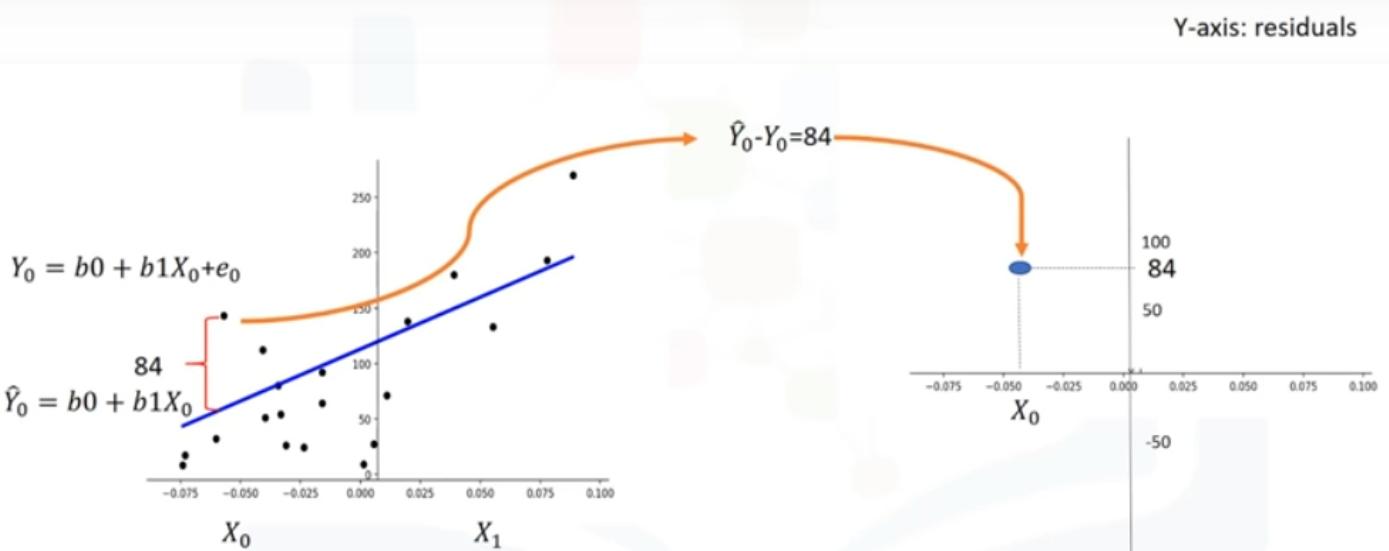
- The scatterplot: where each point represents a different y
- The fitted linear regression line (\hat{y}).



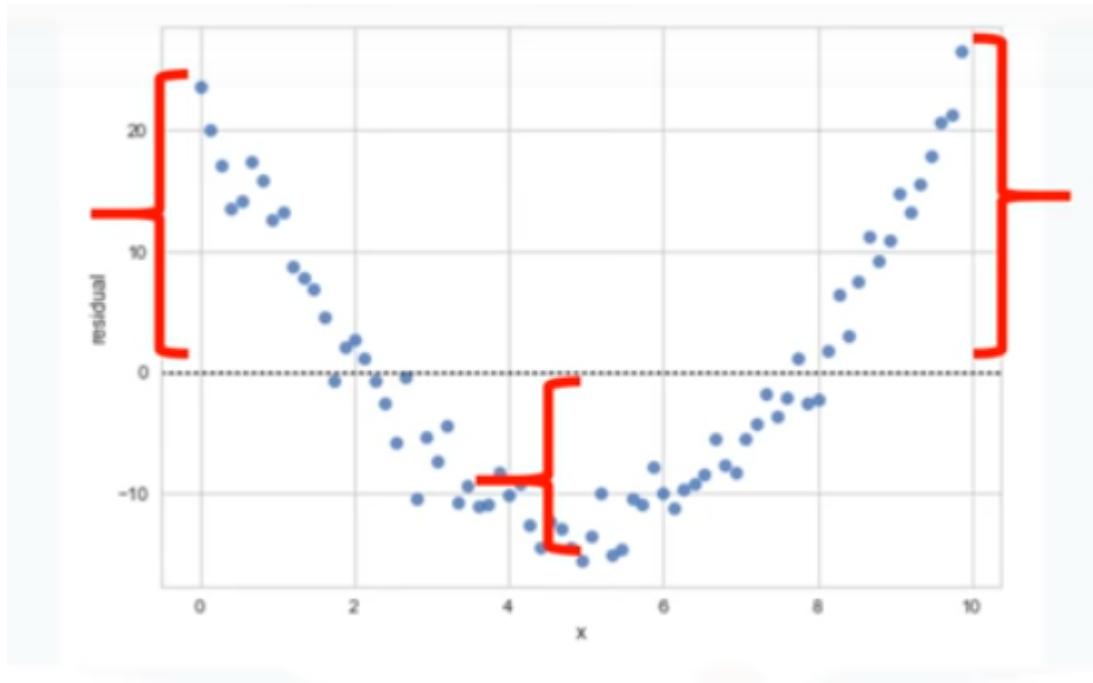
- Eje horizontal representa la variable independiente.
- Eje vertical variable dependiente.
- Cada punto representa un punto objetivo.
- la Linea representa el valor predecido.

```
import seaborn as sns
```

- De esta manera podemos hacerlo en Python.
- La grafica residual representa el error entre los valores actuales.
- Al examinar el valor predecido vs el actual encontramos una diferencia, la cual obtenemos restando el valor predecido del actual valor objetivo.
- Cuando graficamos ese valor en el eje vertical con la variable dependiente en la horizontal, de esta manera.

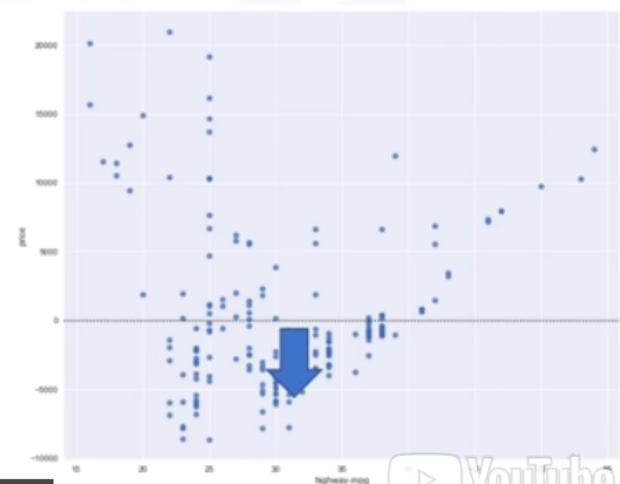


- Esperamos ver los resultados que tengan media de cero, distribuidos al rededor del eje x con varianza similar, no hay curvatura.
- Este tipo de grafico residual indica que la grafica lineal es apropiada.
- Si hay curvatura los valores del error cargan en x

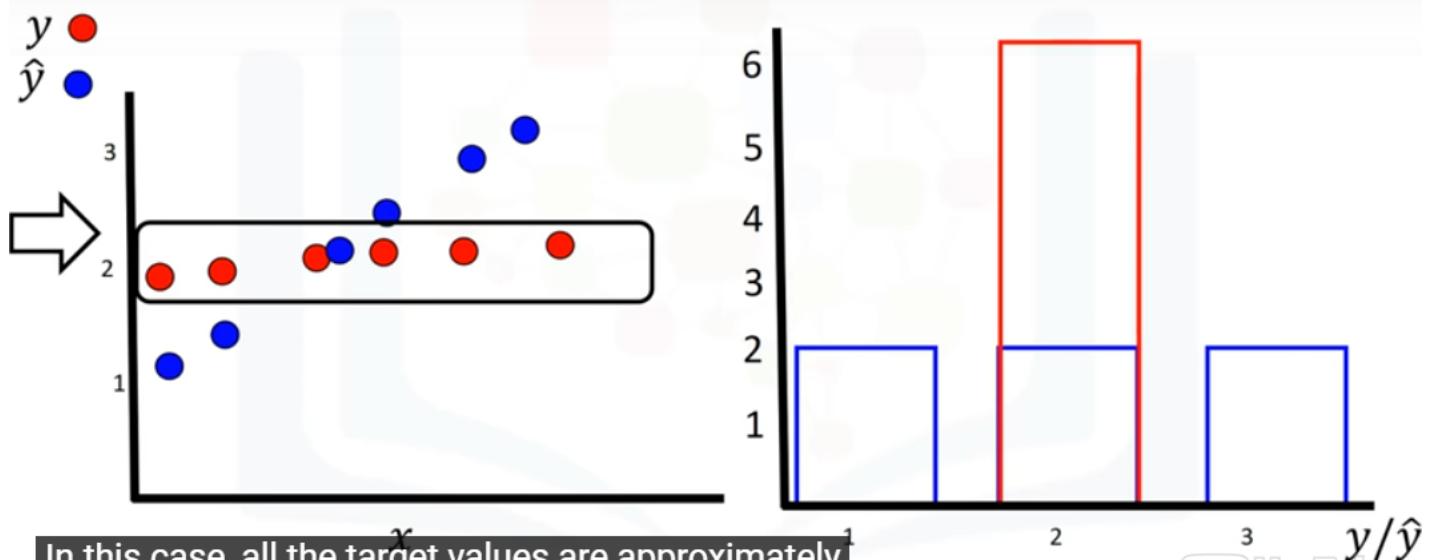


- En este caso el primer corchete indica residuos positivos, el segundo negativos, el tercero el error es largo.
- Estos errores no son aleatoriamente separados, lo cual sugiere que la regresión lineal es incorrecta, ya que esto indicaría que no funcionaría con una función lineal.
- De esta forma podemos graficar los residuos en Python:

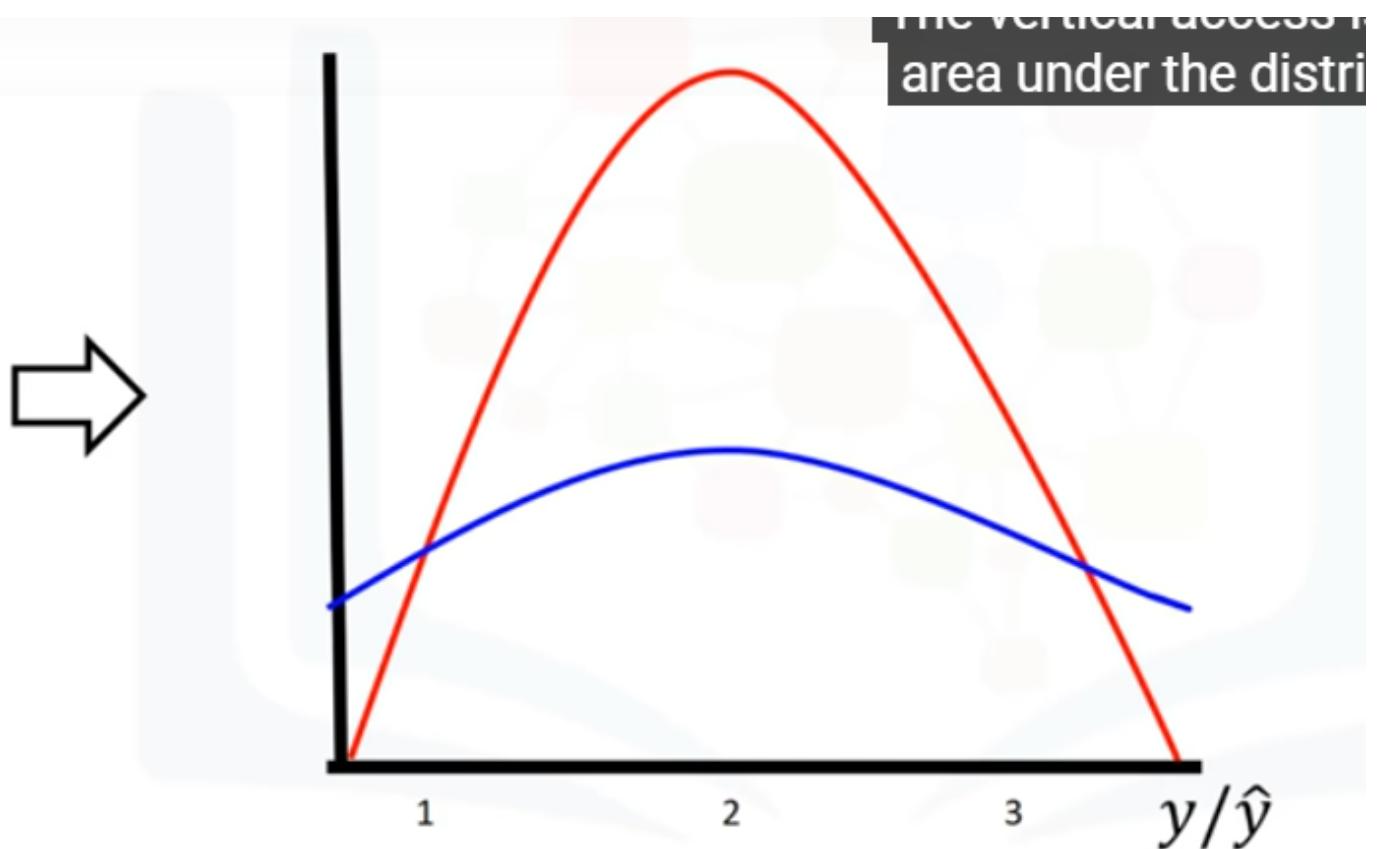
```
import seaborn as sns
sns.residplot(df['highway-mpg'], df['price'])
```



[View on YouTube](https://www.youtube.com/watch?v=JhEUoqe24efg)



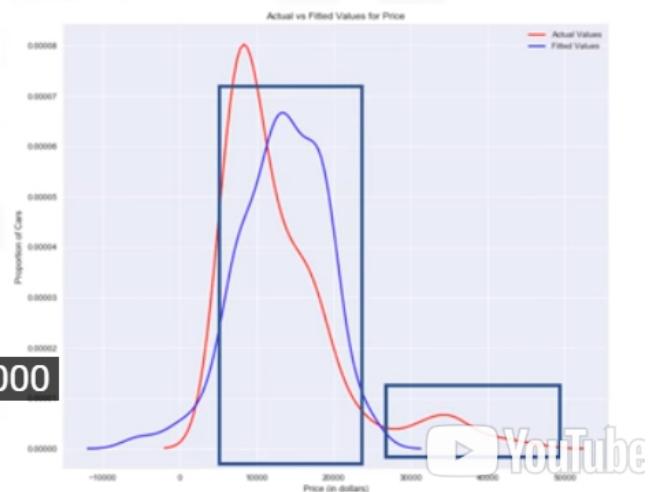
- Aquí graficamos nuestra grafica de distribución y lo que hacemos es contar en que valor se encuentran las predicciones, estos valores son continuos.
- Para valores discretos utilizamos un histograma, de esta manera:



Compare the distribution plots:

- The fitted values that result from the model
- The actual values

The prices in the region form \$10 000 to \$20 000
are much closer to the target value.



- En este caso vemos que la linea azul son los resultados del modelo, mientras que los rojos los valores actuales.
- Si observamos nos damos cuenta que los valores son muy similares, ahora creando la grafica de distribución:

```
import seaborn as sns
```

```
ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual Value")

sns.distplot(Yhat, hist=False, color="b", label="Fitted Values" , ax=ax1)
```

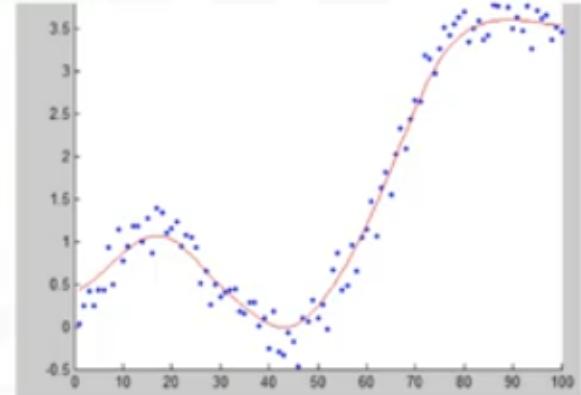
▼ Polynomial Regression and Pipelines

Polynomial Regressions

- A special case of the general linear regression model
- Useful for describing curvilinear relationships.

Curvilinear relationships:

By squaring or setting higher-order terms of the predictor variables.



- Quadratic – 2nd order

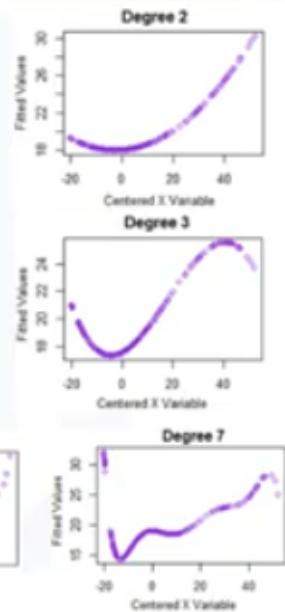
$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2$$

- Cubic – 3rd order

$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3$$

- Higher order

$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3 + \dots$$



1. Calculate Polynomial of 3rd order

```
f=np.polyfit(x,y,3)  
p=np.poly1d(f)
```

2. We can print out the model

```
print (p)
```

$$-1.557(x_1)^3 + 204.8(x_1)^2 + 8965x_1 + 1.37 \times 10^5$$

De esta manera podemos definir una función polinomial en Python

- The "preprocessing" library in scikit-learn,

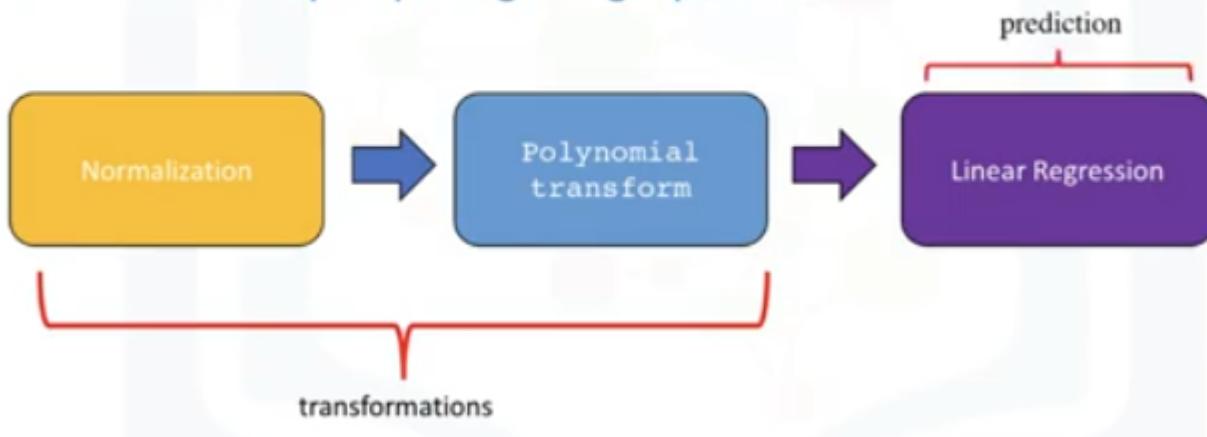
```
from sklearn.preprocessing import PolynomialFeatures  
pr=PolynomialFeatures(degree=2, include_bias=False)  
x_polly=pr.fit_transform(x[['horsepower', 'curb-weight']])
```

- For example we can Normalize the each feature simultaneously:

```
from sklearn.preprocessing import StandardScaler  
SCALE=StandardScaler()  
SCALE.fit(x_data[['horsepower', 'highway-mpg']])  
  
x_scale=SCALE.transform(x_data[['horsepower', 'highway-mpg']])
```

Pipelines

- There are many steps to getting a prediction



- Para hacerlo en Python, primero importamos las librerías

```

from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
  
```

Input=[('scale',StandardScaler()),('polynomial',PolynomialFeatures(degree=2),...
('model',LinearRegression())]

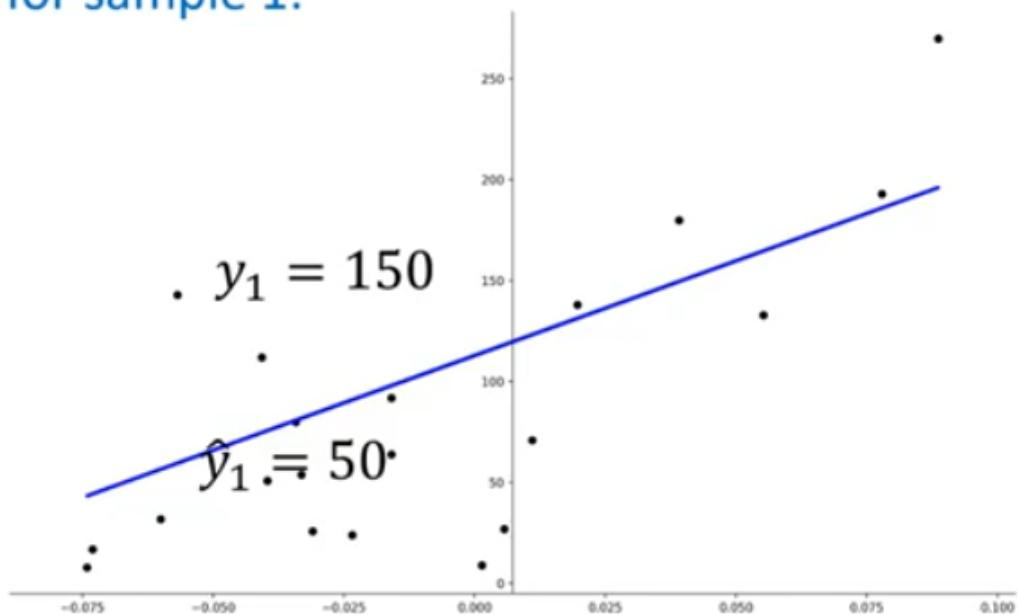
- Pipeline constructor
`pipe=Pipeline(Input)`

- Creamos una lista de tuplas, el primer elemento es el estimador del modelo, el segundo el constructor y nuestra input, de esta forma tenemos nuestro pipeline constructor.
- Podemos entrenar el pipeline aplicandole el metodo de entrenamiento, lo cual producira una predicción.
- El metodo normaliza los datos, hace una transformación polinomial y la salida una predicción.

▼ Measures for In-Sample Evaluation

- Esto nos ayuda a observar que tanto se acomoda el modelo al dataset.
- Hay dos medidas importantes para determinar que encaje el modelo: Error (MSE) y R-squared.
- Para encontrar el MSE lo hacemos con la diferencia del valor actual y el predecido (\hat{y}) y luego se eleva al cuadrado.

• For Example for sample 1:



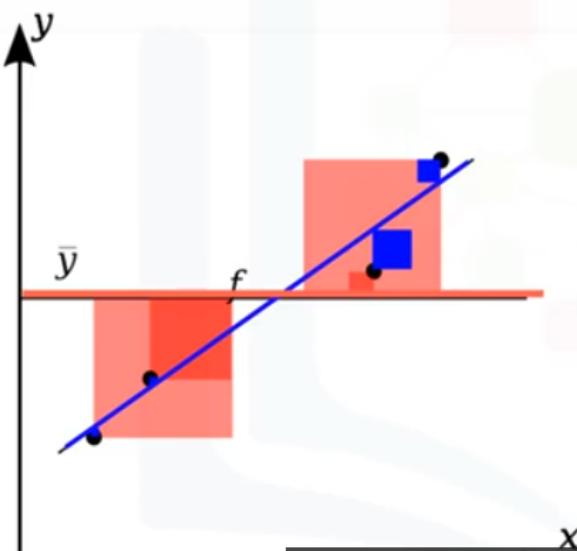
• In python we can measure the MSE as follows

```
from sklearn.metrics import mean_squared_error  
  
mean_squared_error(df['price'], Y_predict_simple_fit)  
  
3163502.944639888
```

R-squared/ R²

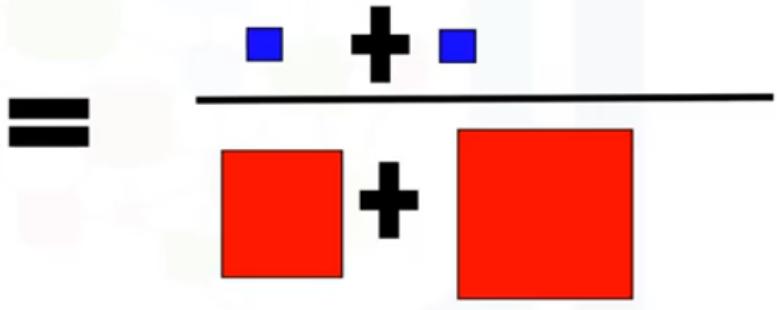
- The Coefficient of Determination or R squared (R²)
- Is a measure to determine how close the data is to the fitted regression line.
- R²: the percentage of variation of the target variable (Y) that is explained by the linear model.
- Think about as comparing a regression model to a simple model i.e the mean of the data points

$$R^2 = \left(1 - \frac{\text{MSE of regression line}}{\text{MSE of the average of the data}} \right)$$

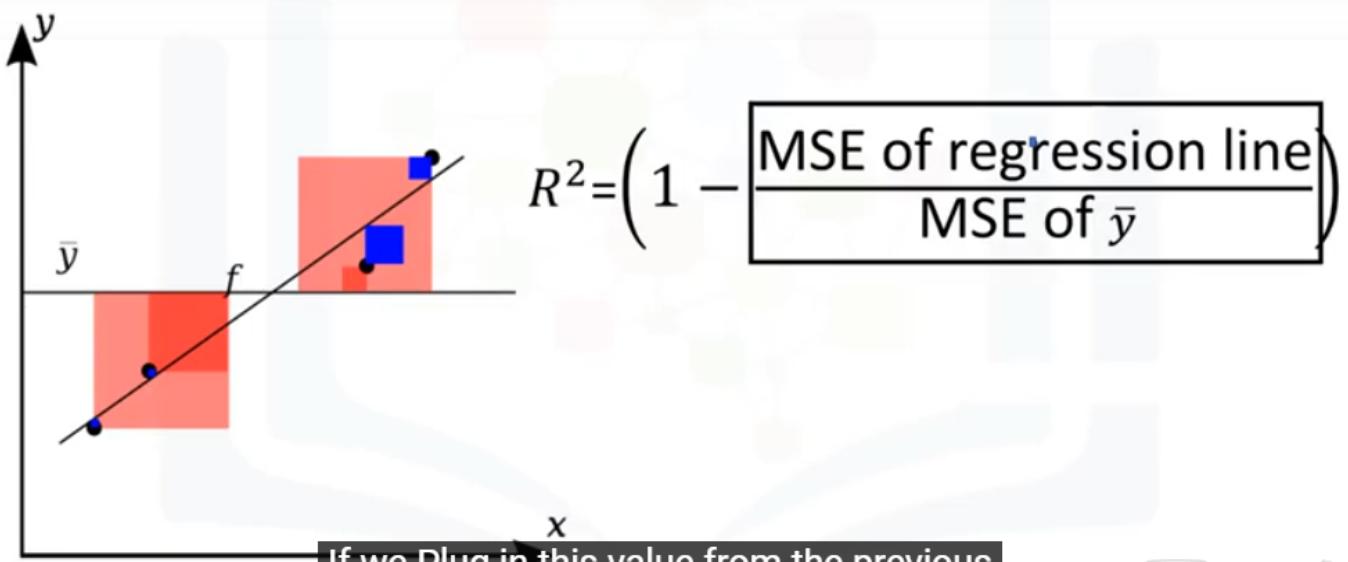


- The blue line represents the regression line
- The blue squares represents the MSE of the regression line
- The red line represents the average value of the data points
- The red squares represent the MSE of the red line
- We see the area of the blue squares is much smaller then the area of the red squares

$$\frac{\text{MSE of regression line}}{\text{MSE of } \bar{y}}$$



- Para este caso el MSE en el numerador es pequeño, mientras que el denominador es grande, esto nos da como resultado un numero muy pequeño, que llevandolo a un extremos nos daria un valor que tiende a 0.



- Generally the values of the MSE are between 0 and 1.
- WE can calculate the the R^2 as follows

```
X = df[['highway-mpg']]
```

```
Y = df['price']
```

```
lm.fit(X, Y)
```

```
lm.score(X, y)
```

```
0.496591188
```

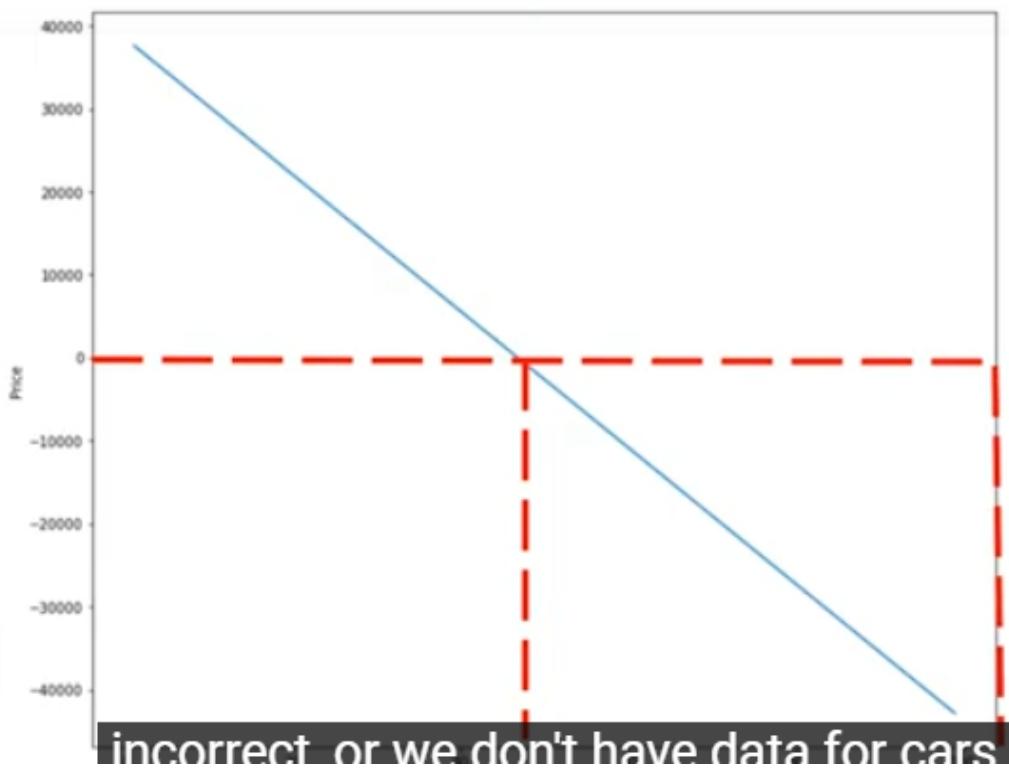
▼ Prediction and Decision Making

Es importante para encontrar el mejor modelo tener en cuenta lo siguiente:

- Los valores predecidos tengan sentido.
- Visualización.
- Medidas numericas para evaluación.
- Comparar modelos.

- First we train the model

Si analizamos de esta manera, podemos observar que en este caso la predicción tiene sentido.



Para este caso cuando tenemos valores negativos no tiene sentido, lo cual podría interpretarse que no tenemos vehículos en dicho rango con esas características.

- First we import numpy

```
import numpy as np
```

- We use the numpy function arange to generate a sequence from 1 to 100

```
new_input=np.arange(1,101,1).reshape(-1,1)
```



Si queremos crear una secuencia en numpy lo hacemos de la siguiente forma, teniendo en cuenta que el primer valor es el inicio, el segundo el límite y el tercero el incremento.

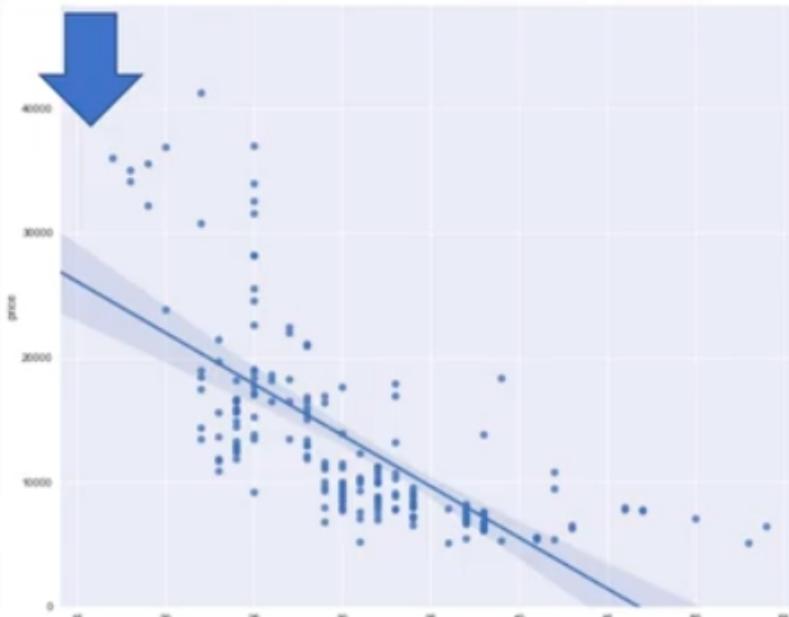
- We can predict new values

```
yhat=lm.predict(new_input)
```

```
array([[ 37601.57247984,   36779.83910151,   35958.10572319,   35136.37234487,
       34314.63896655,   33492.90558823,   32671.1722099 ,   31849.43883158,
       31027.70545326,   30205.97207494,   29384.23869662,   28562.50531829,
       27740.771933997,  26919.03856165,  26097.30518333,  25275.57180501,
       24453.83842668,  23632.10504836,  22810.37167004,  21988.63829172,
       21166.9049134 ,  20345.17153508,  19523.43815675,  18701.70477843,
       17879.97140011,  17058.23802179,  16236.50464347,  15414.77126514,
       14593.03788682,  13771.3045085 ,  12949.57113018,  12127.83775186,
       11306.10437353,  10484.37099521,  9662.63761689,  8840.90423857,
       8019.17086025,   7197.43748192,   6375.7041036 ,   5553.97072528,
       4732.23734696,   3910.50396864,   3088.77059031,   2267.03721199,
       1445.30383367,   623.57045535,  -198.16292297,  -1019.8963013 ,
      -1841.62967962,  -2663.36305794,  -3485.09643626,  -4306.82981458,
      -5128.5631929 ,  -5950.29657123,  -6772.02994955,  -7593.76332787,
      -8415.49670619,  -9237.23008451,  -10058.96346284,  -10880.69684116,
     -11702.43021948,  -12524.1635978 ,  -13345.89697612,  -14167.63035445,
     -14989.36373277,  -15811.09711109,  -16632.83048941,  -17454.56386773,
     -18276.29724606,  -19098.03062438,  -19919.7640027 ,  -20741.49738102,
     -21563.23075934,  -22384.96413767,  -23206.69751599,  -24028.43089431,
     -24850.16427263,  -25671.89765095,  -26493.63102927,  -27315.3644076 ,
     -28137.09778592,  -28958.83116424,  -29780.56454256,  -30602.29792088,
     -31424.03129921,  -32245.76467753,  -33067.49805585,  -33889.23143417,
     -34710.96481249,  -35532.69819082,  -36354.43156914,  -37176.16494746,
```

O si queremos utilizar la salida para predecir nuevos valores, podemos generar el array de numpy, donde observamos que muchos de los valores son negativos, si utilizamos una grafica de regresión podemos observar mejor estos datos.

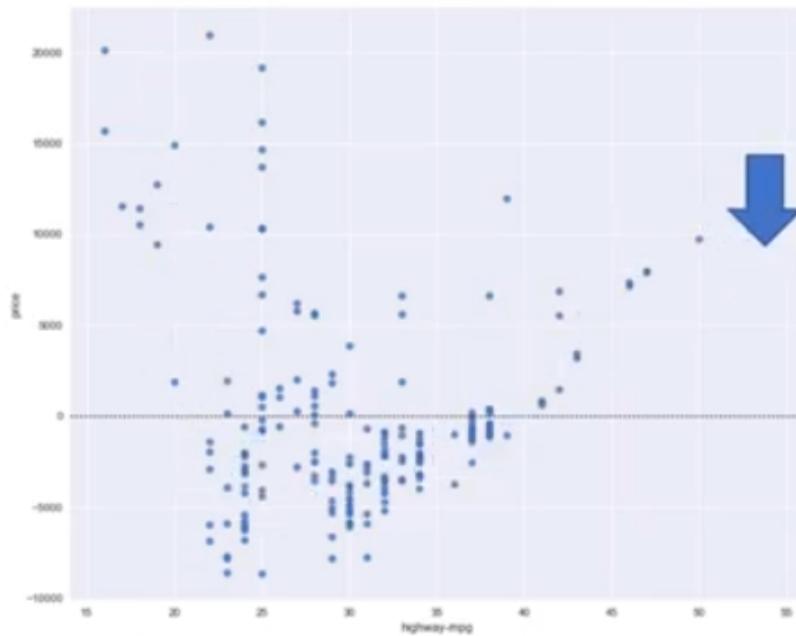
- Simply visualizing your data with a regression



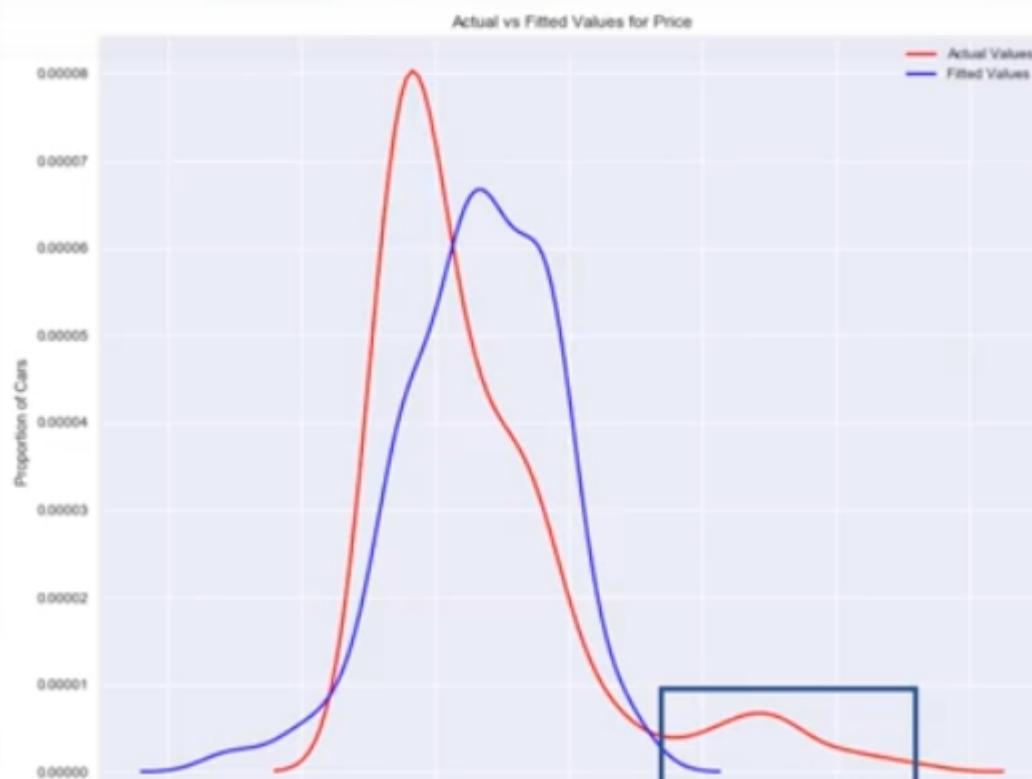
Para este caso el efecto en la variable independiente es muy evidente, ya que los datos tienden a bajar cuando la variable dependiente incrementa, también observamos un comportamiento no

lineal, y si detallamos podemos ver una curvatura en ellos,

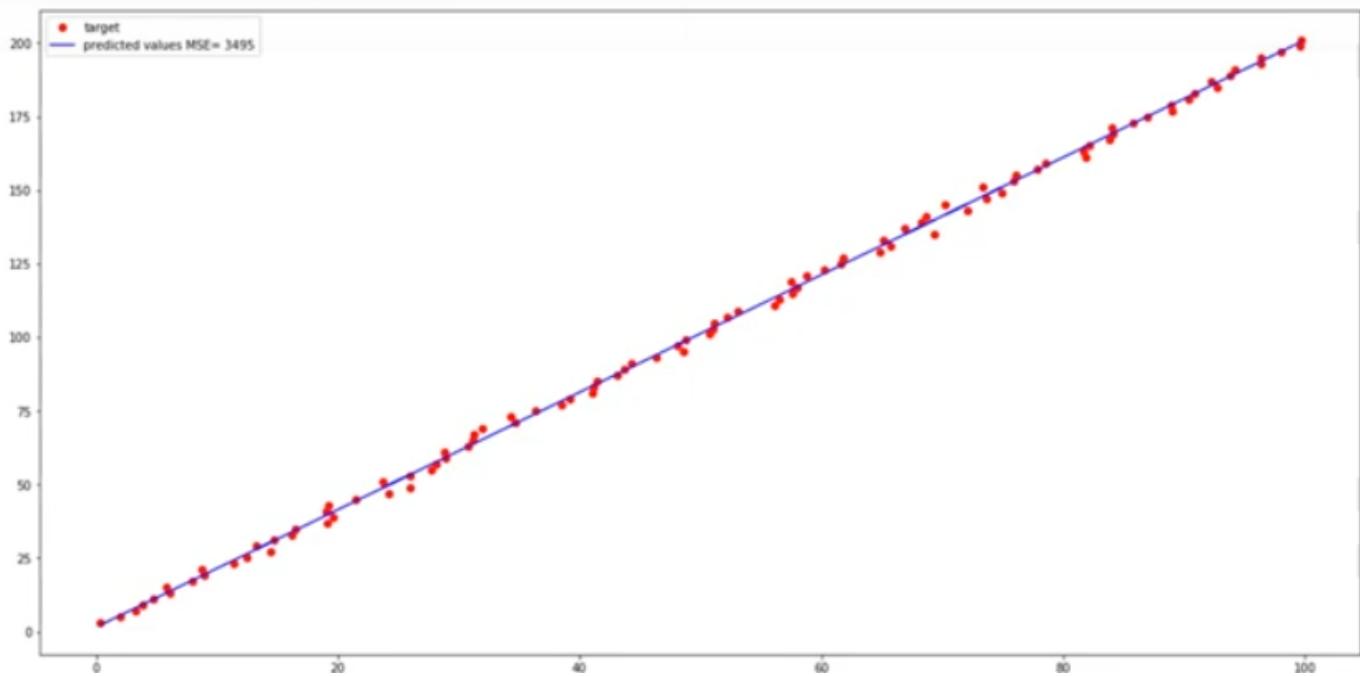
Residual Plot



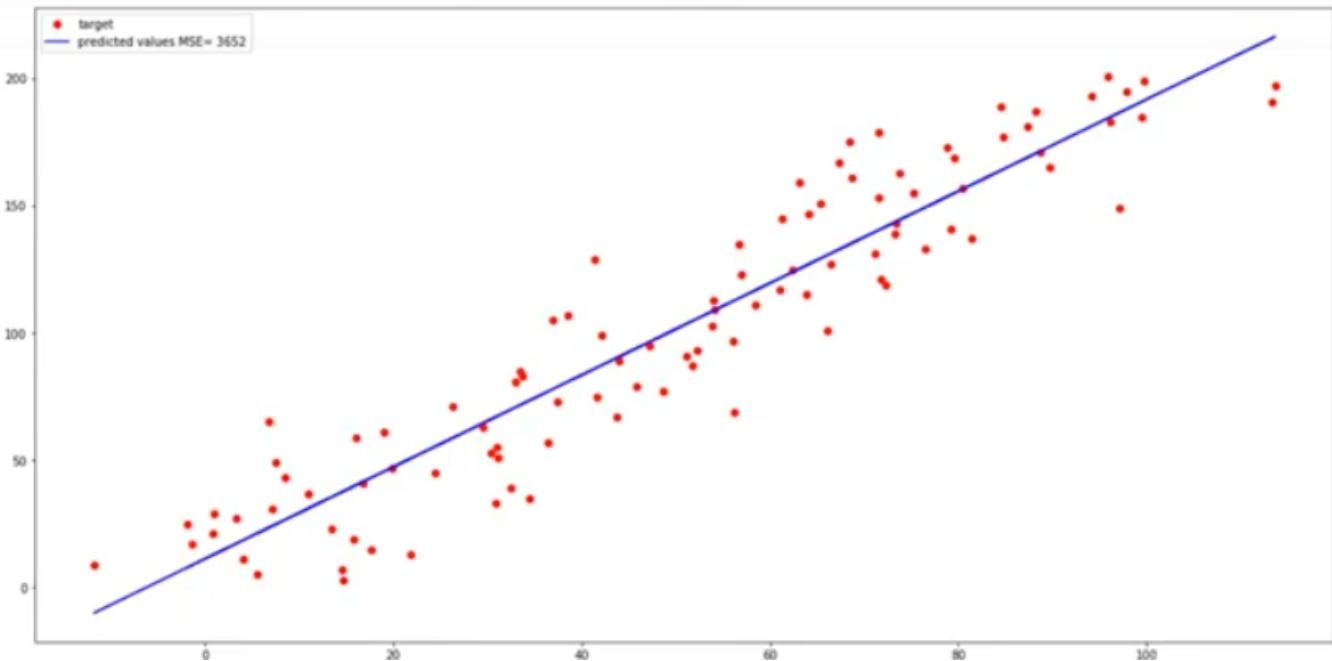
donde podemos probar una grafica de distribución, de la siguiente manera



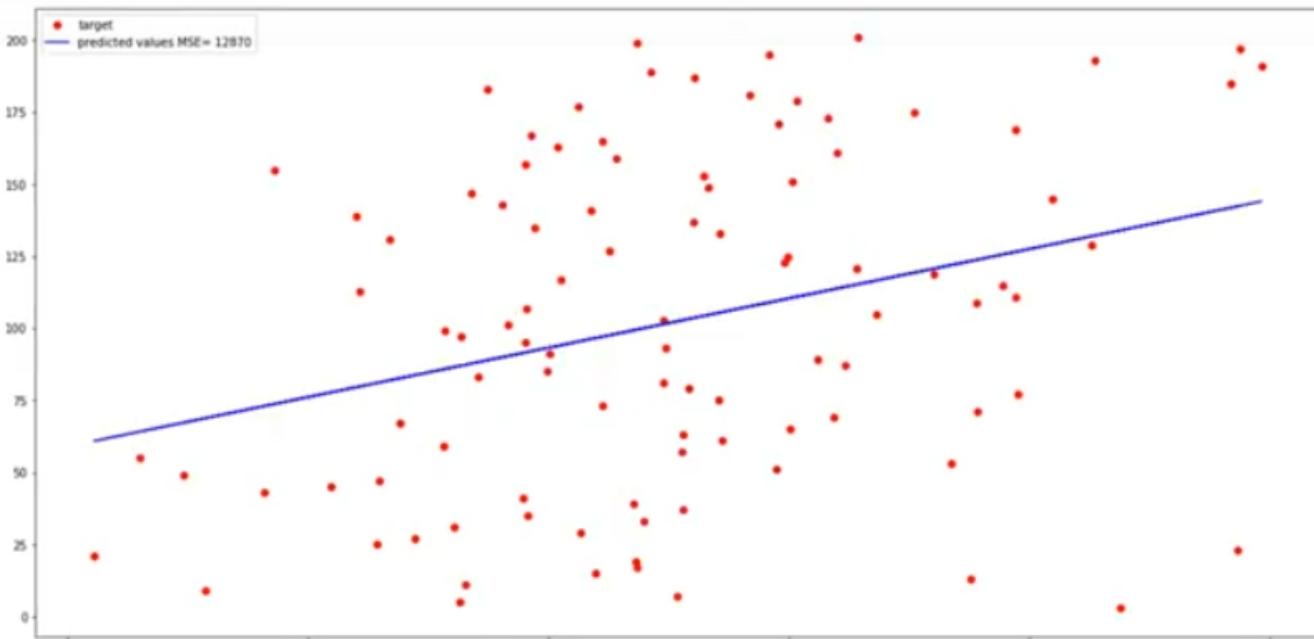
el rango de 30.000 a 50.000 no es muy preciso, lo cual sugiere que necesitamos mas datos en este rango, ademas que el MSE es la medida numerica mas intuitiva para determinar si un modelo es bueno o no.



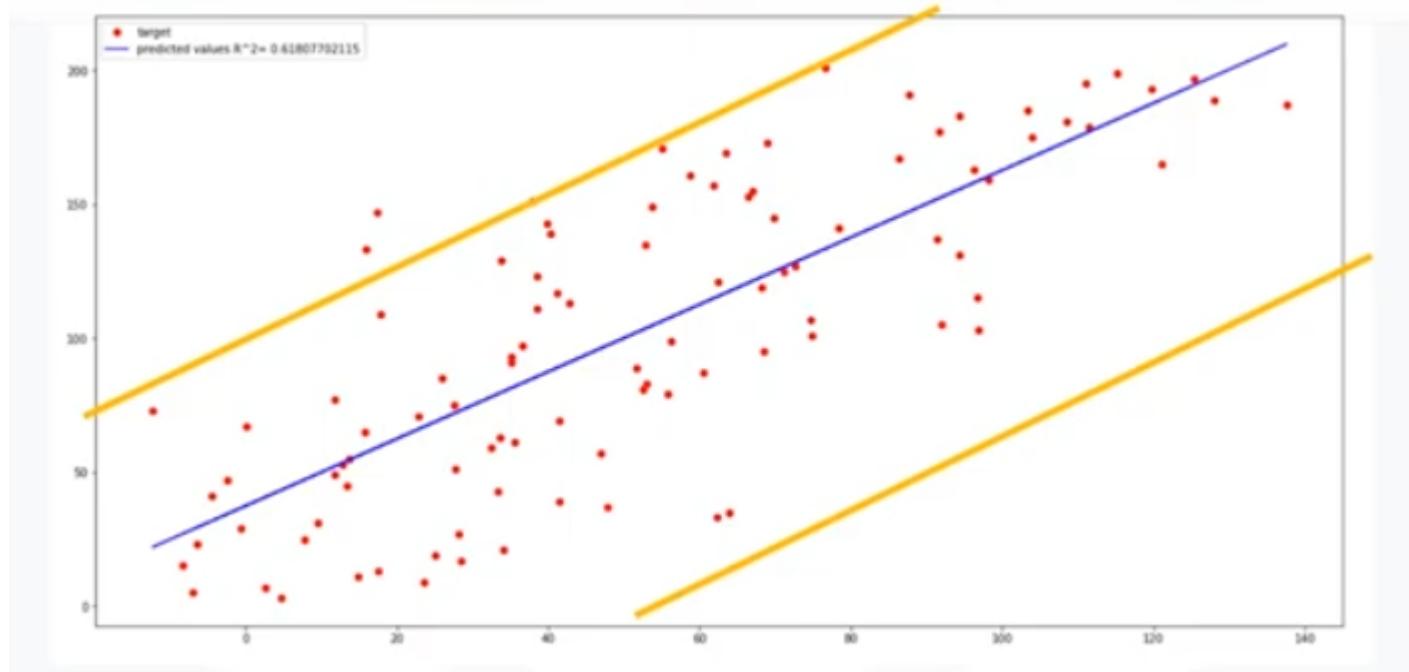
Evidenciamos un MSE de 3,495, mientras que en este otro ejemplo tenemos un MSE de 3.652



Nuestra grafica final tiene un MSE de 12,870



Entre mas aumenta el MSE, el objetivo se aleja de los valores predecidos.



Si utilizamos el MSE cuadratico, evidenciamos que los datos disminuyen con la variable independiente, y un valor aceptable de R^2 depende de que campo se este estudiando, ya que algunos autores indican que deberia estar entre 0.10 o mayor.

Comparing MLR and SLR

1. Is a lower MSE always implying a better fit?
 - Not necessarily.
2. MSE for an MLR model will be smaller than the MSE for an SLR model, since the errors of the data will decrease when more variables are included in the model
3. Polynomial regression will also have a smaller MSE than regular regression
4. A similar inverse relationship holds for R^2

Lab 4

Question #1 a):

Create a linear regression object called "lm1".

```
# Write your code below and press Shift+Enter to execute
lm1 = LinearRegression()
lm1
```



```
LinearRegression()
```

Question #1 b):

Train the model using "engine-size" as the independent variable and "price" as the dependent variable?

```
# Write your code below and press Shift+Enter to execute
lm1.fit(df[['engine-size']], df[['price']])
lm1
```



```
LinearRegression()
```

Question #1 c):

Find the slope and intercept of the model.

Slope

```
# Write your code below and press Shift+Enter to execute
lm1.coef_
```

```
array([[166.86001569]])
```

► Click here for the solution

Intercept

```
# Write your code below and press Shift+Enter to execute
lm1.intercept_
```

```
array([-7963.33890628])
```

Question #1 d):

What is the equation of the predicted line? You can use x and yhat or "engine-size" or "price".

```
: # Write your code below and press Shift+Enter to execute
# using X and Y
Yhat=-7963.34 + 166.86*X

Price=-7963.34 + 166.86*df['engine-size']
```

Question #2 a):

Create and train a Multiple Linear Regression model "lm2" where the response variable is "price", and the predictor variable is "normalized-losses" and "highway-mpg".

```
# Write your code below and press Shift+Enter to execute
lm2 = LinearRegression()
lm2.fit(df[['normalized-losses' , 'highway-mpg']],df['price'])
```

```
LinearRegression()
```

Question #2 b):

Find the coefficient of the model.

```
# Write your code below and press Shift+Enter to execute  
lm2.coef_
```

```
array([ 1.49789586, -820.45434016])
```

Question #3:

Given the regression plots above, is "peak-rpm" or "highway-mpg" more strongly correlated with "price"? Use the method ".corr()" to verify your answer.

```
# Write your code below and press Shift+Enter to execute  
df[["peak-rpm", "highway-mpg", "price"]].corr()
```

	peak-rpm	highway-mpg	price
peak-rpm	1.000000	-0.058598	-0.101616
highway-mpg	-0.058598	1.000000	-0.704692
price	-0.101616	-0.704692	1.000000

Question #4:

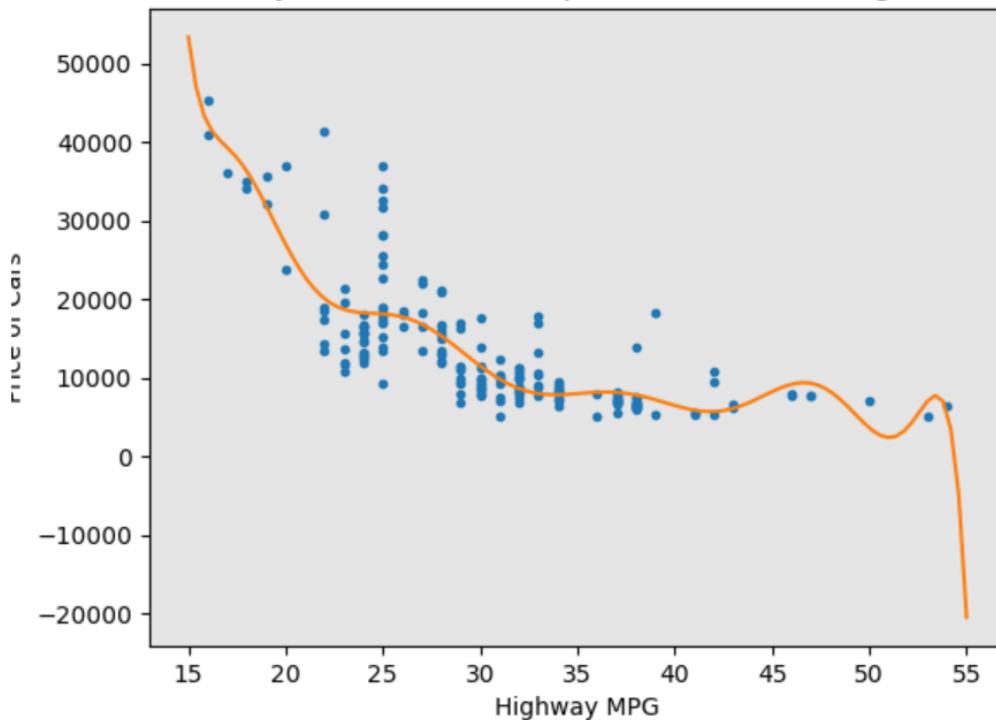
Create 11 order polynomial model with the variables x and y from above.

```
# Write your code below and press Shift+Enter to execute
```

```
f1 = np.polyfit(x, y, 11)
p1 = np.poly1d(f1)
print(p1)
PlotPolly(p1,x,y, 'Highway MPG')
```

```
11          10          9          8          7
-1.243e-08 x + 4.722e-06 x - 0.0008028 x + 0.08056 x - 5.297 x
6           5           4           3           2
+ 239.5 x - 7588 x + 1.684e+05 x - 2.565e+06 x + 2.551e+07 x - 1.491e+08 x + 3.879e+08
```

Polynomial Fit with Matplotlib for Price ~ Length



Question #5:

Create a pipeline that standardizes the data, then produce a prediction using a linear regression model using the features Z and target y.

```
# Write your code below and press Shift+Enter to execute
Input=[('scale',StandardScaler()),('model',LinearRegression())]

pipe=Pipeline(Input)

pipe.fit(Z,y)

ypipe=pipe.predict(Z)
ypipe[0:10]
```

```
array([13699.11161184, 13699.11161184, 19051.65470233, 10620.36193015,
       15521.31420211, 13869.66673213, 15456.16196732, 15974.00907672,
       17612.35917161, 10722.32509097])
```

▼ Graded Review Questions 4

Question 1

1/1 point (graded)

Let `X` be a dataframe with 100 rows and 5 columns. Let `y` be the target with 100 samples. Assuming all the relevant libraries and data have been imported, the following line of code has been executed:

```
LR = LinearRegression()
LR.fit(X, y)
yhat = LR.predict(X)
```

How many samples does `yhat` contain?

5

500

100

0



Show answer

[Submit](#)

You have used 2 of 2 attempts

✓ Correct (1/1 point)

Question 2

1/1 point (graded)

What value of R^2 (coefficient of determination) indicates your model performs best?

 -100 -1 0 1

[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 3

1/1 point (graded)

Which statement is true about polynomial linear regression?

 Polynomial linear regression is not linear in any way. Although the predictor variables of polynomial linear regression are not linear, the relationship between the parameters or coefficients is linear. Polynomial linear regression uses wavelets.

[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 4

1/1 point (graded)

The larger the mean squared error, the better your model performs:

False

True



[Show answer](#)

[Submit](#)

You have used 1 of 1 attempt

Correct (1/1 point)

Question 5

1/1 point (graded)

Assume all the libraries are imported. y is the target and X is the features or dependent variables. Consider the following lines of code:

```
Input=[('scale',StandardScaler()),('model',LinearRegression())]  
pipe=Pipeline(Input)  
pipe.fit(X,y)  
ypipe=pipe.predict(X)
```

What is the result of $ypipe$?

Polynomial transform, standardize the data, then perform a prediction using a linear regression model.

Standardize the data, then perform prediction using a linear regression model.

Polynomial transform, then standardize the data.



[Show answer](#)

[Submit](#)

You have used 2 of 2 attempts

Correct (1/1 point)

▼ Module 5 – Model Evaluation

▼ Model Evaluation and Refinement

Los objetivos para este modulo son:

- Evaluación del modelo.
- Sobreentrenamiento, subentrenamiento y selección del modelo.
- Regresión Rigde.
- Busqueda de malla o Grid Search.

Y responderemos a la siguiente pregunta:

- **Como podemos estar seguros de que nuestro modelo funciona en el mundo real y funciona optimamente?**

▼ Model Evaluation

- Una muestra de la evaluación dice como de bien nuestro modelo se desempeña con los datos utilizados para su entrenamiento.
- El problema? No puede decirnos que tan bien nuestro modelo fue entrenado y puede ser usado para predecir nuevos datos.
- Solución: Conjunto de entrenamiento y conjunto de validación.

Data:

- Split dataset into:
 - Training set (70%), 
 - Testing set (30%) 
- Build and train the model with a training set
- Use testing set to assess the performance of a predictive model
- When we have completed testing our model we should use all the data to train the model to get the best performance

Con la siguiente función en Python podemos dividir los datos de la siguiente manera:

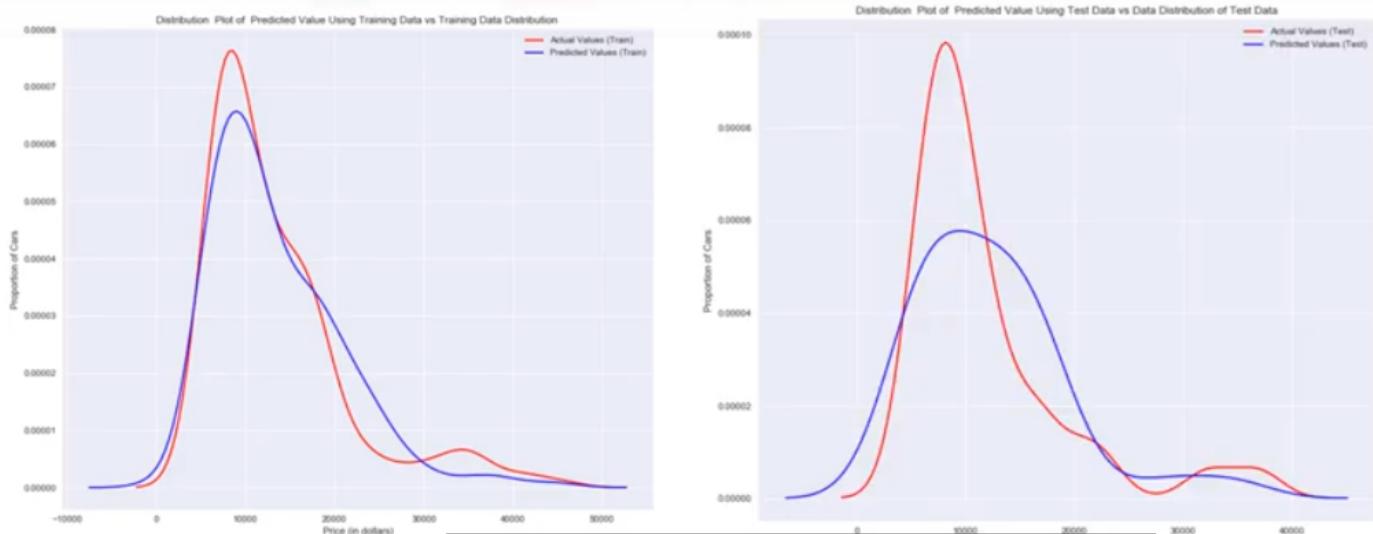
```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=0)
```

- **x_data:** features or independent variables
- **y_data:** dataset target: df['price']
- **x_train, y_train:** parts of available data as training set
- **x_test, y_test:** parts of available data as testing set
- **test_size:** percentage of the data for testing (here 30%)
- **random_state:** number generator used for random sampling

- El error de generalización mide como nuestros datos están haciendo sus predicciones con datos nunca vistos.
- El error que obtenemos usando nuestros datos de prueba es una aproximación a este error.

Observandolo graficamente obtenemos lo siguiente:



Si observamos la grafica de la derecha observamos que en la vida real los datos azules son diferentes a las predicciones, y se debe al error de generalización. Utilizando muchos datos de entrenamiento podemos determinar como nuestro modelo esta desempeñandose.

[Ver más ta...](#)[Compartir](#)

Lots of Training Data



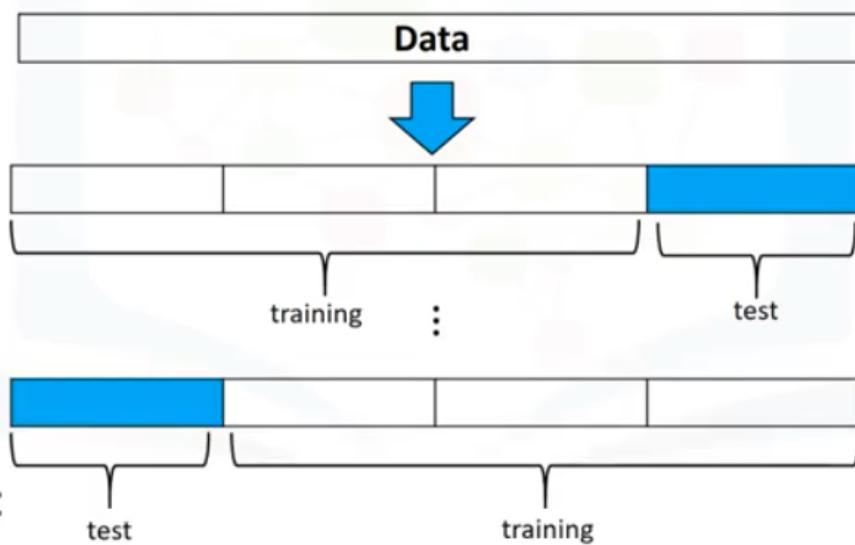
Model



will perform in the real world, but the precision
of the performance will be low.



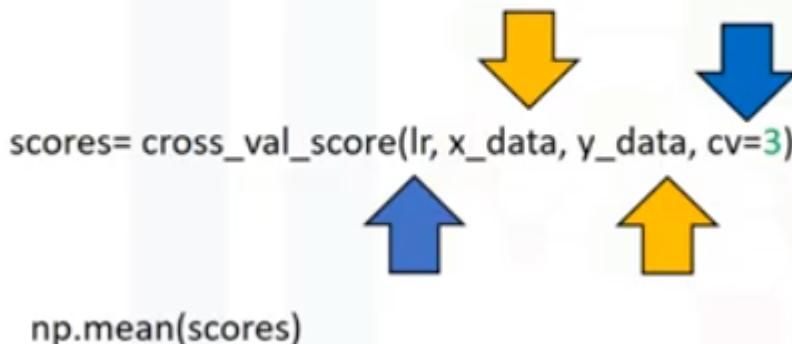
- Si utilizamos muchos pocos datos de entrenamiento y mas de validación la exactitud del modelo sera mayor, pero su desempeño bajo, para solucionar este dilema recurrimos a la cross validation.
- Most common out-of-sample evaluation metrics
- More effective use of data (each observation is used for both training and testing)



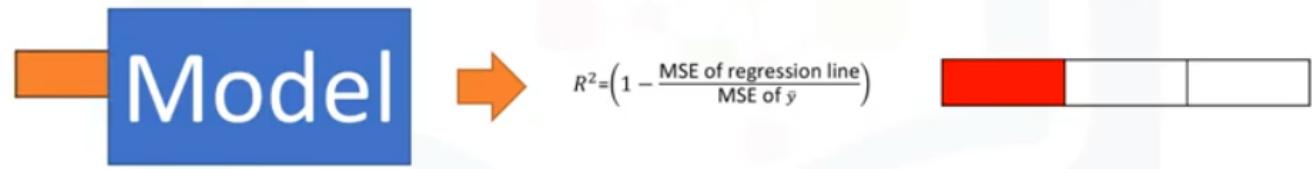
- Al finalizar utilizamos el promedio de los resultados como un estimado de el error de muestra, aunque la metrica de evaluación depende del modelo. Por ejemplo con R-squared. En el

siguiente ejemplo observaremos como dividimos nuestro conjunto de entrenamiento en 3 partes, y en un array almacenaremos los resultados 1 para cada partición que se haya elegido.

```
from sklearn.model_selection import cross_val_score
```



scores



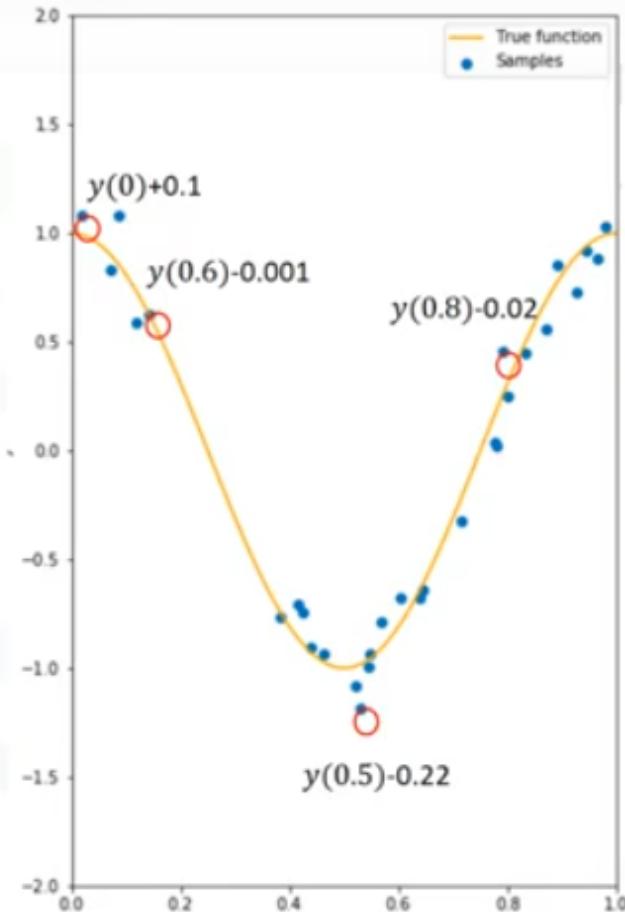
- It returns the prediction that was obtained for each element when it was in the test set
- Has a similar interface to `cross_val_score()`

```
from sklearn.model_selection import cross_val_predict
```

```
yhat= cross_val_predict (lr2e, x_data, y_data, cv=3) ←
```

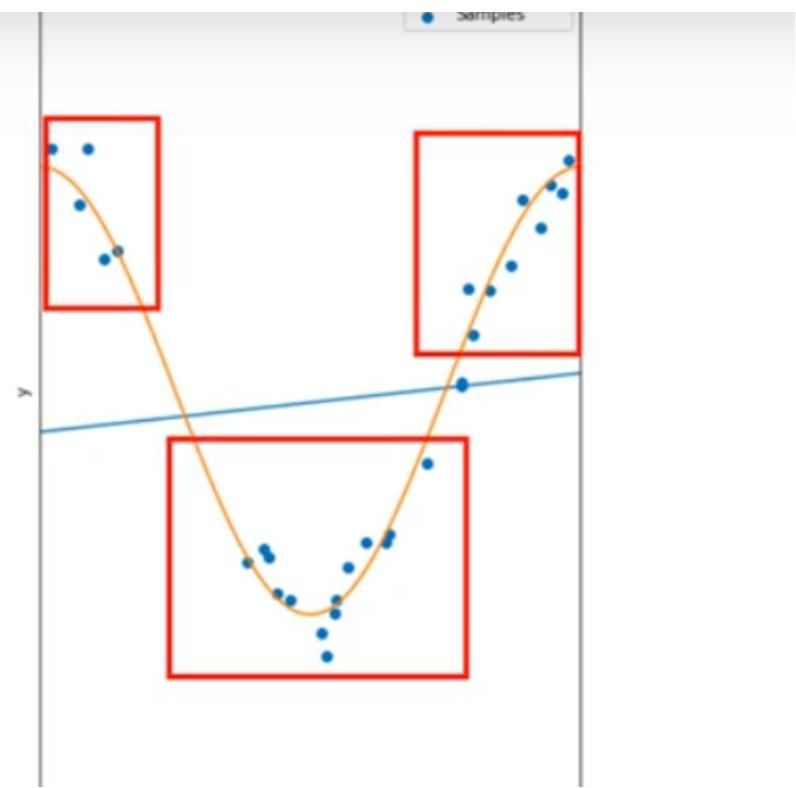
▼ Overfitting, Underfitting and Model Selection

$y(x) + \text{noise}$

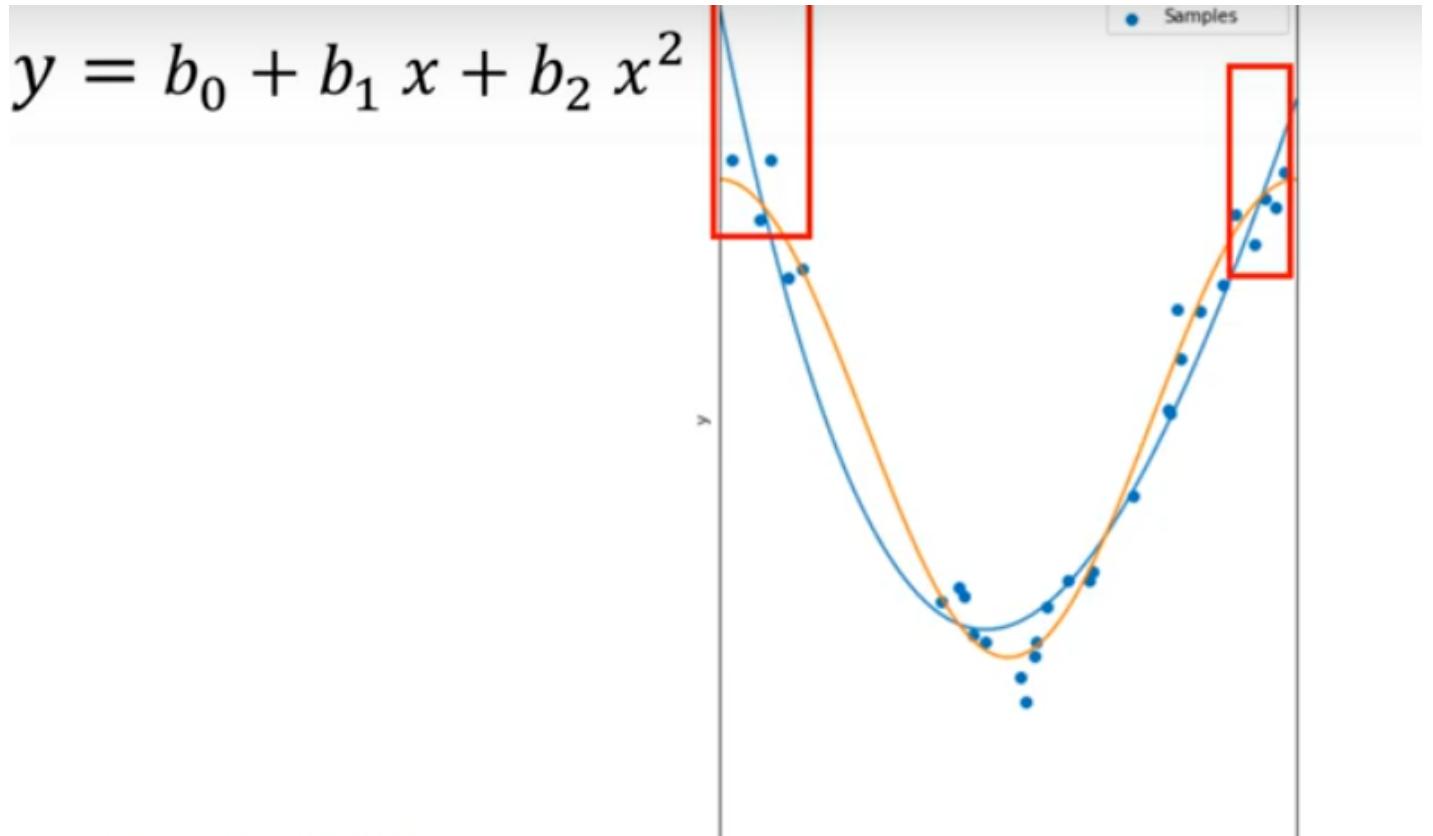


En este caso el objetivo es encontrar el orden del modelo polinomial que mejor se ajuste a la función $y(x)$, donde en el siguiente ejemplo observamos que no coinciden los valores.

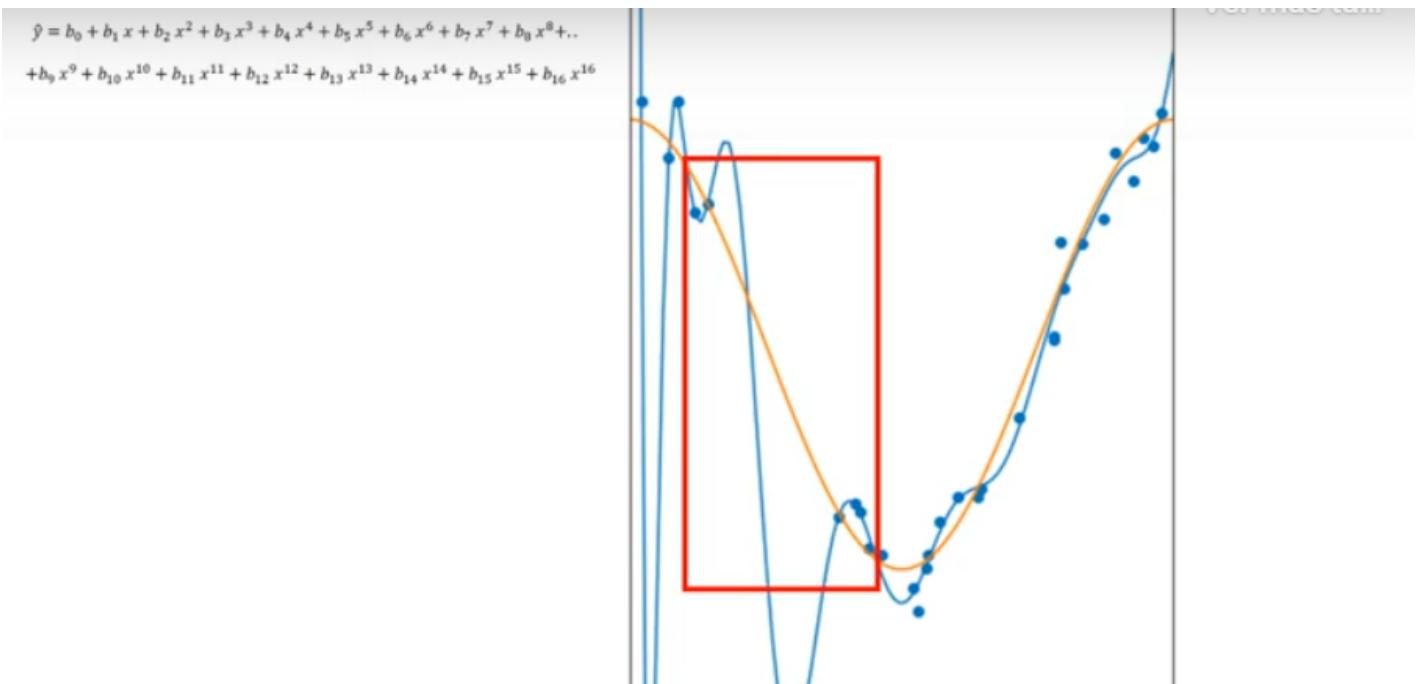
$$y = b_0 + b_1 x$$



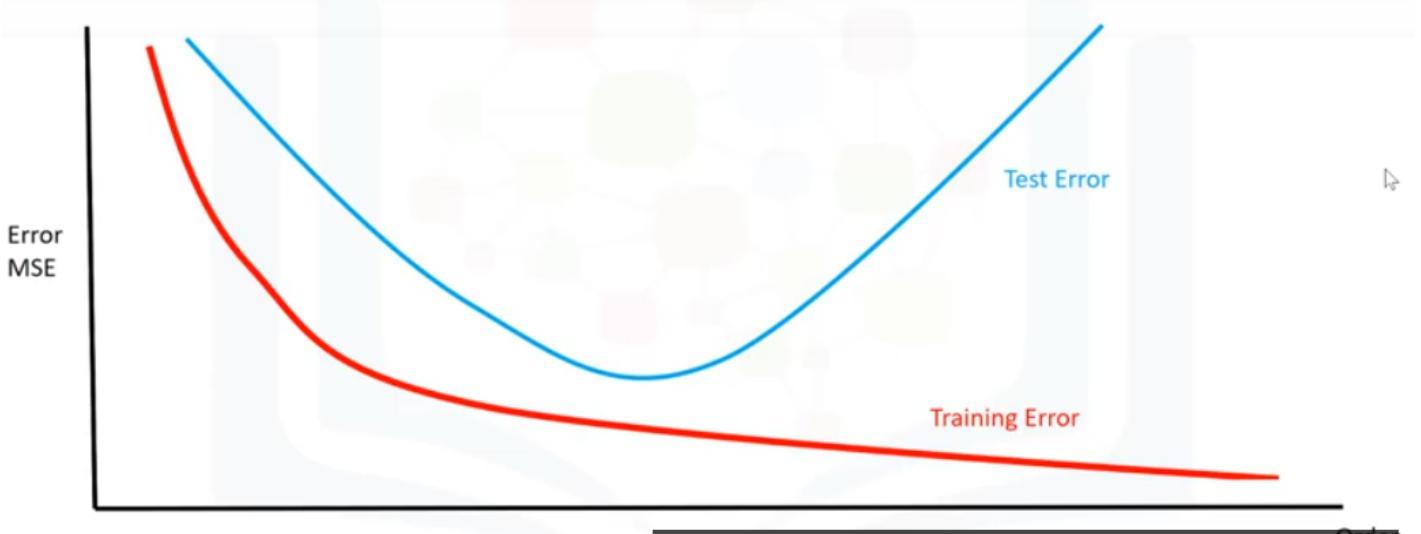
- Esto es llamado subentrenamiento cuando el modelo es muy simple para adaptarse a los datos.
- Si incrementamos el orden de los datos el modelo se ajustara mejor a los datos, pero aun también presenta subentrenamiento



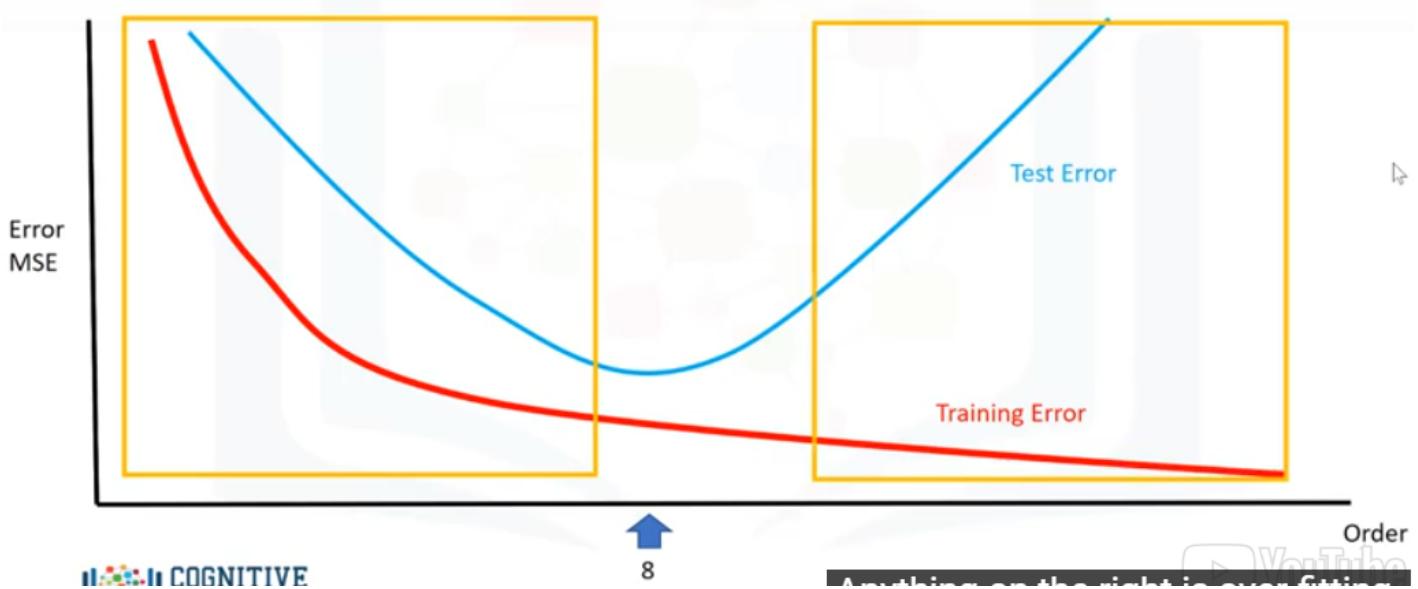
Si aumentamos mucho el polinomio de la función caemos en sobreentrenamiento, lo cual indica que esta aprendiendo de memoria los datos de entrenamiento, más no siguiendo la función, exagerando su flexibilidad.



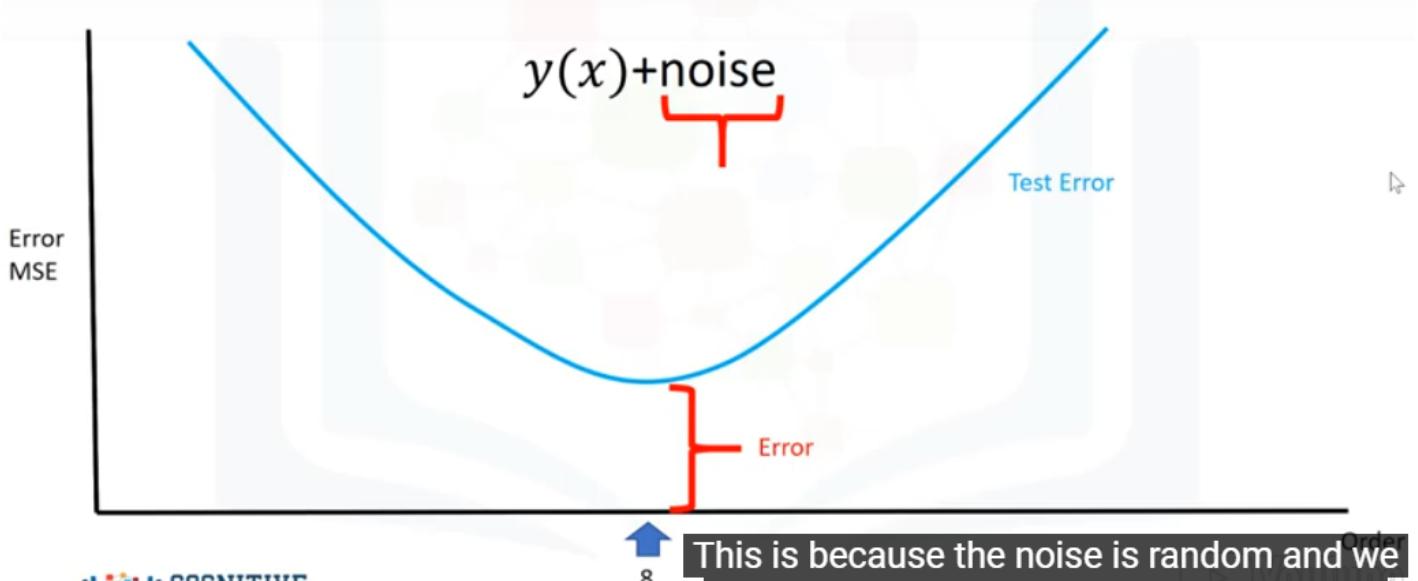
Aquí observamos una grafica con el MSE, donde observamos que entre mas aumenta el polinomio de la función menor es el error en entrenamiento, pero mayor es el error en la validación.



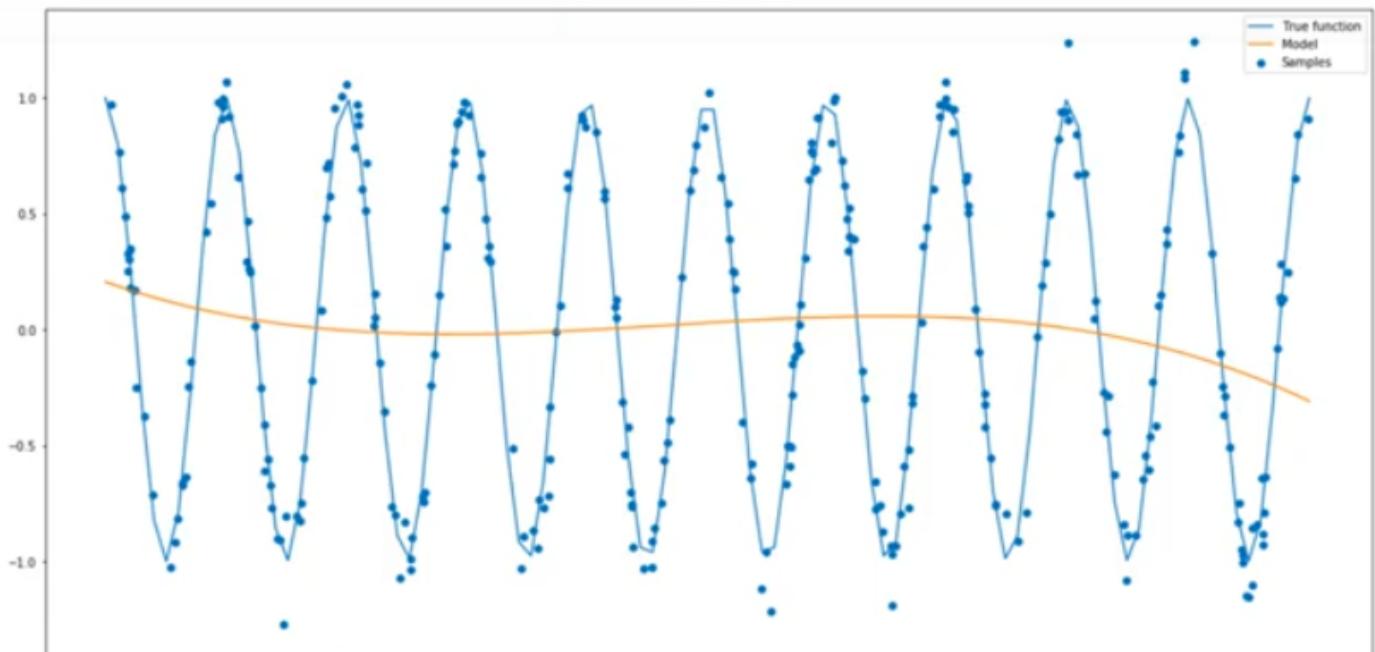
Determinando el mejor nivel de polinomio es 8, y todo lo que esta a la izquierda es considerado subentrenamiento, y lo que esta a la derecha sobreentrenamiento.



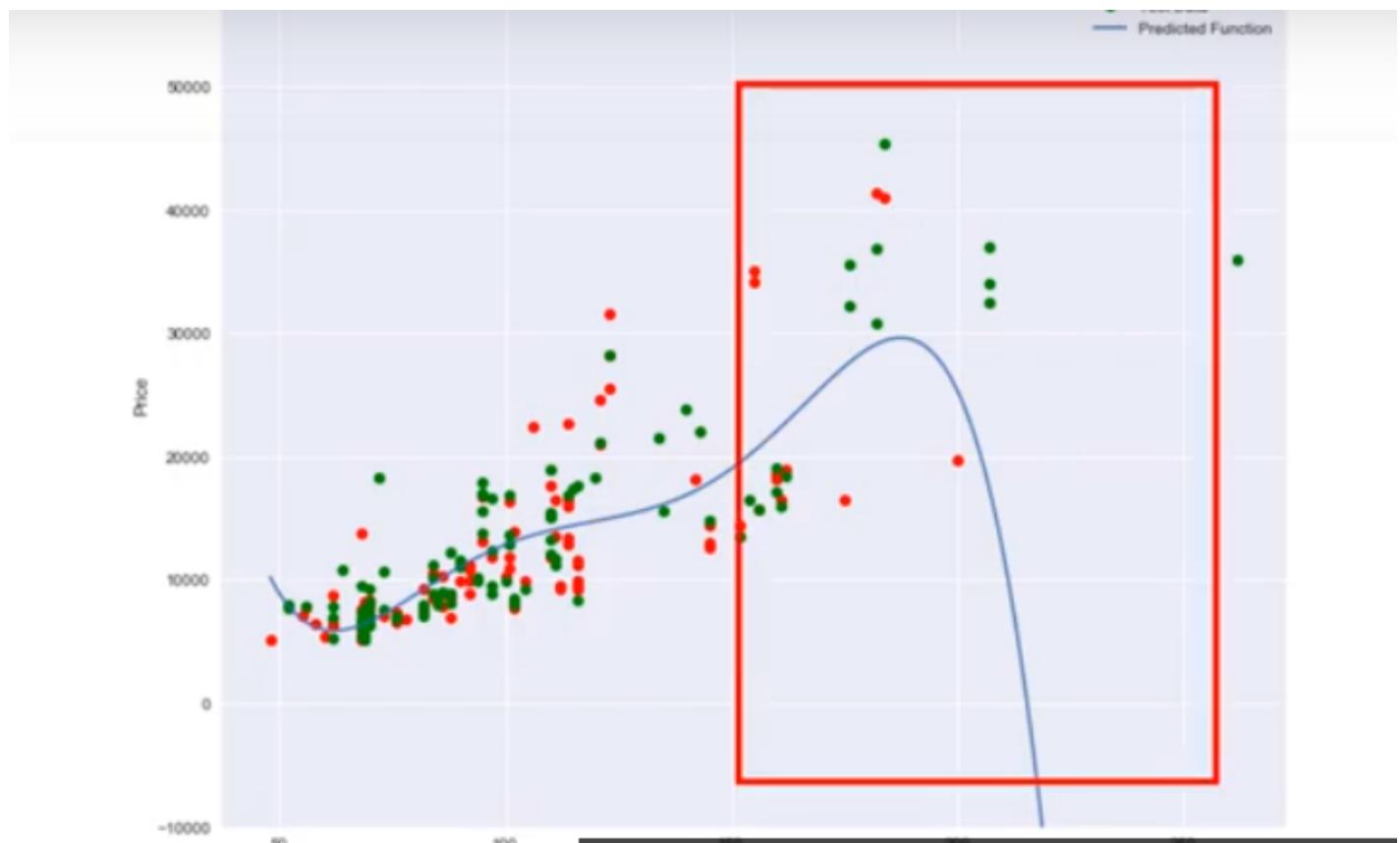
Aunque tengamos el mejor polinomio tendremos un ligero error, pero es debido al ruido, el cual es aleatorio y no podemos predecir ni reducir.



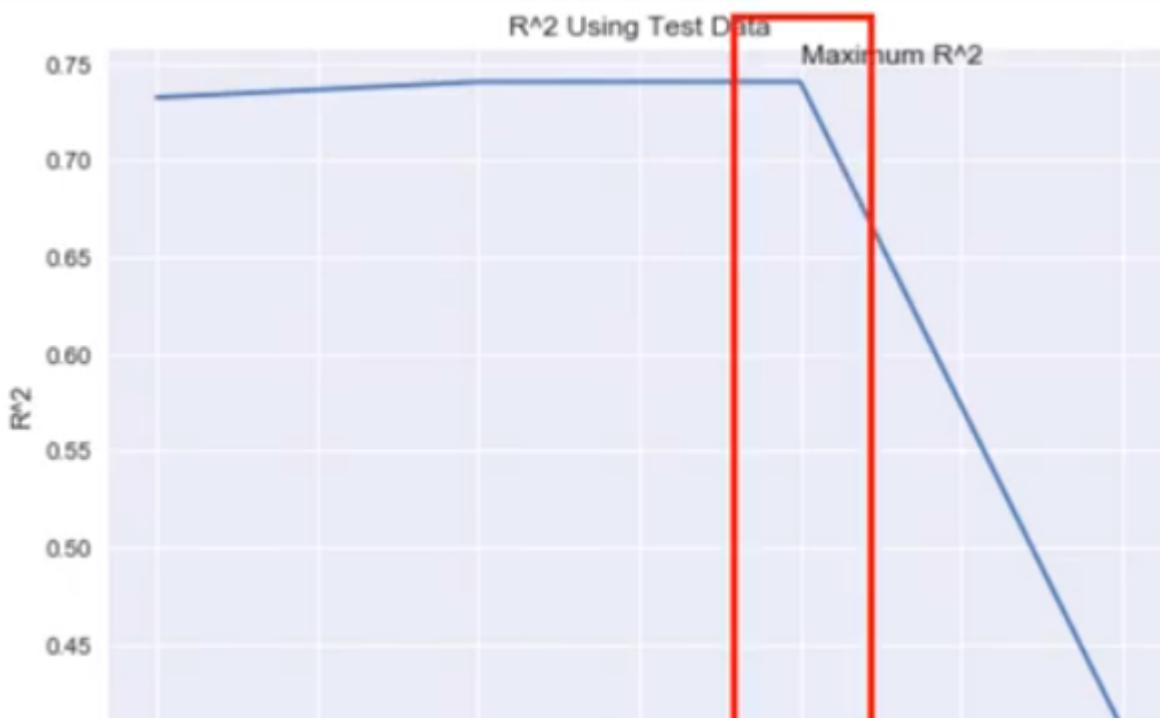
En este otro ejemplo los datos estan en una funcion senoidal, y para los datos reales el modelo tendra dificultades en ajustarse o tal vez no tenemos los datos correctos para estimar la funcion.



Para este caso con un modelo grado 4 el precio predecido disminuye, lo cual es erroneo, para confirmarlo debemos utilizar el R-squared para ver si es correcto.



Entre mas cercano a 1 este este error mas exacto es el modelo.



El R-squared disminuye drásticamente cuando el polinomio es de grado 4 validando la suposición inicial, y si queremos calcular diferentes valores de error podemos hacer lo siguiente:

```
order=[1,2,3,4]
```

```
for n in order:
```

```
pr=PolynomialFeatures(degree=n)
```

```
x_train_pr=pr.fit_transform(x_train[['horsepower']])
```

```
x_test_pr=pr.fit_transform(x_test[['horsepower']])
```

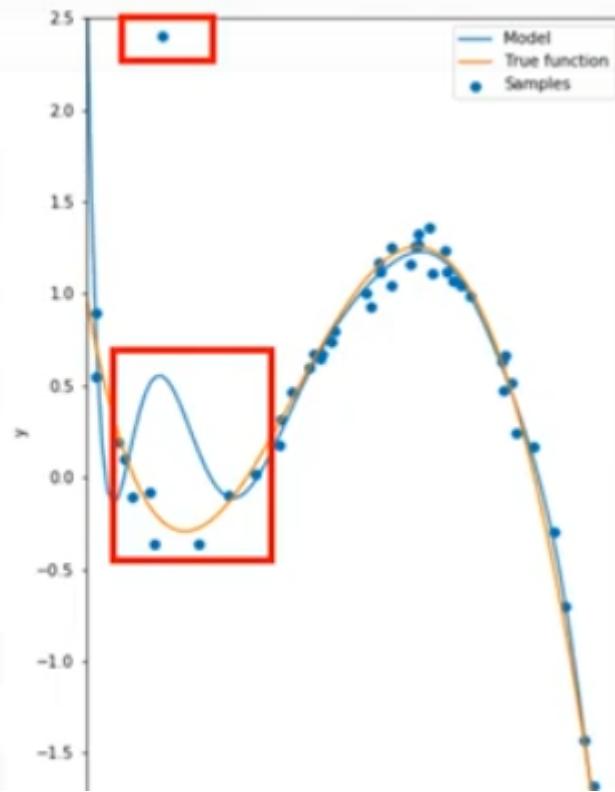
```
lr.fit(x_train_pr,y_train)
```

```
Rsqu_test.append(lr.score(x_test_pr,y_test))
```

▼ Ridge Regression

En muchas ocasiones nuestros datos tienen outliers, y si utilizamos una función polinómica con un grado alto podemos tener el siguiente problema.

$$1 + 2x - 3x^2 - 4x^3 + x^4$$



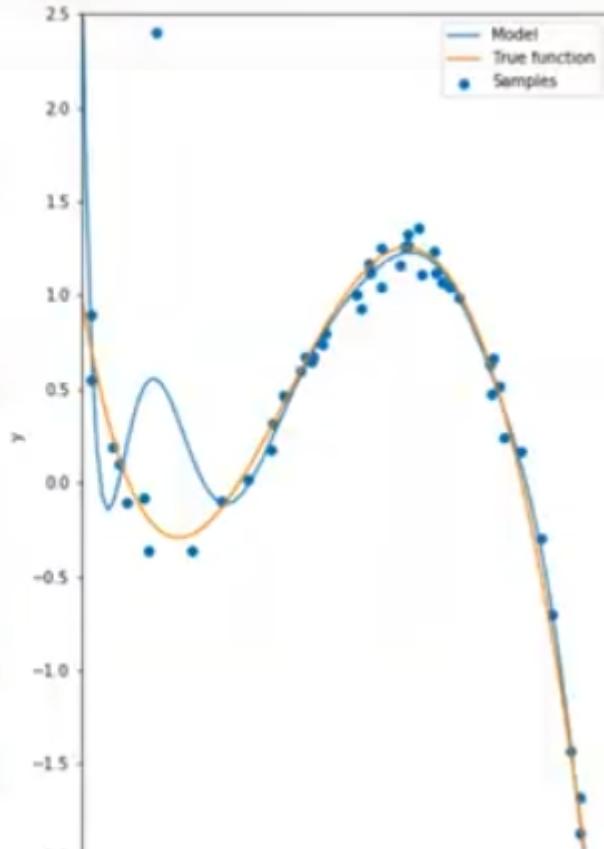
Es por eso que con la regresión Ridge incluimos un valor de alfa que nos ayuda a controlar un poco los parámetros del modelo, como observamos a continuación:

$$\hat{y} = 1 + 2x - 3x^2 - 2x^3 - 12x^4 - 40x^5 + 80x^6 + 71x^7 - 141x^8 - 38x^9 + 75x^{10}$$

Alpha	x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}
0	2	-3	-2	-12	-40	80	71	-141	-38	75
0.001	2	-3	-7	5	4	-6	4	-4	4	6
0.01	1	-2	-5	-0.04	0.15	-1	1	-0.5	0.3	1
1	0.5	-1	-1	-0.614	0.70	-0.38	-0.56	-0.21	-0.5	-0.1
10	0	-0.5	-0.3	-0.37	-0.30	-0.30	-0.22	-0.22	-0.22	-0.17

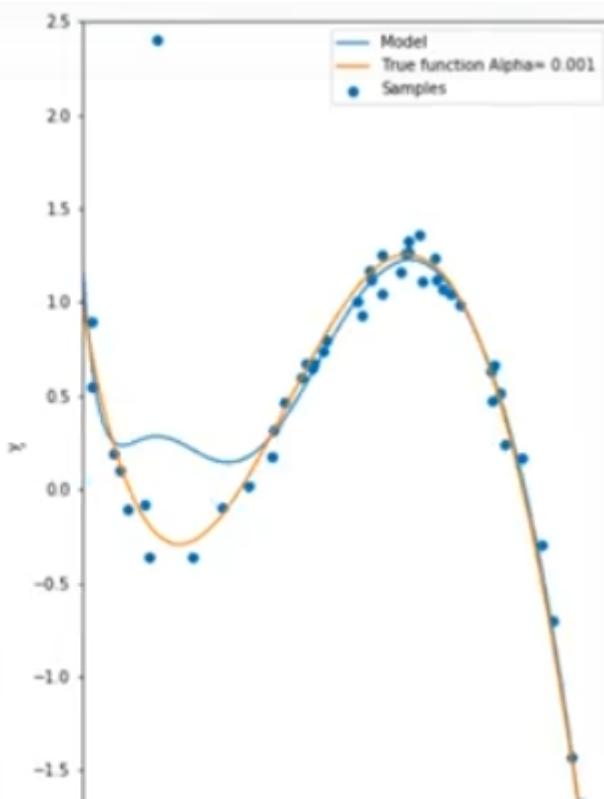
Si alfa es muy grande los coeficientes pueden acercarse a 0 y subentrenar el modelo, si hacemos lo opuesto podemos sobreentrenarlo.

alpha
0
0.001
0.01
1
10



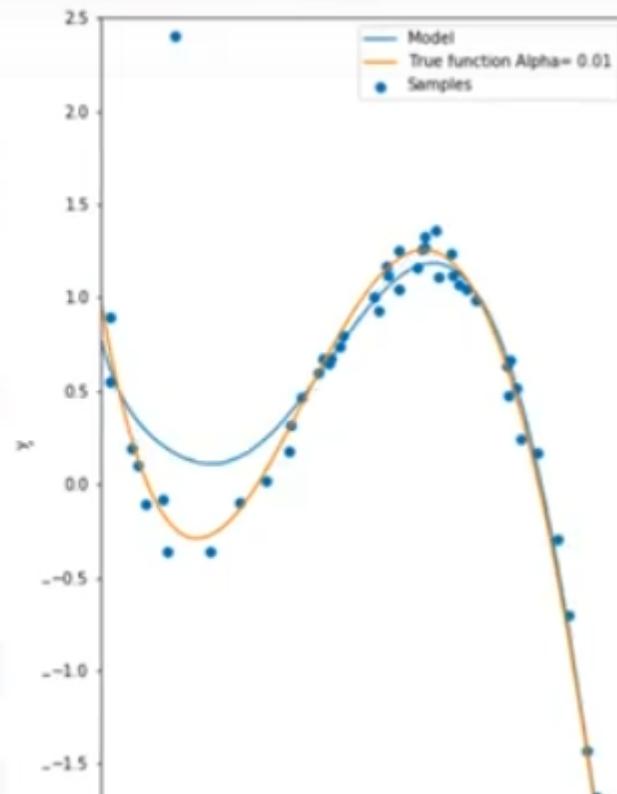
- Para alfa 0.001 el sobreentrenamiento empieza

alpha
0
0.001
0.01
1
10



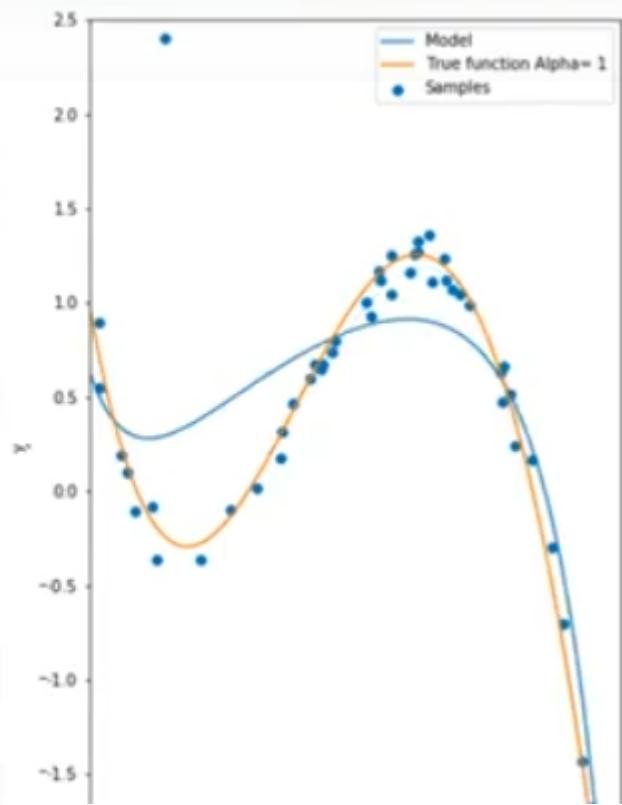
- Para alfa 0.01 la función predice bien.

alpha
0
0.001
0.01
1
10

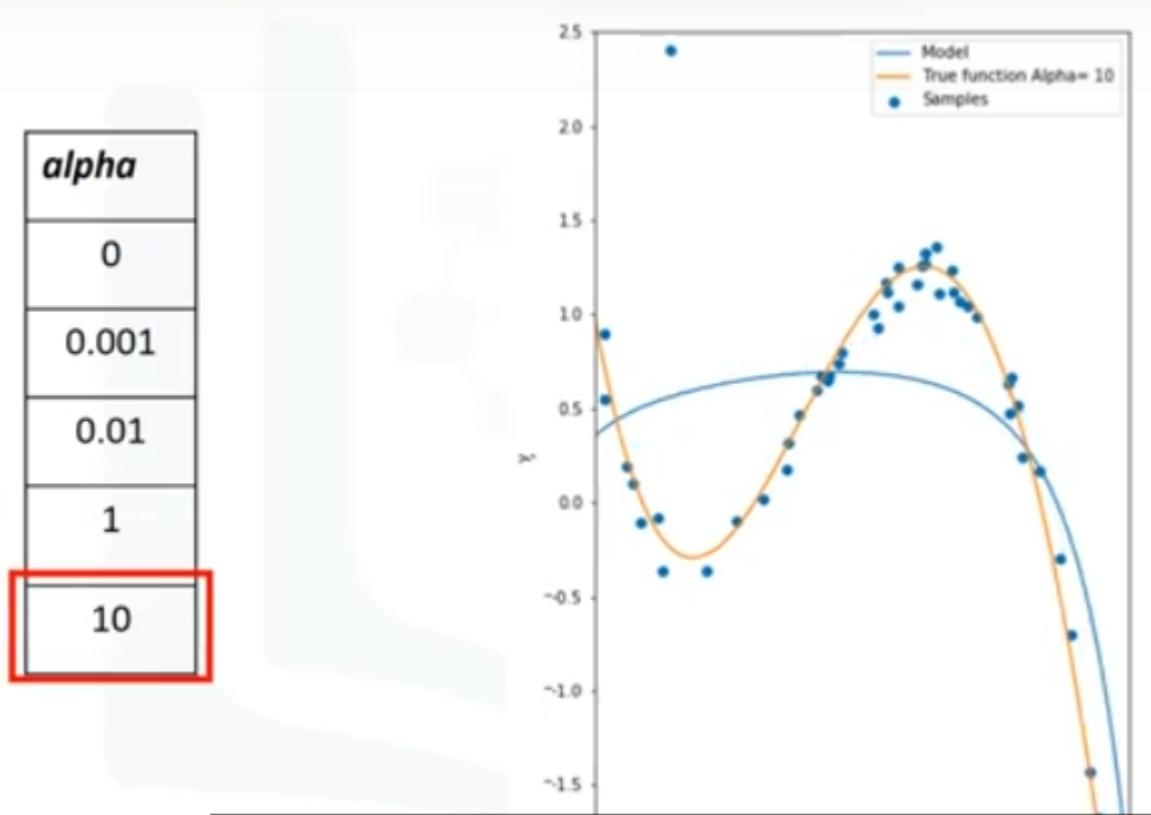


- Para alfa 1 evidenciamos los primeros indicios de subentrenamiento, ya que la función no tiene mucha flexibilidad.

alpha
0
0.001
0.01
1
10



- para alfa 10 evidenciamos un extremo subentrenamiento.



Para elegir el alfa adecuado utilizamos la validación cruzada, de esta forma:

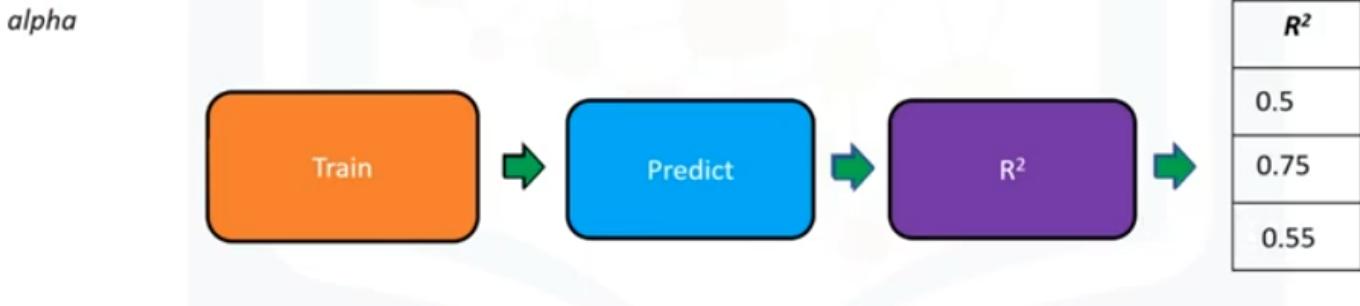
```
from sklearn.linear_model import Ridge
```

```
RidgeModel=Ridge(alpha=0.1)
```

```
RidgeModel.fit(X,y)
```

```
Yhat=RidgeModel.predict(X)
```

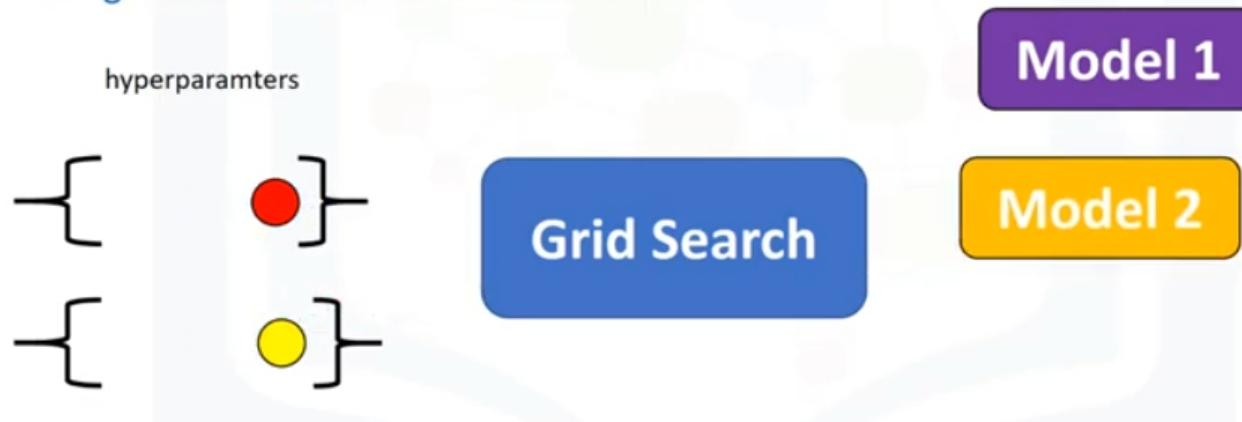
Lo que hacemos es tomar diferentes valores de alfa y entrenamos el modelo y con el conjunto de validación luego calculamos el Rsquared para los diferentes valores.



Debemos elegir el valor de alfa que arroje un mayor valor en el R2

▼ Grid Search

- Nos permite buscar entre diferentes parametros para elegir los mejores para nuestro modelo, esto busca los hiperparametros.
- In the last section, the term alpha in Ridge regression is called a hyperparameter
- Scikit-lean has a means of automatically iterating over these hyperparameters using cross-validation called Grid Search



Para este caso debemos crear un diccionario con los parametros donde la clave sera el alfa y los valores los respectivos parametros a probar.

```
parameters = [ { 'alpha' : [1, 10, 100, 1000] } ]
```

Alpha	1	10	100	1000
-------	---	----	-----	------

Ridge()

Grid search nos devuelve los puntajes para cada uno de los parametros, y en este caso observamos el que mayor valor nos arroje

Grid Search CV

Alpha	1	10	100	1000
R^2	0.74	0.35	0.073	0.008

Si lo queremos hacer en Python debemos poner lo siguiente:

```

 sklearn.linear_model import Ridge
DA0101EN_Grid Search 4:34
sklearn.model_selection import GridSearchCV
Ver más ta... C

parameters1= [{"alpha": [0.001,0.1,1, 10, 100, 1000,10000,100000,100000]}]

RR=Ridge()

Grid1 = GridSearchCV(RR, parameters1,cv=4)

Grid1.fit(x_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y_data)

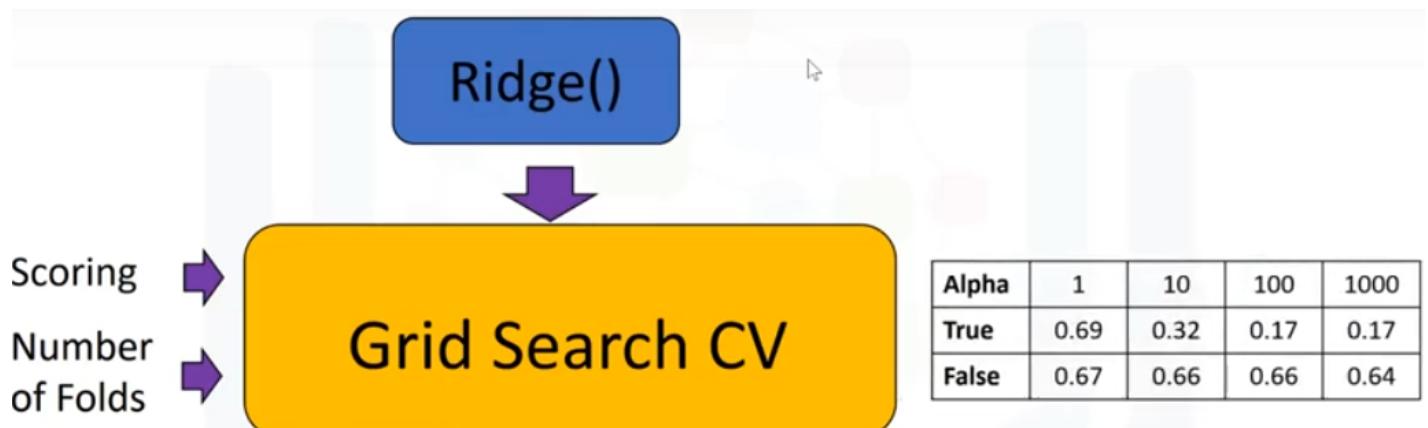
Grid1.best_estimator_

scores = Grid1.cv_results_
scores['mean_test_score']

parameters = [{ 'alpha': [1, 10, 100, 1000], 'normalize': [True, False] }]

```

Alpha	1	10	100	1000
Normalize	True	True	True	True
	False	False	False	False



```

DA0101EN - Grid Search 4:34
Ver más ta... C

In[1]: from sklearn.linear_model import Ridge
        from sklearn.model_selection import GridSearchCV

parameters2= [{"alpha": [0.001,0.1,1, 10, 100], 'normalize': [True, False]}]

RR=Ridge()

Grid1 = GridSearchCV(RR, parameters2, cv=4)

Grid1.fit(x_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y_data)

Grid1.best_estimator_

scores = Grid1.cv_results_

```

```

DA0101EN - Grid Search 4:34
Ver más ta... Compartir

param mean_val mean_test inzip(scores['params'],scores['mean_test_score'],scores['mean_train_score']):
    print(param, "R^2 on test data:", mean_val,"R^2 on train data:" ,mean_test)

```

```

{'alpha': 0.001, 'normalize': True} R^2 on tesst data: 0.66605547293 R^2 on train data: 0.814001968709
{'alpha': 0.001, 'normalize': False} R^2 on tesst data: 0.665488366584 R^2 on train data: 0.814002698794
{'alpha': 0.1, 'normalize': True} R^2 on tesst data: 0.694175625356 R^2 on train data: 0.810546768311
{'alpha': 0.1, 'normalize': False} R^2 on tesst data: 0.665488937796 R^2 on train data: 0.814002698794
{'alpha': 1, 'normalize': True} R^2 on tesst data: 0.690486934584 R^2 on train data: 0.749104440368
{'alpha': 1, 'normalize': False} R^2 on tesst data: 0.665494127178 R^2 on train data: 0.814002698472
{'alpha': 10, 'normalize': True} R^2 on tesst data: 0.321376875232 R^2 on train data: 0.341856042902
{'alpha': 10, 'normalize': False} R^2 on tesst data: 0.665545680812 R^2 on train data: 0.81400266666
{'alpha': 100, 'normalize': True} R^2 on tesst data: 0.0170551710263 R^2 on train data: 0.0496044796826
{'alpha': 100, 'normalize': False} R^2 on tesst data: 0.666029359996 R^2 on train data: 0.813999791851
{'alpha': 1000, 'normalize': True} R^2 on tesst data: -0.0301961745066 R^2 on train data: 0.005184451599
{'alpha': 1000, 'normalize': False} R^2 on tesst data: 0.668968215369 R^2 on train data: 0.813870488264
{'alpha': 10000, 'normalize': True} R^2 on tesst data: -0.0351687400461 R^2 on train data: 0.000520784757979
{'alpha': 10000, 'normalize': False} R^2 on tesst data: 0.673346359342 R^2 on train data: 0.812583743226
{'alpha': 100000, 'normalize': True} R^2 on tesst data: -0.0356685844558 R^2 on train data: 5.2101975528e-05
{'alpha': 100000, 'normalize': False} R^2 on tesst data: 0.657818838432 R^2 on train data: 0.789541446486
{'alpha': 100000, 'normalize': True} R^2 on tesst data: -0.0356685844558 R^2 on train data: 5.2101975528e-05
{'alpha': 100000, 'normalize': False} R^2 on tesst data: 0.657818838432 R^2 on train data: 0.789541446486

```

▼ Lab 5

Question #1):

Use the function "train_test_split" to split up the dataset such that 40% of the data samples will be utilized for testing. Set the parameter "random_state" equal to zero. The output of the function should be the following: "x_train1", "x_test1", "y_train1" and "y_test1".

```
# Write your code below and press Shift+Enter to execute
x_train1, x_test1, y_train1, y_test1 = train_test_split(x_data, y_data, test_size=0.4, random_state=0)
print("number of test samples :", x_test1.shape[0])
print("number of training samples:",x_train1.shape[0])

number of test samples : 81
number of training samples: 120
```

Question #2):

Find the R² on the test data using 40% of the dataset for testing.

```
# Write your code below and press Shift+Enter to execute
x_train1, x_test1, y_train1, y_test1 = train_test_split(x_data, y_data, test_size=0.4, random_state=0)
lre.fit(x_train1[['horsepower']],y_train1)
lre.score(x_test1[['horsepower']],y_test1)

0.7139364665406973

▼ Click here for the solution
x_train1, x_test1, y_train1, y_test1 = train_test_split(x_data, y_data, test_size=0.4,
random_state=0)
lre.fit(x_train1[['horsepower']],y_train1)
lre.score(x_test1[['horsepower']],y_test1)
```

Question #3):

Calculate the average R² using two folds, then find the average R² for the second fold utilizing the "horsepower" feature:

```
# Write your code below and press Shift+Enter to execute
Rc=cross_val_score(lre,x_data[['horsepower']], y_data,cv=2)
Rc.mean()
```

0.5166761697127429

▼ Click here for the solution

```
Rc=cross_val_score(lre,x_data[['horsepower']], y_data,cv=2)
Rc.mean()
```

Question #4a):

We can perform polynomial transformations with more than one feature. Create a "PolynomialFeatures" object "pr1" of degree two.

```
: # Write your code below and press Shift+Enter to execute  
pr1=PolynomialFeatures(degree=2)
```

Question #4b):

Transform the training and testing samples for the features 'horsepower', 'curb-weight', 'engine-size' and 'highway-mpg'. Hint: use the method "fit_transform".

```
# Write your code below and press Shift+Enter to execute  
x_train_pr1=pr1.fit_transform(x_train[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']])  
  
x_test_pr1=pr1.fit_transform(x_test[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']])
```

Question #4c):

How many dimensions does the new feature have? Hint: use the attribute "shape".

```
]: # Write your code below and press Shift+Enter to execute  
x_train_pr1.shape  
  
]: (110, 15)
```

Question #4d):

Create a linear regression model "poly1". Train the object using the method "fit" using the polynomial features.

```
: # Write your code below and press Shift+Enter to execute  
poly1=LinearRegression().fit(x_train_pr1,y_train)
```

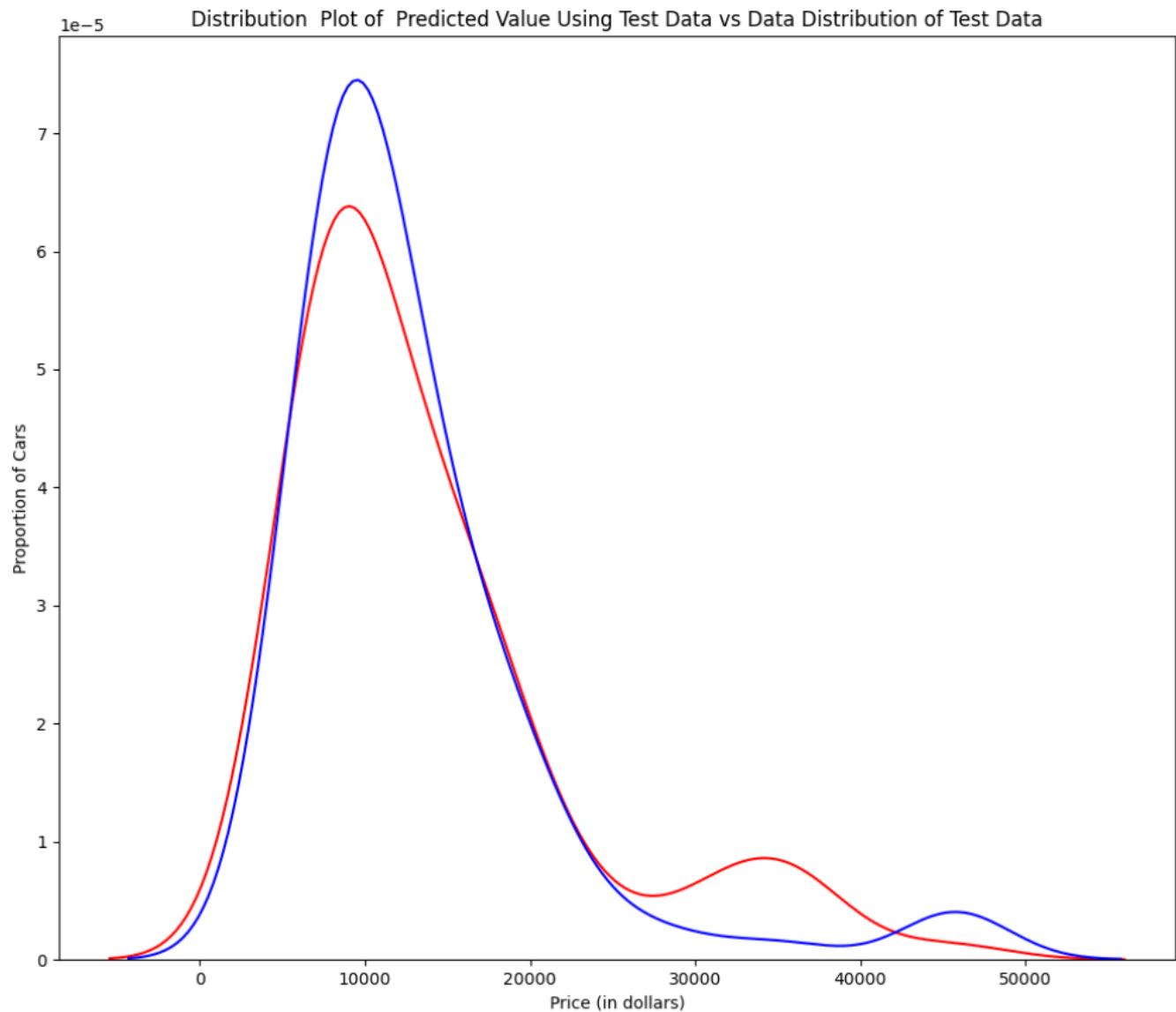
Question #4e):

Use the method "predict" to predict an output on the polynomial features, then use the function "DistributionPlot" to display the distribution of the predicted test output vs. the actual test data.

```
# Write your code below and press Shift+Enter to execute
yhat_test1=poly1.predict(x_test_pr1)

Title='Distribution Plot of Predicted Value Using Test Data vs Data Distribution of Test Data'

DistributionPlot(y_test, yhat_test1, "Actual Values (Test)", "Predicted Values (Test)", Title)
```



Question #4f):

Using the distribution plot above, describe (in words) the two regions where the predicted prices are less accurate than the actual prices.

```
: # Write your code below and press Shift+Enter to execute  
#El valor predecido es mayor al actual, donde el precio de los autos es 10.000 y el precio  
#es menor al predecir en el rango de 30.000 a 40.000, por lo tanto el modelo no es preciso  
#en esos rangos.
```

Question #5):

Perform Ridge regression. Calculate the R^2 using the polynomial features, use the training data to train the model and use the test data to test the model. The parameter alpha should be set to 10.

```
# Write your code below and press Shift+Enter to execute  
RidgeModel = Ridge(alpha=10)  
RidgeModel.fit(x_train_pr, y_train)  
RidgeModel.score(x_test_pr, y_test)
```

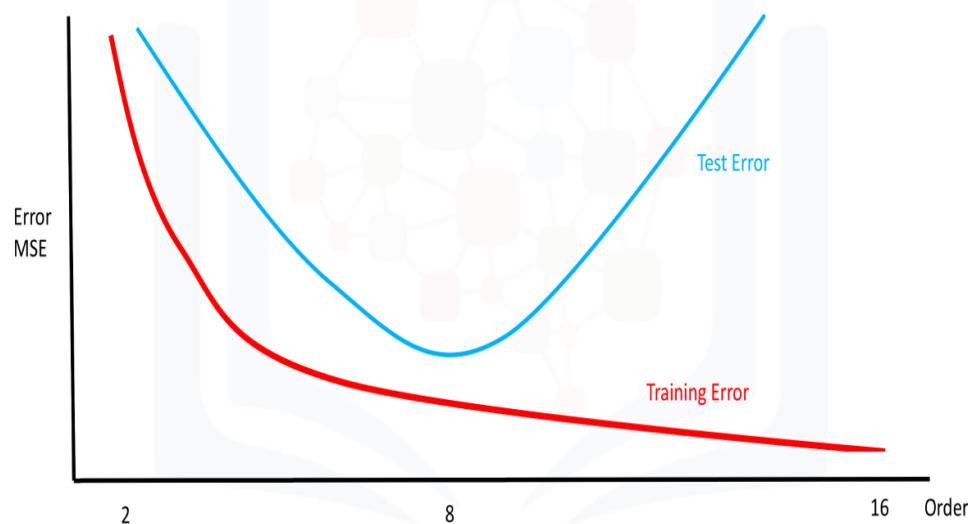
```
0.5418576440208844
```

▼ Graded Review Questions 5

Question 1

1/1 point (graded)

In the following plot, the vertical axis shows the mean square error and the horizontal axis represents the order of the polynomial. The red line represents the training error the blue line is the test error. What is the best order of the polynomial given the possible choices in the horizontal axis?



2

8

16



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 2

1/1 point (graded)

What is the correct use of the "train_test_split" function such that 40% of the data samples will be utilized for testing; the parameter "random_state" is set to zero; and the input variables for the features and targets are `x_data`, `y_data` respectively?

`train_test_split(x_data, y_data, test_size=0, random_state=0.4)`

`train_test_split(x_data, y_data, test_size=0.4, random_state=0)`

`train_test_split(x_data, y_data)`



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Questions 3

1/1 point (graded)

What is the output of `cross_val_score(lre, x_data, y_data, cv=2)` ?

The predicted values of the test data using cross-validation.

The average R^2 on the test data for each of the two folds.

This function finds the free parameter alpha.



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 4

1/1 point (graded)

What is the code to create a ridge regression object "RR" with an alpha term equal 10?

 RR=LinearRegression(alpha=10) RR=Ridge(alpha=10) RR=Ridge(alpha=1)

[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 5

1/1 point (graded)

What dictionary value would we use to perform a grid search for the following values of alpha: 1,10, 100? No other parameter values should be tested.

 alpha=[1,10,100] {'alpha': [1,10,100]} {'alpha': [0.001,0.1,1, 10, 100, 1000,10000,100000,100000],'normalize':[True,False]}]

[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

✓ Correct (1/1 point)

▼ Final Exam

Question 1

1/1 point (graded)

What does the following command do?

```
df.dropna(subset=["price"], axis=0)
```

- Drop the "not a number" values from the column "price".

- Drop the row "price".

- Rename the dataframe "price".



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

- Correct (1/1 point)

Question 2

1/1 point (graded)

How would you provide many of the summary statistics for all the columns in the dataframe "df"?

- df.describe(include = "all")

- df.head()

- type(df)

- df.shape



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

- Correct (1/1 point)

Question 3

1/1 point (graded)

How would you find the shape of the dataframe df?

- df.describe()
- df.head()
- type(df)
- df.shape



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 4

1/1 point (graded)

What task does the following command, df.to_csv("A.csv"), perform:

- Change the name of the column to "A.csv".
- Load the data from a csv file called "A" into a dataframe.
- Save the dataframe df to a csv file called "A.csv".



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

✓ Correct (1/1 point)

Question 5

1/1 point (graded)

What task does the following line of code perform?

```
result = np.linspace(min(df["city-mpg"]), max(df["city-mpg"]), 5)
```

- Builds a bin array ranging from the smallest value to the largest value of "city-mpg" in order to build 4 bins of equal length.
- Builds a bin array ranging from the smallest value to the largest value of "city-mpg" in order to build 5 bins of equal length.
- Determines which bin each value of "city-mpg" belongs to.



Show answer

Submit

You have used 2 of 2 attempts

- Correct (1/1 point)

Question 6

1/1 point (graded)

What task does the following line of code perform:

```
df['peak-rpm'].replace(np.nan, 5,inplace=True)
```

- Replace the "not a number" values with 5 in the column 'peak-rpm'.
- Rename the column 'peak-rpm' to 5.
- Add 5 to the dataframe.



Save | Show answer

Submit

You have used 1 of 2 attempts

- Correct (1/1 point)

Question 7

0/1 point (graded)

How do you "one-hot encode" the column 'fuel-type' in the dataframe df?

 pd.get_dummies(df["fuel-type"]) df.mean(["fuel-type"]) df[df["fuel-type"]==1]=1

✗

[Show answer](#)

You have used 2 of 2 attempts

✗ Incorrect (0/1 point)

Question 8

0/1 point (graded)

What does the vertical axis on a scatterplot represent?

 Independent variable Dependent variable[Show answer](#)

You have used 1 of 1 attempt

✗ Incorrect (0/1 point)

Question 9

Question 10

1/1 point (graded)

If we have 10 columns and 100 samples, how large is the output of df.corr()?

 10 x 100 10 x 10 100x100 100x100[Show answer](#)[Submit](#)

You have used 2 of 2 attempts

Correct (1/1 point)

Question 11

1/1 point (graded)

What is the largest possible element resulting in the following operation "df.corr()"?

 100 1000 1[Save](#) | [Show answer](#)[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 12

1/1 point (graded)

If the Pearson Correlation of two variables is zero:

The two variable have zero mean.

The two variables are not correlated.



[Show answer](#)

[Submit](#)

You have used 1 of 1 attempt

Correct (1/1 point)

Question 13

1/1 point (graded)

If the p-value of the Pearson Correlation is 1:

The variables are correlated.

The variables are not correlated.

None of the above.



[Show answer](#)

[Submit](#)

You have used 2 of 2 attempts

Correct (1/1 point)

Question 14

1/1 point (graded)

What does the following line of code do: lm = LinearRegression()?

Fit a regression object "lm".

Create a linear regression object.

Predict a value.



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 15

1/1 point (graded)

If the predicted function is:

$$\hat{Y} = a + b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X_4$$

The method is:

Polynomial Regression

Multiple Linear Regression



[Show answer](#)

[Submit](#)

You have used 1 of 1 attempt

Correct (1/1 point)

Question 16

1/1 point (graded)

What steps do the following lines of code perform:

```
Input=[('scale',StandardScaler()),('model',LinearRegression())]
```

```
pipe=Pipeline(Input)
```

```
pipe.fit(Z,y)
```

```
y=pipe.predict(Z)
```

Standardize the data, then perform a polynomial transform on the features Z.

Find the correlation between Z and y.

Standardize the data, then perform a prediction using a linear regression model using the features Z and targets y.



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 17

1/1 point (graded)

What is the maximum value of R^2 that can be obtained?

10

1

0



[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 18

1/1 point (graded)

We create a polynomial feature `PolynomialFeatures(degree=2)`. What is the order of the polynomial?

 0 1 2

[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 19

1/1 point (graded)

You have a linear model. The average R^2 value on your training data is 0.5. You perform a 100th order polynomial transform on your data, then use these values to train another model. Your average R^2 is 0.99. Which comment is correct?

 100th order polynomial will work better on unseen data. You should always use the simplest model. The results on your training data is not the best indicator of how your model performs. You should use your test data to get a better idea.

[Save](#) | [Show answer](#)

[Submit](#)

You have used 1 of 2 attempts

Correct (1/1 point)

Question 20

0/1 point (graded)

You train a ridge regression model. You get a R^2 of 1 on your validation data and you get a R^2 of 0.5 on your training data. What should you do?

 Nothing. Your model performs flawlessly on your validation data. Your model is under fitting perform a polynomial transform. Your model is overfitting, increase the parameter alpha.[Show answer](#)[Submit](#)

You have used 2 of 2 attempts

[Colab paid products - Cancel contracts here](#) 1s completed at 5:11 PM