Análisis y visualización de datos con Python

Dos cursos en un Notebook

LUIS ALEXIS ROJAS RONDAN

Contenido

[1.0 Análisis de datos con Python 2](#_Toc170170790)

[1.1 Módulo 1 2](#_Toc170170791)

[1.1.1 Paquetes de Python 2](#_Toc170170792)

[1.1.2 Importandos y exportando datos 2](#_Toc170170793)

[1.1.3 Entendiendo los datos 3](#_Toc170170794)

[1.1.4 Tipos de datos en pandas 3](#_Toc170170795)

[1.2 Módulo 2 3](#_Toc170170796)

[1.2.1 Missing values 3](#_Toc170170797)

[1.2.2 Data formating 4](#_Toc170170798)

[1.2.3 Correccionando datos 4](#_Toc170170799)

[1.2.4 Normalizacion 5](#_Toc170170800)

[1.2.5 Binnings 5](#_Toc170170801)

[1.3 Módulo 3 5](#_Toc170170802)

[1.4 Módulo 4 5](#_Toc170170803)

[1.5 Módulo 5 5](#_Toc170170804)

[2.0 Visualización de datos con Python 5](#_Toc170170805)

[2.1 Módulo 1 5](#_Toc170170806)

[2.2 Módulo 2 5](#_Toc170170807)

[2.3 Módulo 3 5](#_Toc170170808)

[2.4 Módulo 4 5](#_Toc170170809)

[2.5 Módulo 5 5](#_Toc170170810)

# 1.0 Análisis de datos con Python

## 1.1 Módulo 1

### 1.1.1 Paquetes de Python

Librerías de Python más utilizadas en el análisis de datos

|  |  |  |
| --- | --- | --- |
| Computación científica🡪 | Pandas🡪 | Estructuras de datos y herramientas |
| Numpy🡪 | Arreglos y matrices |
| SciPy🡪 | Integrales, ecuaciones diferenciales y optimizacion |

|  |  |  |
| --- | --- | --- |
| Visualización🡪 | Matplotlib🡪 | Plots y gráficos |
| Seaborn🡪 | Mapas de calor, tiempos de series y violín plots |

|  |  |  |
| --- | --- | --- |
| Algoritmos🡪 | Scikit-learn🡪 | Machine learning, regresión lineal, clasificación y mas |
| Stats models🡪 | Exploracion de datos, modelos de estimación estadística y test de estadistica |

### 1.1.2 Importandos y exportando datos

Propiedades de datos de entrada

* Formatos (csv, json, xlsx, hdf,…)
* Ubicación de archivos
  + Computadora
  + Internet

***Importando datos de un archivo .csv contenido en una página web***

import pandas as pd

url = ‘http://example.com’

df = pd.read\_csv(url, header = None)

df.head(n) 🡨muestra las primeras n filas de df

df.tail(n) 🡨Muestra las últimas n filas del df

***Remplazar los header (títulos de cada columna)***

header = [‘valor1, ‘valor2’ ,’valor3’]

df.header = header

***Exportar datos del df como .csv***

path = ‘./archivo.csv’

df.to\_csv(path)

### 1.1.3 Entendiendo los datos

Pasos para analizar los datos

* Entender los datos antes de procesarlos
* Chequear los datos
  + Tipo de datos
  + Distribucion
* Localizar usos potenciales de los datos

### 1.1.4 Tipos de datos en pandas

Importante, la tabla muestra los tipos de datos de los df de pandas y su comparación con los datos nativos de Python.

|  |  |  |
| --- | --- | --- |
| Pandas type | Nativo python | Descripción |
| object | string | Números y cadenas |
| Int64 | int | Números |
| Float64 | float | Números con decimales |
| Datatime64, timedata[ns] | N/A | Tiempo |

df.type

## 1.2 Módulo 2

### 1.2.1 Missing values

Valores perdidos, ocurre cuando los valores no coinciden con alguna variable.

Pueden representarse como: ‘?’, ‘N/A’, 0 o como una celda en blanco.

¿Qué hacer con los missing data?

* Checar la colección de datos
* Borrar valores perdidos
  + Borrar la variable
  + Borrar la entrada de datos
* Remplazar valores perdidos
  + Remplazar con promedio
  + Remplazar con frecuencia
  + Remplazar con otras funciones
* Dejar los valores perdido

dataframe.dropna()

axis = 0 🡨 borra las entradas por fila

axis = 1 🡨 borra las entradas por columna

df.dropna(subset=[‘price’], axis = 0, inplace=True)

***Remplzar***

Dataframe.replace(missing\_values, new\_values)

mean = df[‘normalized’].mean()

df[‘normalized’].replace(np.nan, mean)

### 1.2.2 Data formating

Un ejemplo de aplicar calculos entre columnas puede ser convertir de (mpg) a (L/100Km) en un dataset de carros.

|  |  |  |
| --- | --- | --- |
| Mpg |  | L/100Km |
| 21 | 11.2 |
| 21 | 11.2 |
| 19 | 12.4 |

df[‘mpg’] = 235/df[‘mpg’]

df.rename(column=(‘mpg’:’L/100Km’), inplace=True)

### 1.2.3 Correccionando datos

df.dtype() 🡨 Para identificar el tipo de datos se usa el commando

df.astype() 🡨 Para convertir un tipo de dato

df[‘column’]=df[‘column’].astype(‘int’)🡨 Convertir un tipo de dato por columnas.

### 1.2.4 Normalizacion

Consiste básicamente en tener todos los datos en un rango semejante.

|  |  |  |
| --- | --- | --- |
| No-normalizado |  | Normalizado |
| * Diferentes rangos por columna * Difícil de comparar * Alta influencia en los resultados | * Rango de valores similares * Influencia similar intrínseca en modelos analíticos. |

Existen tres técnicas de normalización

1. **Simple Feature Scaling**

1. **Min-max**

1. **Z-score**

Normalización utilizando pandas.

Para el caso de **Simple Feature Scaling**

df[‘column’]=df[‘column’]/df[‘column’].max()

Para el caso de **Min-max**

df[‘column’]=(df[‘column’]-df[‘column’].min())/(df[‘column’].max()-df[‘column’].min())

Para el caso de **Z-score**

df[‘column’]=(df[‘column’]-df[‘column’].mean())/(df[‘column’].std()

### 1.2.5 Binnings

Agrupar valores en bins

Convertir variables numéricas en categóricas

Agrupar set de datos numéricos en un set de datos bins

Ejemplo práctico; la variable ‘price’ está en un rango de 5000 a 45000

|  |  |  |  |
| --- | --- | --- | --- |
| price | 5000 10000 12000 12000 | 30000 31000 39000 | 44000 45000 |
| bins | Low | Mid | High |

Bins=np.linespace(min(df.[‘price’]),max(df.[‘prace’]), 4)

Goup = [‘low’,’med’,’high’]

df[‘price\_binned’]=pd.cut(df[‘price’],bins,labels=group, include\_lowest=True)

La major opción para visualizar los bins es como histograma

### 1.2.6 Variables categóricas

Para convertir variables categóricas en analíticas es muy simple es asignar valores de 0 o 1 a variables analíticas como ejemplo:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| carro | fuel | … | Gas | Diésel |
| A | Gas | … | 1 | 0 |
| B | Diésel | … | 0 | 1 |
| C | Gas | … | 1 | 0 |
| D | Gas | … | 1 | 0 |

En pandas se usa el método

pd.get\_dummies()

convierte variables categoricas en 0 y 1

pd.get\_dummies(df[‘fuel’])

## 1.3 Módulo 3

### 1.3.1 Estadistica descriptiva

Se utiliza el método .describe() para obtener resumen estadístico del df

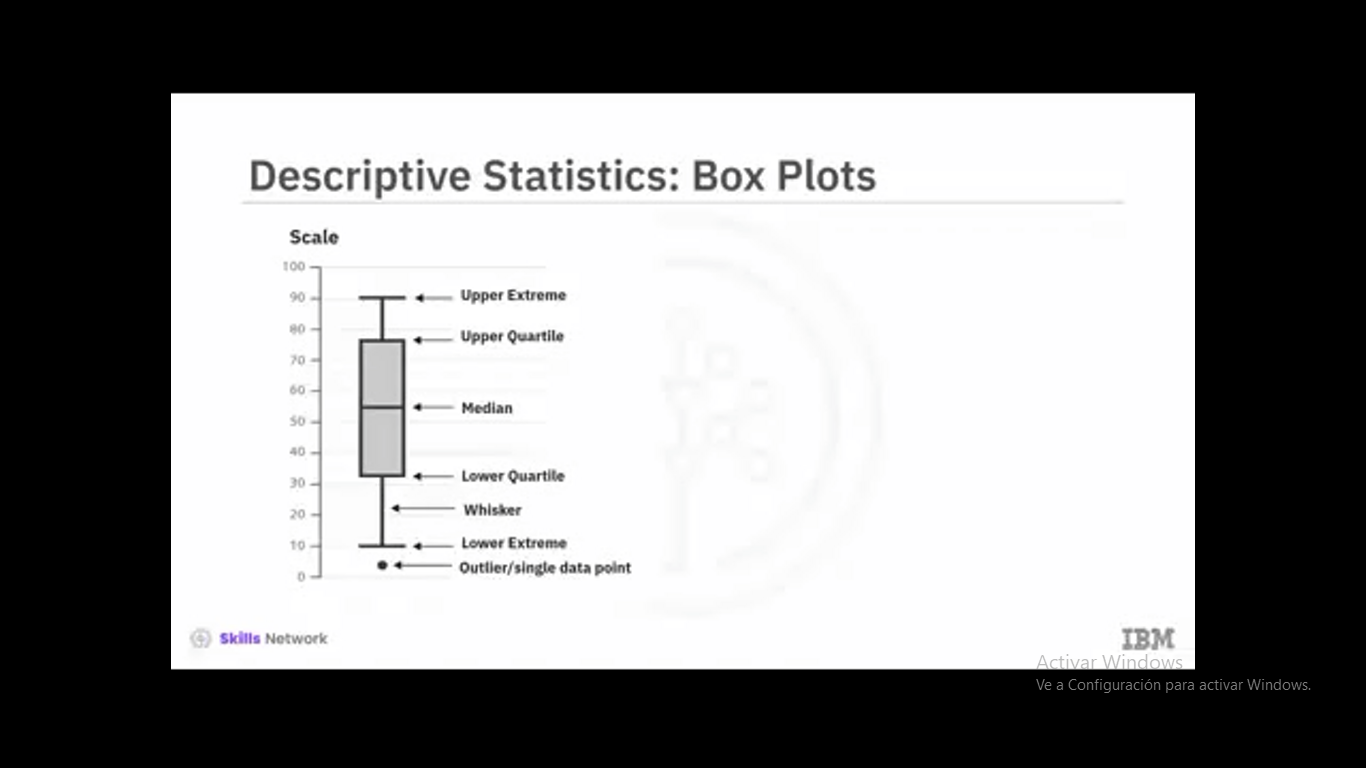
drive\_counts = df[‘drive’].value\_counts()

drive\_counts.rename(columns={‘drive’:’value\_counts’})

drive\_counts.index.name=’drive\_weels’

**Box plot**

Sns.boxplot(x=’drive\_weels’, y=’price’, data=df)



**Scatter Plot**

* Representa la observacion en un punto
* Relaciona dos valores
  1. Predecir / Variables independientes en el eje x
  2. Objetivo / Variables dependientes en el eje y

Y=df[‘price’]

X=df[‘engine-size’]

plt.scatter(X,Y)

plt.title(‘titulo’)

plt.xlabel(‘enigine-size’)

plt.ylabel(‘price’)

### 1.3.2 Gropu By

GROUP BY : Puede ser aplicada a variables categóricas, agrupa datos en categorías, agrupa simple o múltiples variables.

df\_test=df[[‘drive-wheels’, ‘body-style’, ‘price’]]

df\_grupo = df\_test.groupby([‘drive-wheels’, ‘body-style’], as\_index=False)

df\_grupo

### 1.3.3 Pivot

Una variable se muestra en columnas, y la otra variable se muestra solo en filas.

df\_pivot = df\_grup.pivot(index=’drive\_wheels’, columns = ‘body-style’)

**Heat Map**

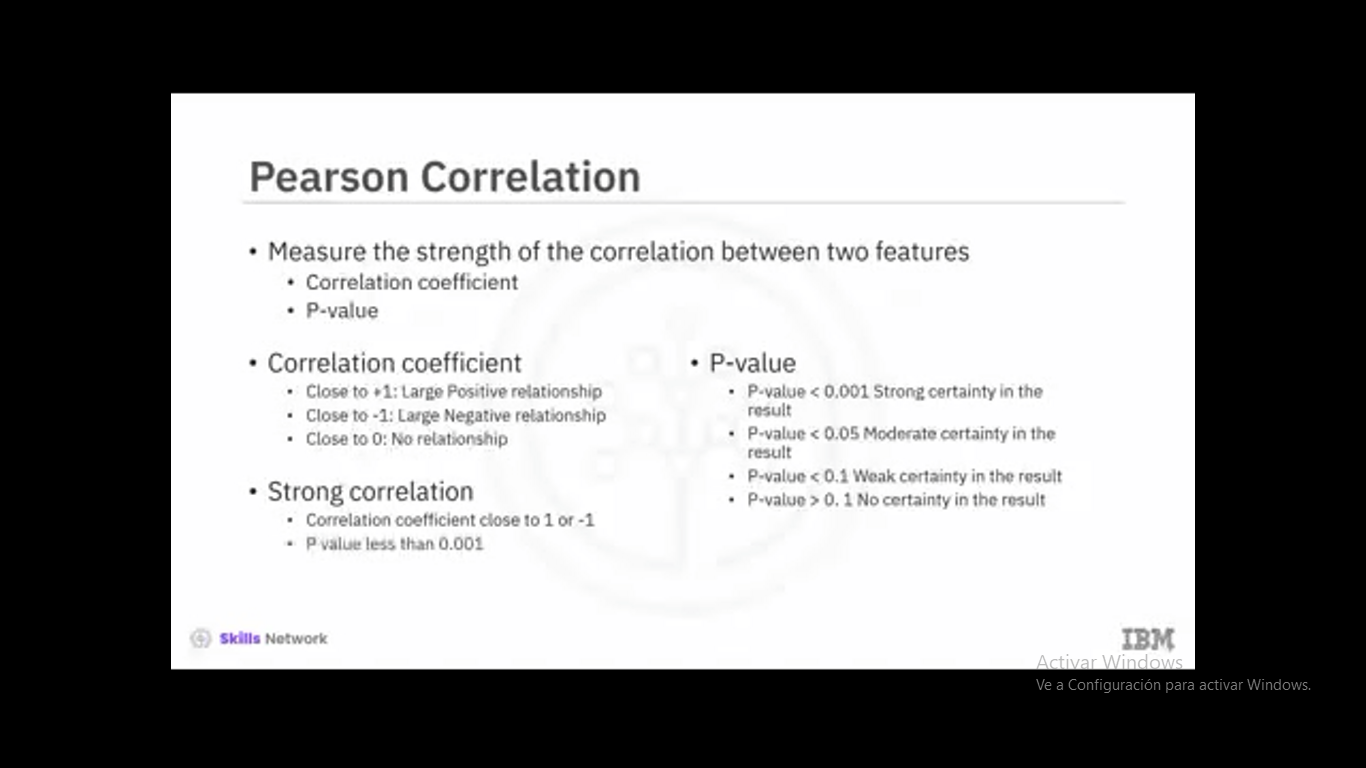
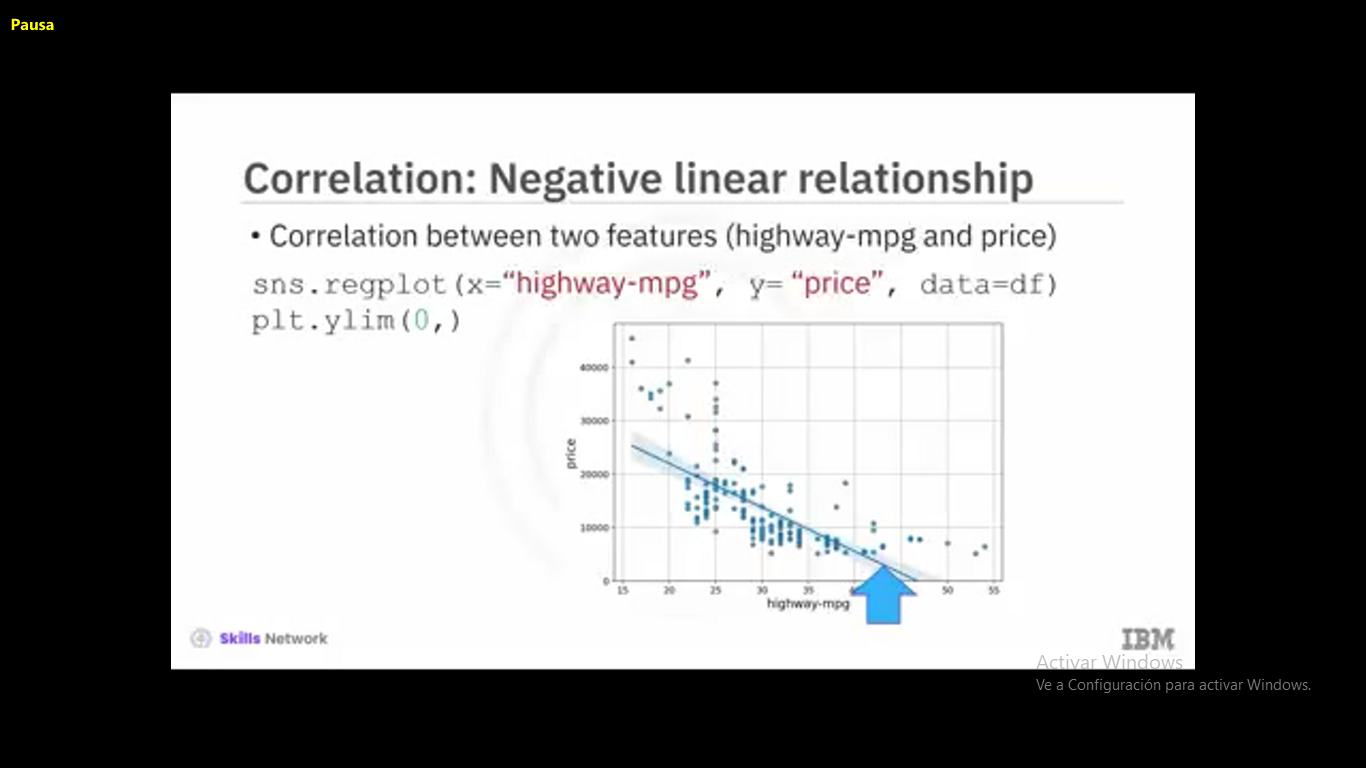
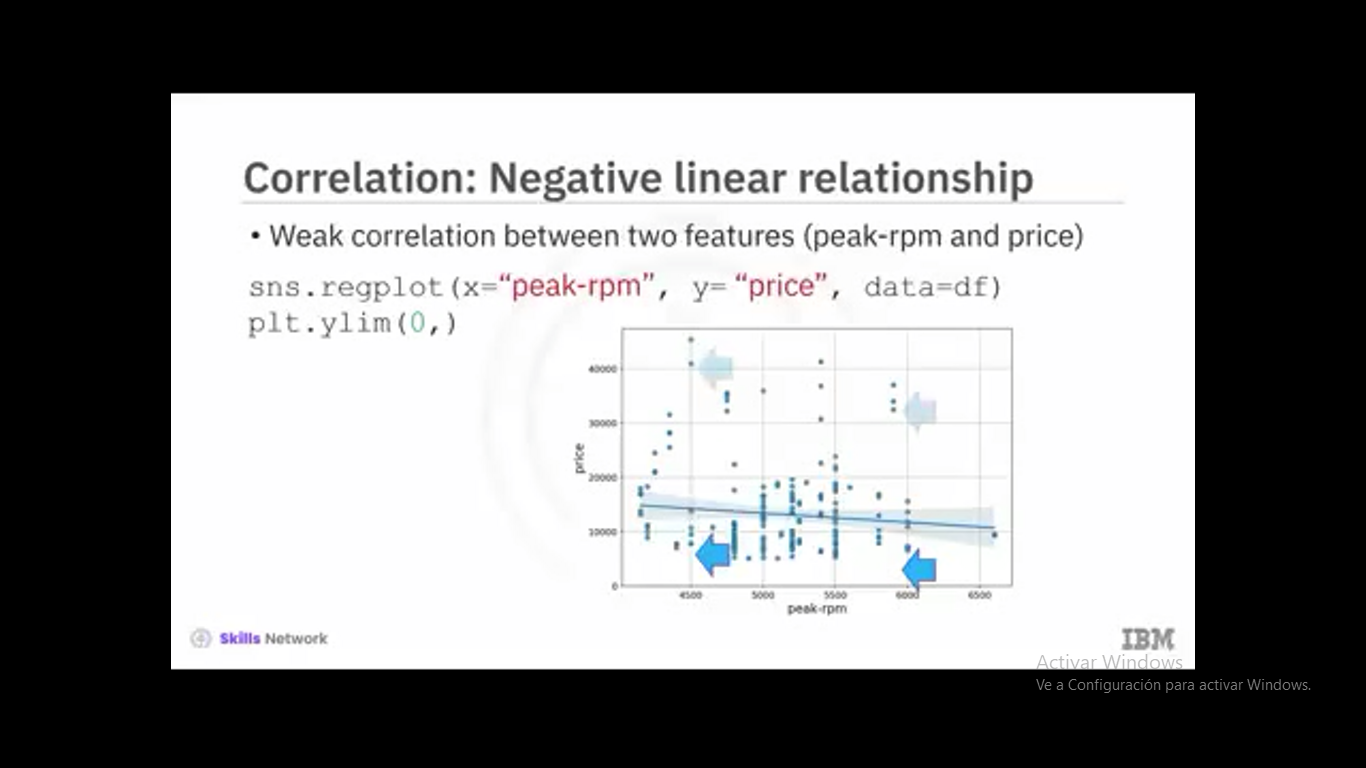
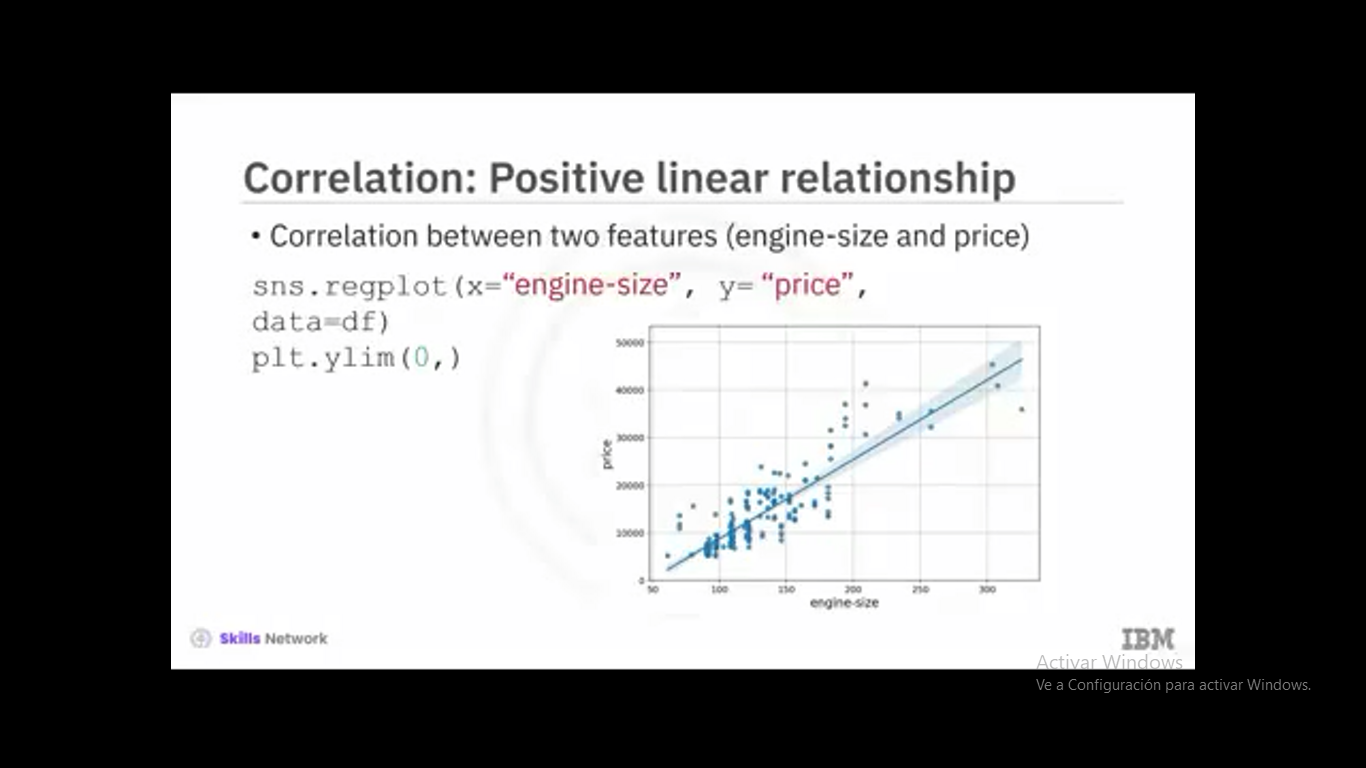
Plot valores objetivos contra múltiples variables.

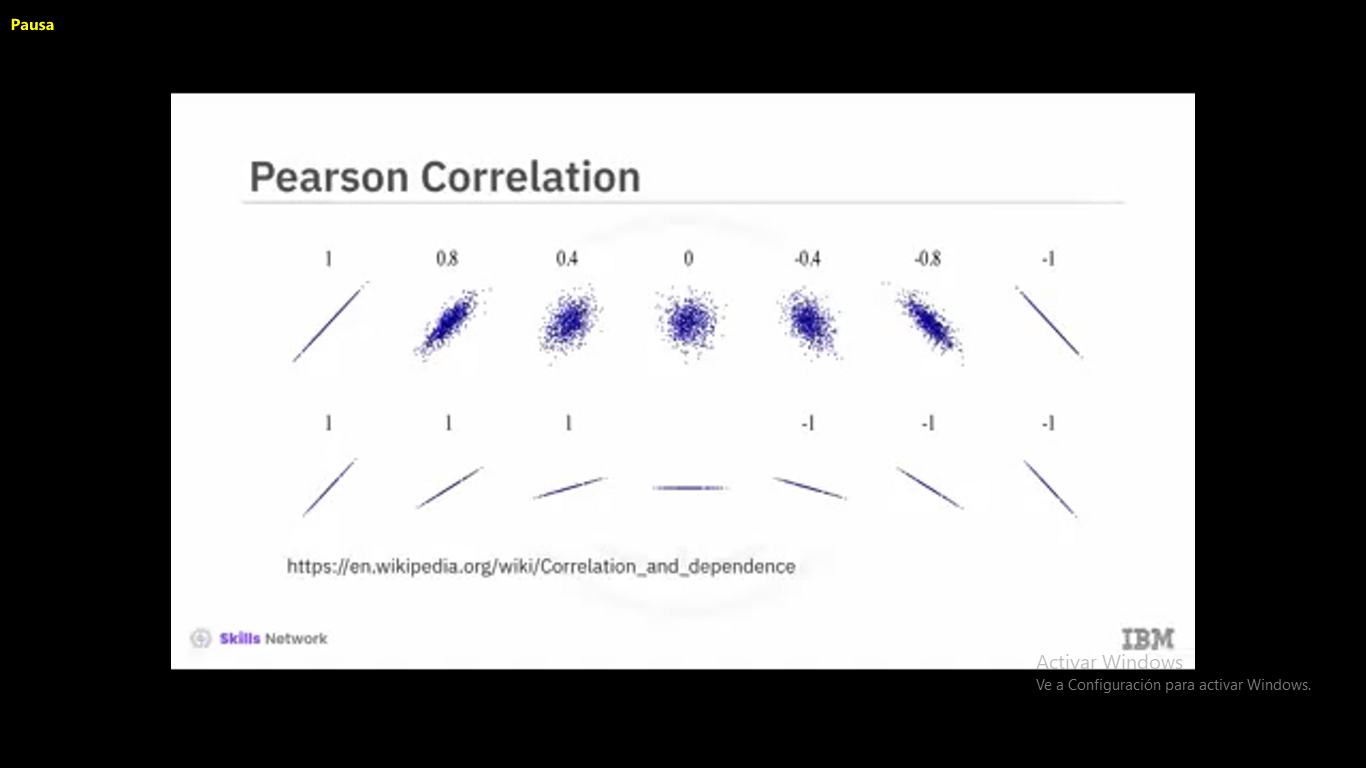
plt.pcolor(df\_pivot, cmap=’RdBu’)

plt.colorbar()

plt.show()

### 1.3.4 Correlación





## 1.4 Módulo 4

### 1.4.1 Regresion lineal simple

Predicción

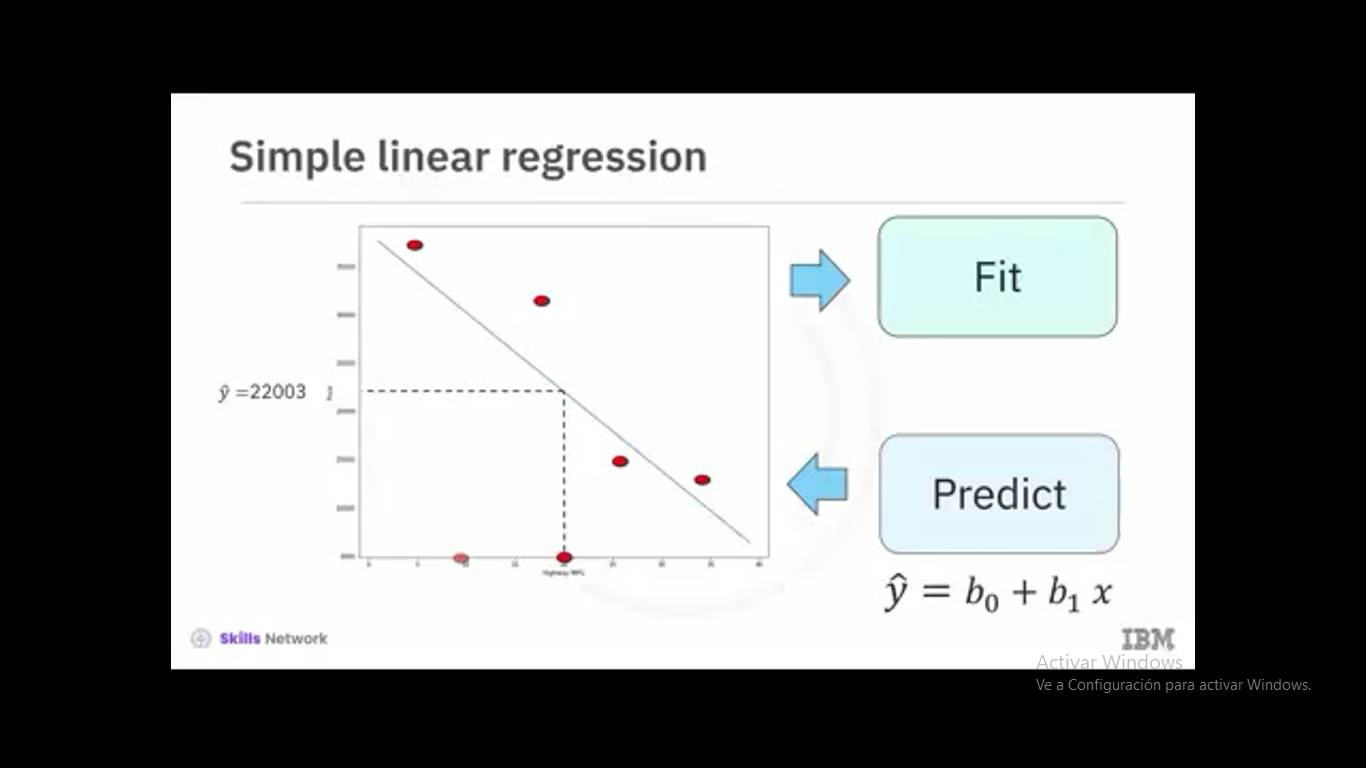
Variable independiente x1

RLS

Multiples variables independiente x1

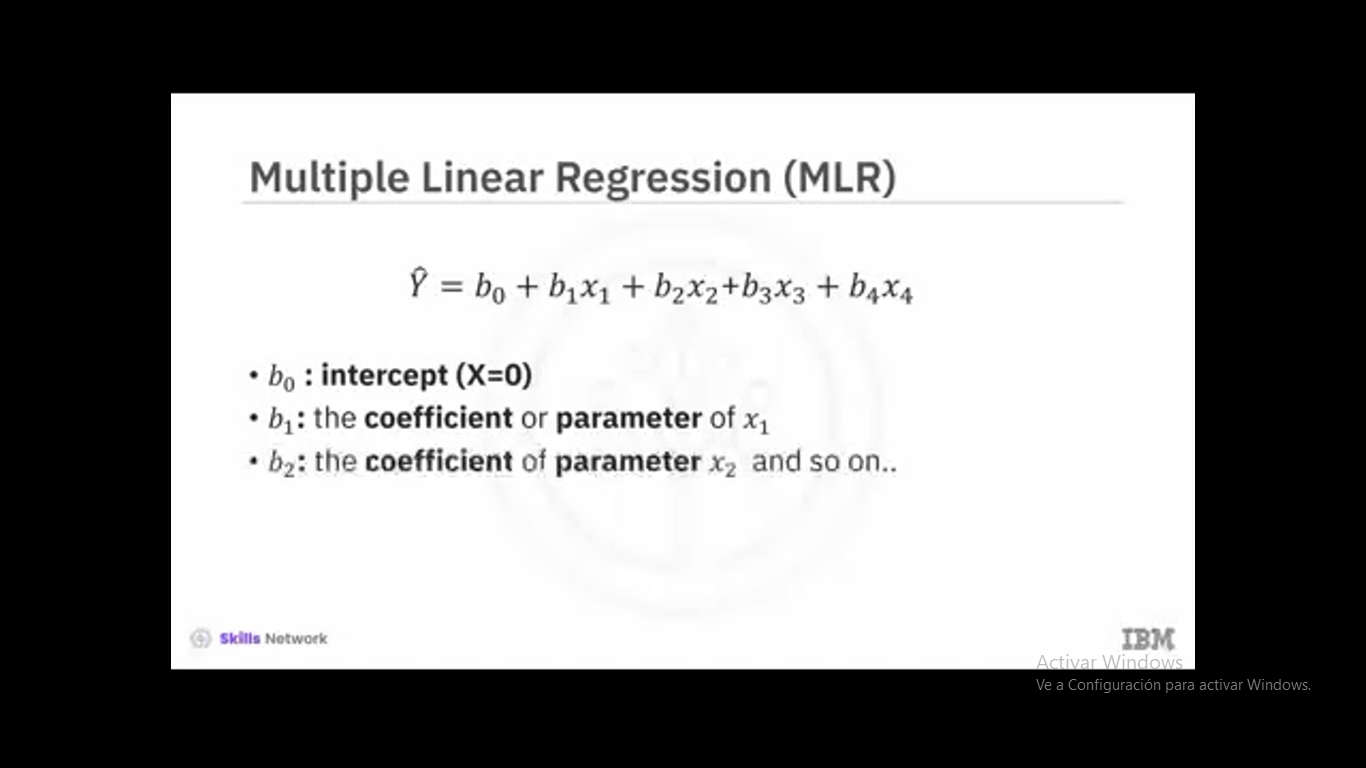
Predicción

RLM



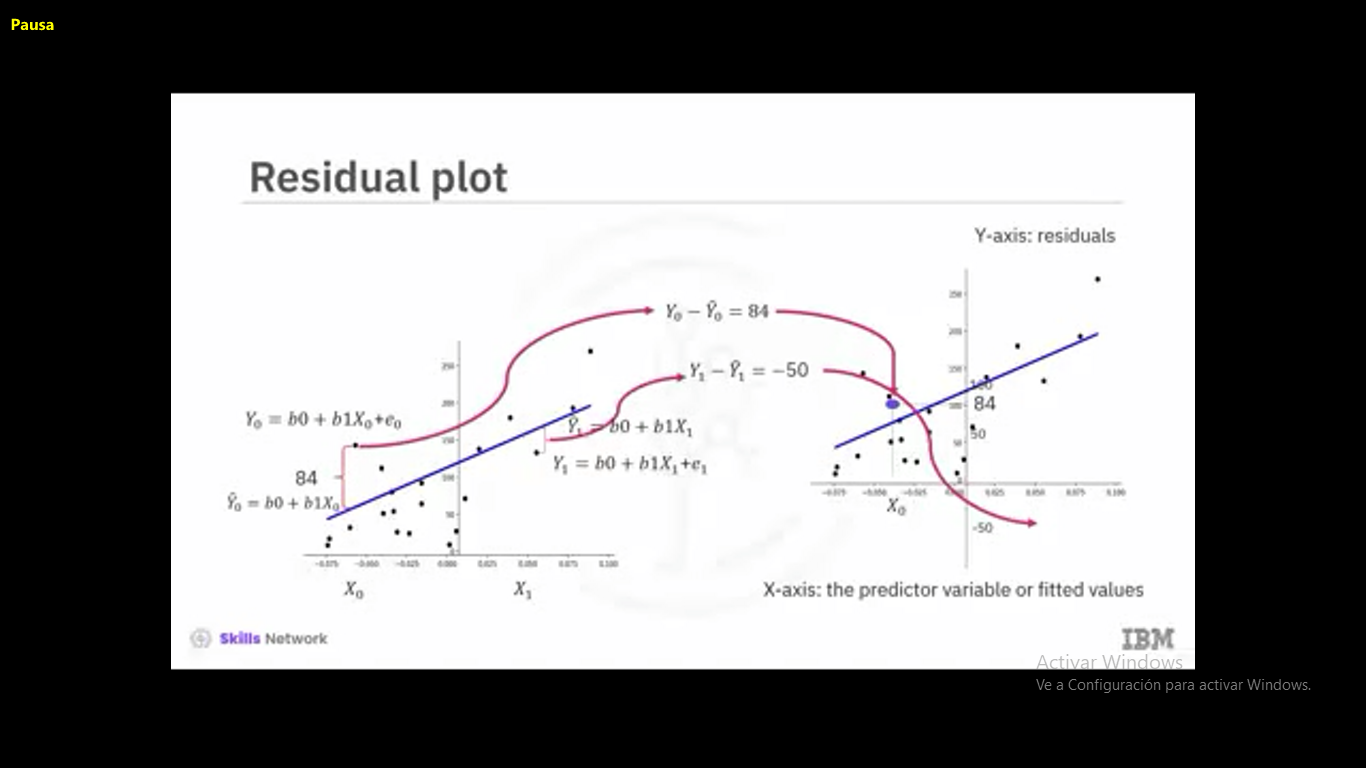
Ejemplos de cómo se implementan los códigos en GitHub ------------SLR.ipynb

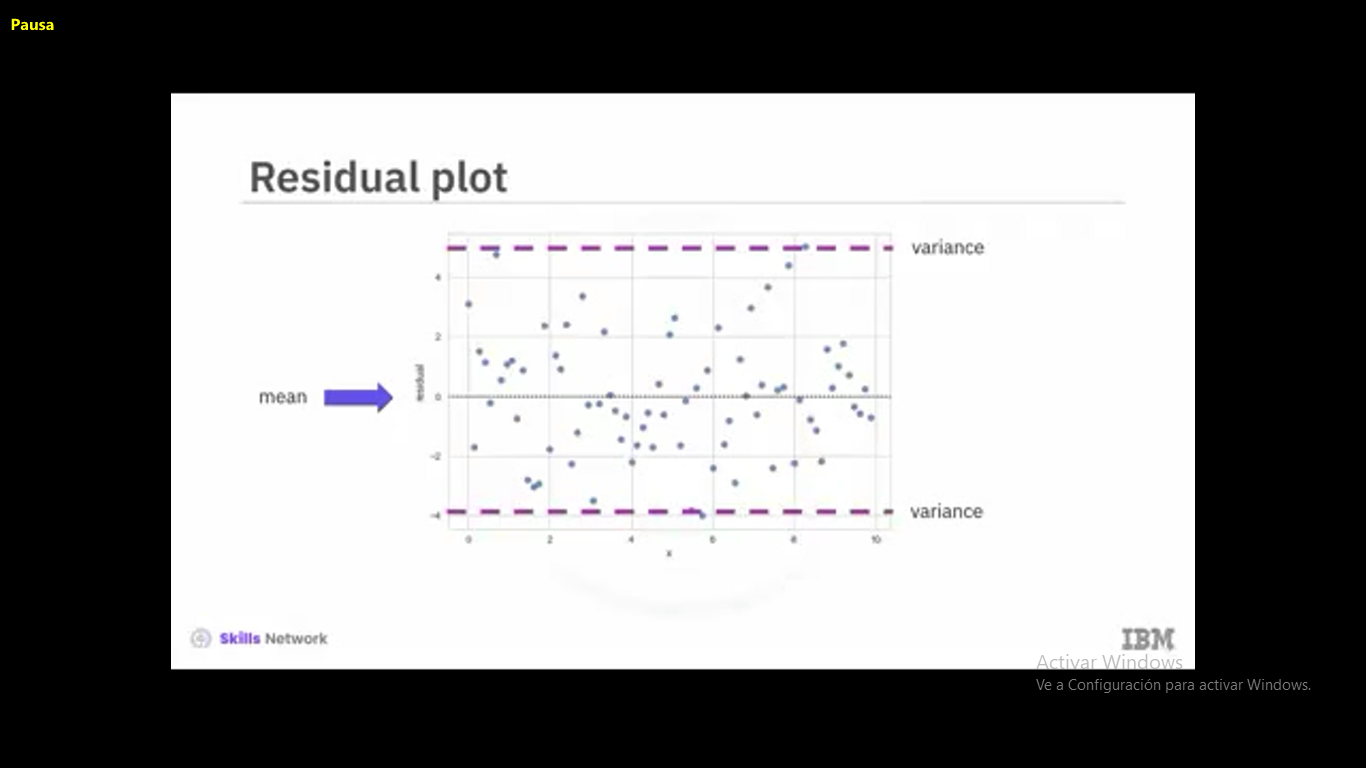
### 1.4.2 Regresion lineal multiple

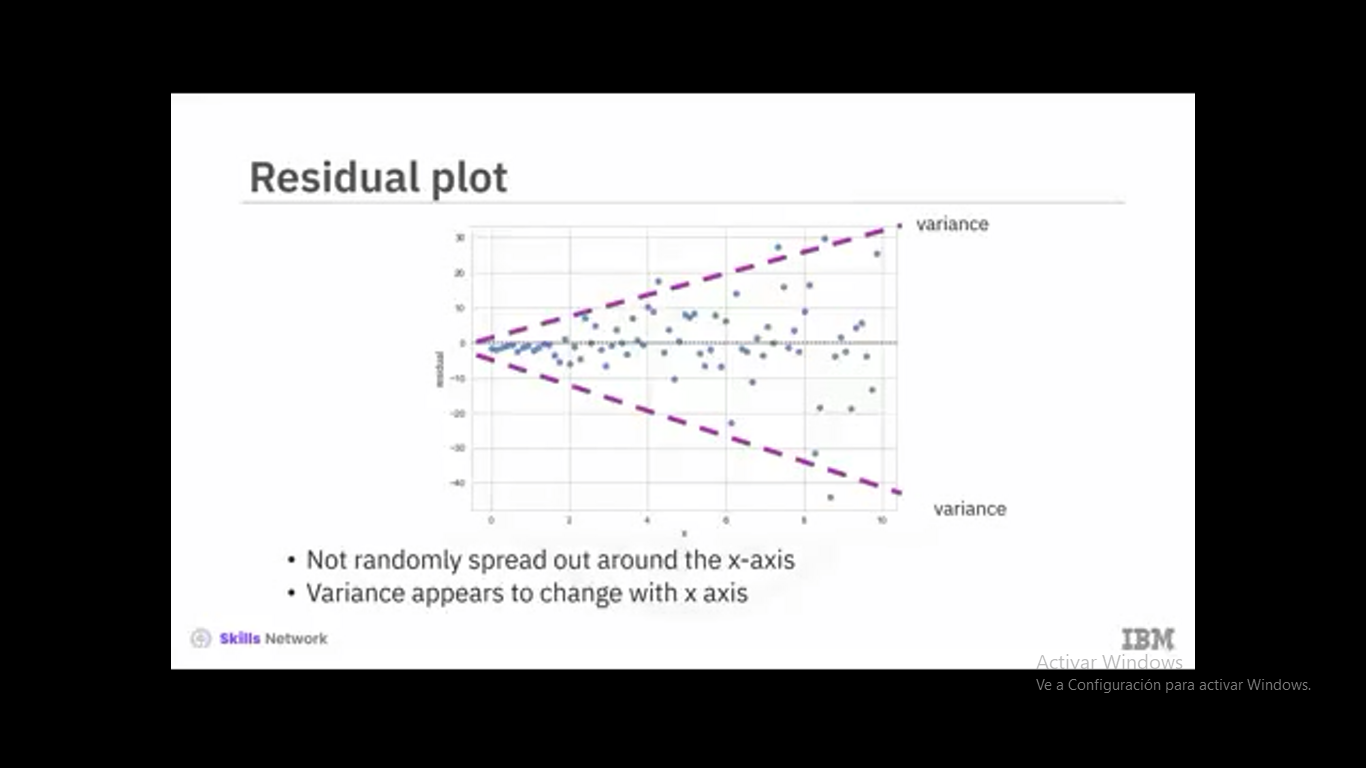


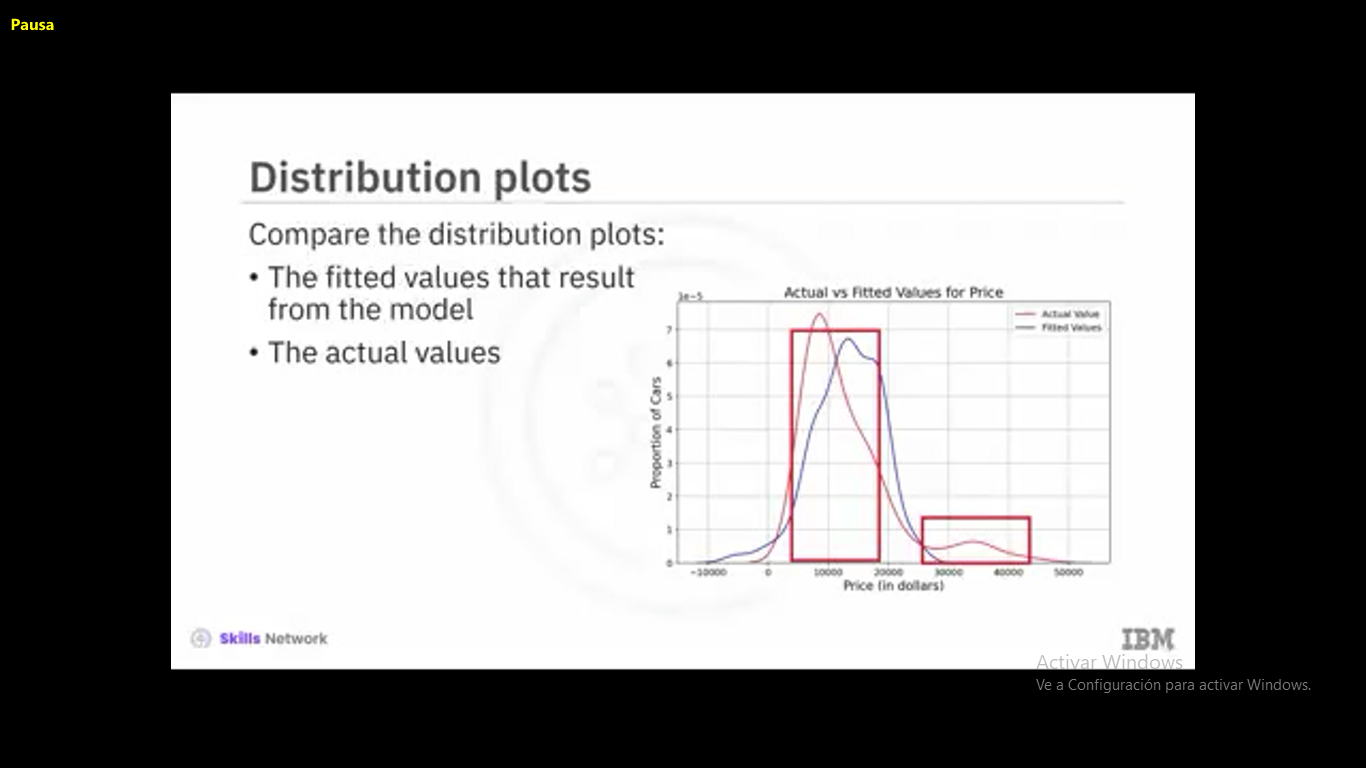
Ejemplos de cómo se implementan los códigos en GitHub ------------MLR.ipynb

### 1.4.3 Residual Plots

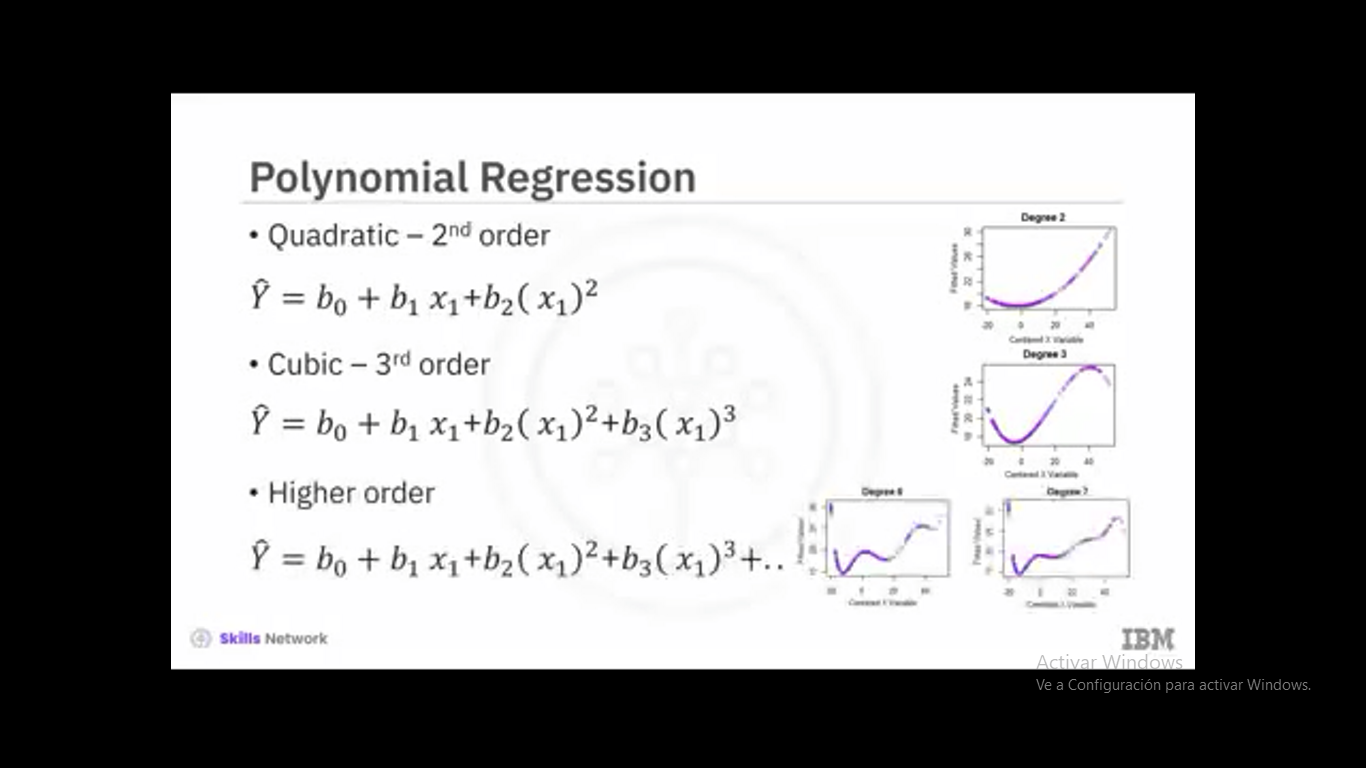


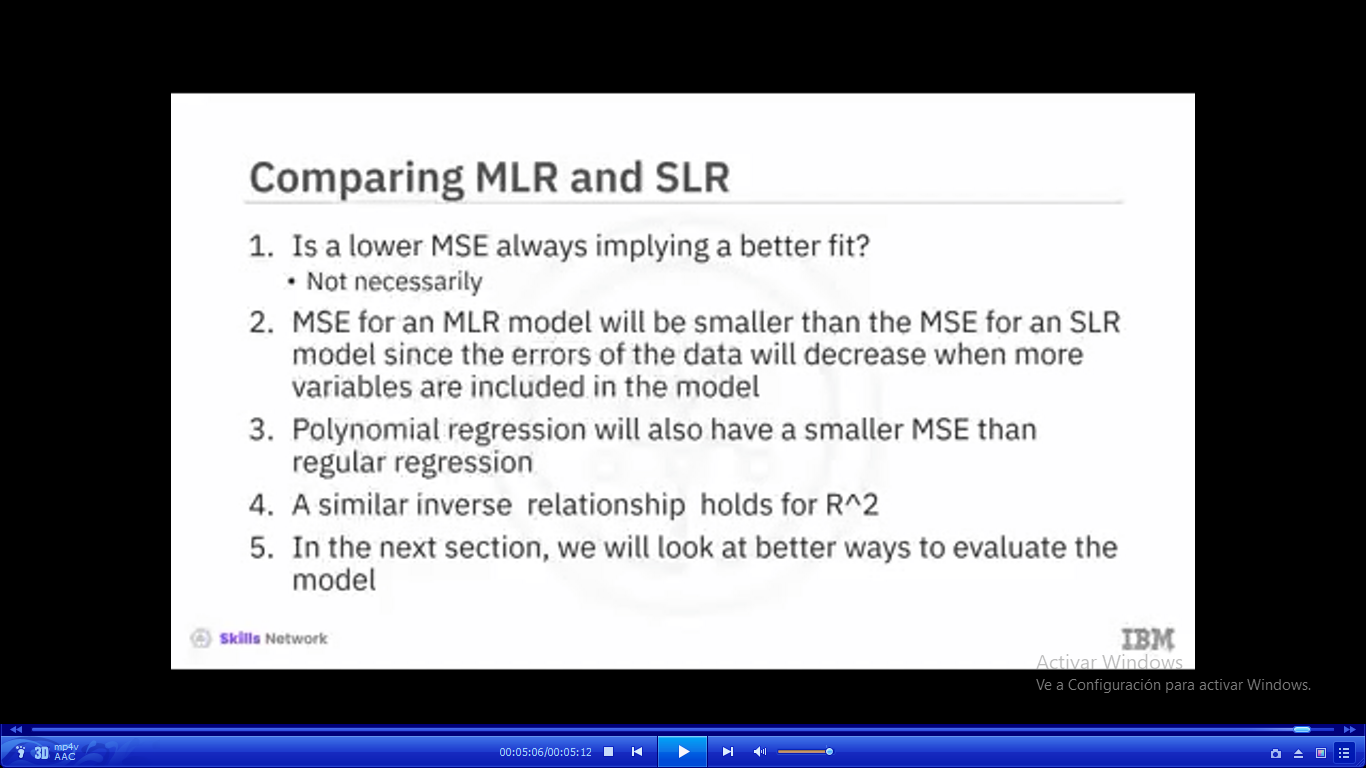






### 1.4.4 Regresión lineal polinómica





## 1.5 Módulo 5

# 2.0 Visualización de datos con Python

## 2.1 Módulo 1

## 2.2 Módulo 2

## 2.3 Módulo 3

## 2.4 Módulo 4

## 2.5 Módulo 5