UNIVERSIDAD DE COLIMA

University of Colima
Faculty of Mechanical and Electrical Engineering
Intelligent Computer Engineering

Graphing Simple Linear Regression.
Data analysis and visualization

Larios Bravo Cristian Armando 20188165

6°D

Place: Mexico, Colima, Coquimatlan.
Date: 24/04/2024.

💎 INTRODUCTION

Machine learning algorithms are programs (math and logic) that adjust themselves to perform better as they are exposed to more data. The "learning" part of machine learning means that those programs change how they process data over time, much as humans change how they process data by learning. So a machine-learning algorithm is a program with a specific way to adjusting its own parameters, given feedback on its previous performance in making predictions about a dataset.

Simple linear regression is a statistical method used to model the relationship between two variables, where one variable (the independent variable) is used to predict the value of another variable (the dependent variable). The relationship between the two variables is assumed to be linear, meaning that the change in the independent variable is proportional to the change in the dependent variable.

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable.

🐜DEVELOPMENT

1. Download and open the jupyter notebook attached in this message.
2. Calculate Simple Linear Regression, from the following for independent and dependent variables of a company that has contracted advertising services for its products, in three media: newspapers, radio and television and the company is determined to measure the impact on the sales of its products TV -Sales Radio - Sales Newspaper - Sales
3. Calculate Multiple Linear Regression: TV, Radio, Newspaper - Sales
4. When you finished this exercices, create a markdown cell and write between 250 and 300 words in Englis, your personal reflection about the simple regression exercise developped in the session . In your reflection try to answer the question. How the development of this activity has helped me to improve my skills in the domain of machine learning algorithms?
5. At the top of the jupyter notebook create a markdown cell where you will add the cover page of your homework. It includes an official cover page with logos, name of the institution, subject and personal data as well as place and date 🆔

22 april 2024 class

Machine Learning

ML stands for "Machine Learning" in English, and refers to a branch of artificial intelligence that focuses on the development of algorithms and techniques that allow computers to learn through experience and data, rather than be explicitly programmed to perform a task.

In other words, machine learning is based on the idea that machines can improve their performance on a task as they receive more data and feedback, and use this information to adjust their models and algorithms.

Machine learning is applied in a wide variety of fields, including image and speech recognition, natural language processing, fraud and anomaly detection, financial risk prediction, and business process optimization.

No description has been provided for this image

Types of Machine Learning

Machine learning can be divided into two main categories: supervised learning and unsupervised learning.

Supervised learning is one in which the learning algorithm is provided with a set of labeled data, that is, data that already has a classification or label assigned. The goal of the algorithm is to learn to classify new data or predict output values based on the input data. For example, a supervised learning algorithm can be trained to classify emails as spam or non-spam, or to predict the price of a house based on its characteristics.

On the other hand, **unsupervised learning** is one in which no labels are provided for the data. The goal of the algorithm is to discover patterns and structures in the data without any specific guidance. For example, an unsupervised learning algorithm can be used to group a store's customers into different categories based on their purchasing behavior, or to find patterns in large unlabeled data sets.

We can say that, while supervised learning focuses on learning to predict output values from labeled data, unsupervised learning focuses on discovering patterns and structures in the data. em> without any specific guide.

Types of ML Algorithms

There are several types of algorithms in machine learning (ML), each with its own specific technique and purpose. Here are some examples:

1. Regression: used to predict continuous values, such as the price of a house or a person's salary.

2. Classification: Used to predict discrete values, such as whether an email is spam or not, or whether an image contains a cat or a dog.
3. Clustering: Used to group similar data into groups or clusters, such as customers with similar purchasing behaviors.
4. Decision Trees: These are used to represent decisions and actions in a tree-like format, where each node represents a decision and each branch represents an action.
5. Neural networks: They are used to imitate the functioning of the human brain and are used for complex classification and prediction tasks.
6. Deep learning: A subcategory of neural networks that uses multiple layers of processing to learn complex features from data.
7. Reinforcement learning: It is used to train models that can make optimal decisions and actions in a specific environment, based on the feedback and rewards received.

These are just a few examples of the many types of ML algorithms that exist, and **choosing the right algorithm will depend on the specific task you want to solve**.

Simple or Multiple Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. In its simplest form, it is a linear function that fits a set of data to predict values of the dependent variable based on the values of the independent variables.

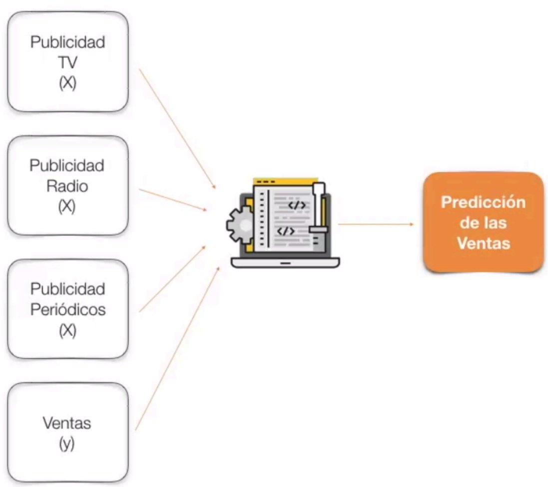
In linear regression, it is assumed that the relationship between the dependent variable and the independent variables is linear, which means that the dependent variable can be expressed as a linear combination of the independent variables.

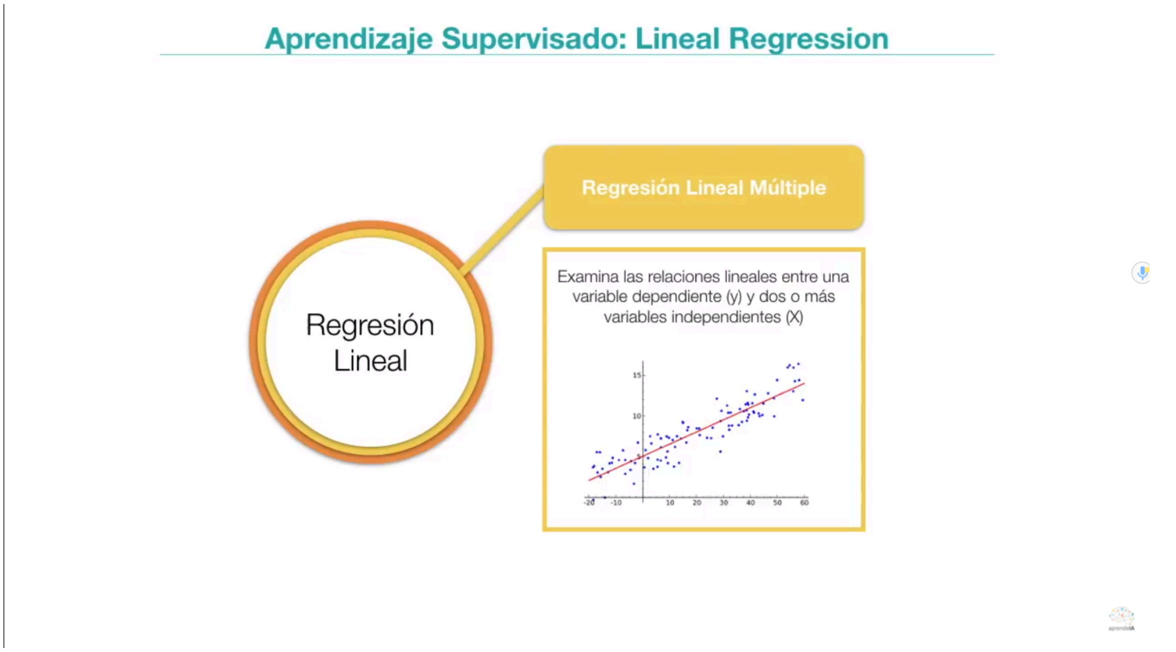
The goal of linear regression is to find the values of the parameters of the linear function that best fit the data. This is achieved by minimizing the sum of the squared errors between the actual values of the dependent variable and the values predicted by the model.

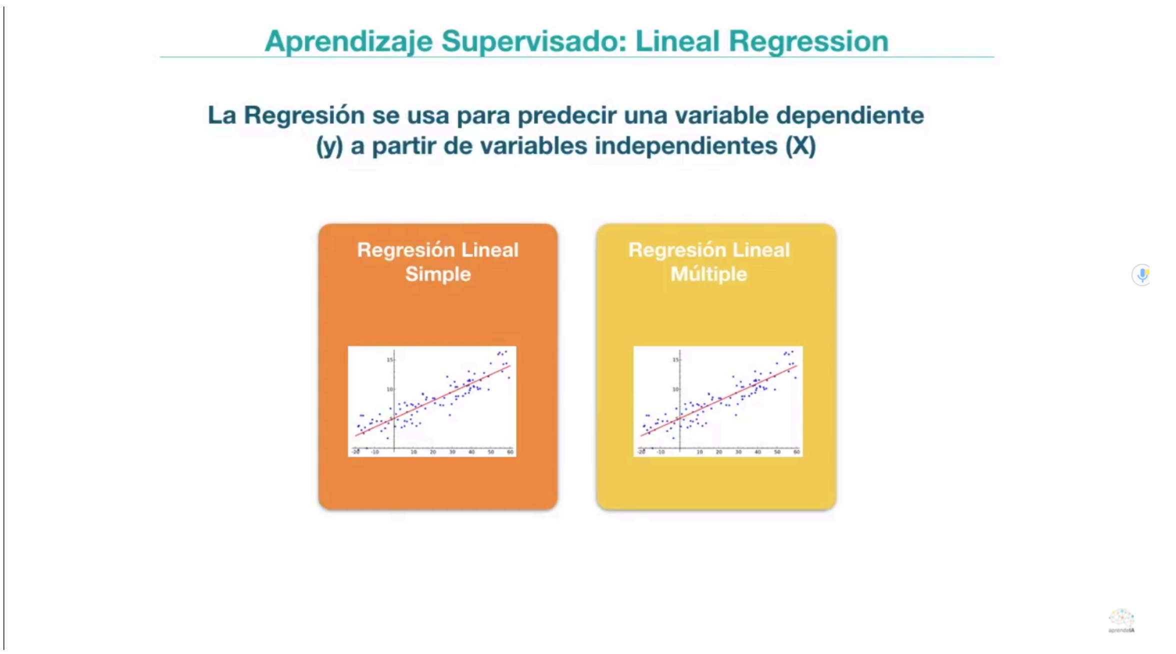
Linear regression is used in a wide variety of fields, including economics, psychology, biology, engineering, and data science. It is a simple but powerful technique that can provide valuable information about the relationship between variables.

Aprendizaje Supervisado: Lineal Regression

Aprendizaje Supervisado: Lineal Regression







Simple Linear Regression with Python

The *statsmodel* package helps to build linear regression

In [ ]:
```
# Importamos librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
# Leemos dataset csv
data = pd.read_csv("Sales_200_days.csv")
```

In [ ]:
```
# Explorar el dataset
```

```
data.head()
```

Out[ ]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [ ]:
```
# Ver la correlación entre las variables
data.corr()
```

Out[ ]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
Newspaper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000

In [ ]:
```
# Importamos statsmodels para obtener la regresión lineal
import statsmodels.formula.api as smf
```

Relationship of sales with the TV variable

In [ ]:
```
# Relacionamos la variable independiente TV con la variable dependiente Ventas (Sal
lm = smf.ols(formula="Sales~TV", data = data).fit()
```

In [ ]:
```
lm.params

# Nota el valor de predicción sería Sales = 7.032594 + 0.047537 * TV
```

Out[ ]:
```
Intercept    7.032594
TV           0.047537
dtype: float64
```

In [ ]:
```
# Determinar el valor del coeficiente de predictibilidad
lm.pvalues
```

Out[ ]:
```
Intercept    1.406300e-35
TV           1.467390e-42
dtype: float64
```

In [ ]:
```
lm.rsquared
```

Out[ ]: `0.611875050850071`

In [ ]:
```
lm.rsquared_adj
```

Out[ ]: `0.6099148238341623`

In [ ]:
```
lm.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.612
Model:	OLS	**Adj. R-squared:**	0.610
Method:	Least Squares	**F-statistic:**	312.1
Date:	Sun, 28 Apr 2024	**Prob (F-statistic):**	1.47e-42
Time:	17:30:14	**Log-Likelihood:**	-519.05
No. Observations:	200	**AIC:**	1042.
Df Residuals:	198	**BIC:**	1049.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	7.0326	0.458	15.360	0.000	6.130	7.935
TV	0.0475	0.003	17.668	0.000	0.042	0.053

Omnibus:	0.531	**Durbin-Watson:**	1.935
Prob(Omnibus):	0.767	**Jarque-Bera (JB):**	0.669
Skew:	-0.089	**Prob(JB):**	0.716
Kurtosis:	2.779	**Cond. No.**	338.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```
# Error de predictibilidad
sales_pred = lm.predict(pd.DataFrame(data["TV"]))
sales_pred
```

Out[ ]:
```
0      17.970775
1       9.147974
2       7.850224
3      14.234395
4      15.627218
         ...
195     8.848493
196    11.510545
197    15.446579
198    20.513985
199    18.065848
Length: 200, dtype: float64
```

In [ ]:
```
# Para agregar una columna con las predicciones de ventas generadas por el modelo c
# Agregamos al dataset una columna de las predicciones
data['SalesTV'] = 7.032594 + 0.047537 * data['TV']
# Agregamos el error residual
data['RSE_SalesTV'] = (data['Sales'] - data['SalesTV'])**2
# Suma de los cuadrados de los errores o de las diferencias
SSD = sum(data['RSE_SalesTV'])
SSD
```

Out[ ]:
```
2102.530583889652
```

In [ ]:
```
data
```

Out[ ]:

	TV	Radio	Newspaper	Sales	SalesTV	RSE_SalesTV
0	230.1	37.8	69.2	22.1	17.970858	17.049816
1	44.5	39.3	45.1	10.4	9.147990	1.567528
2	17.2	45.9	69.3	9.3	7.850230	2.101832
3	151.5	41.3	58.5	18.5	14.234450	18.194921
4	180.8	10.8	58.4	12.9	15.627284	7.438076
...	...	...	...	...	...	...
195	38.2	3.7	13.8	7.6	8.848507	1.558771
196	94.2	4.9	8.1	9.7	11.510579	3.278198
197	177.0	9.3	6.4	12.8	15.446643	7.004719
198	283.6	42.0	66.2	25.5	20.514087	24.859326
199	232.1	8.6	8.7	13.4	18.065932	21.770919

200 rows × 6 columns

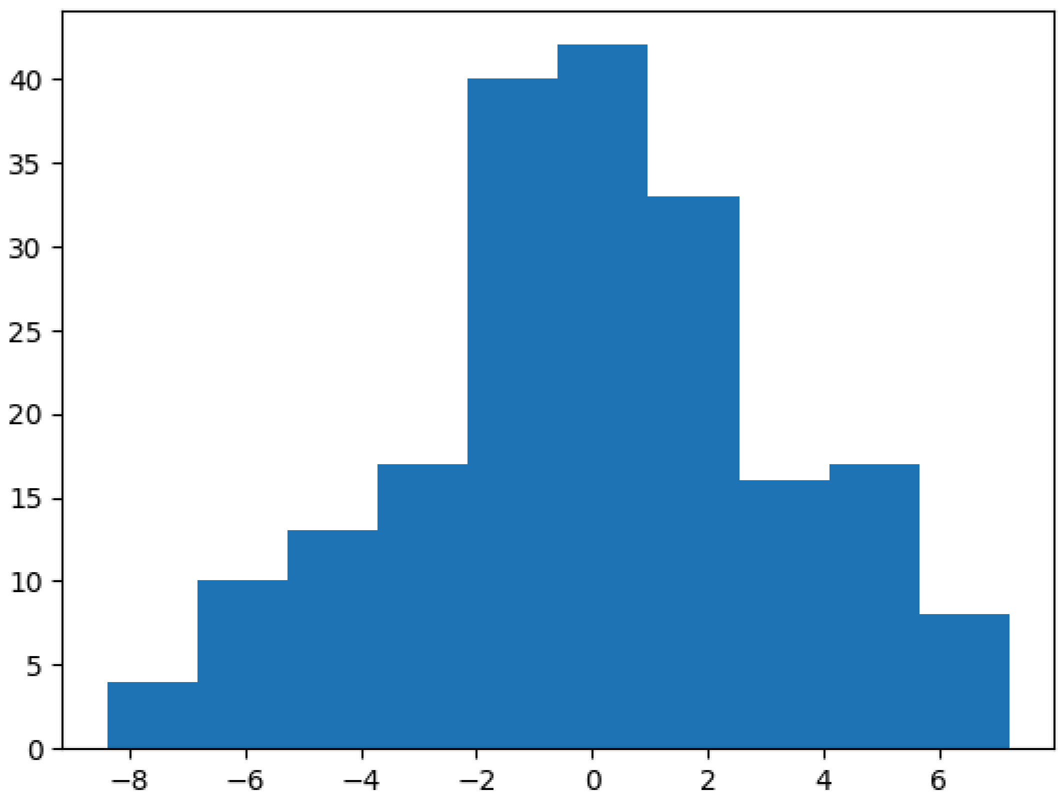
In [ ]:
```
# Error residual
# El error estandar residual
RSE = np.sqrt(SSD/(len(data)-2))
RSE
# Esta es la desviación estandar de los residuos
```

Out[ ]: 3.2586563692380976

In [ ]:
```
# El promedio de ventas
sales_m = np.mean(data['Sales'])
sales_m
```

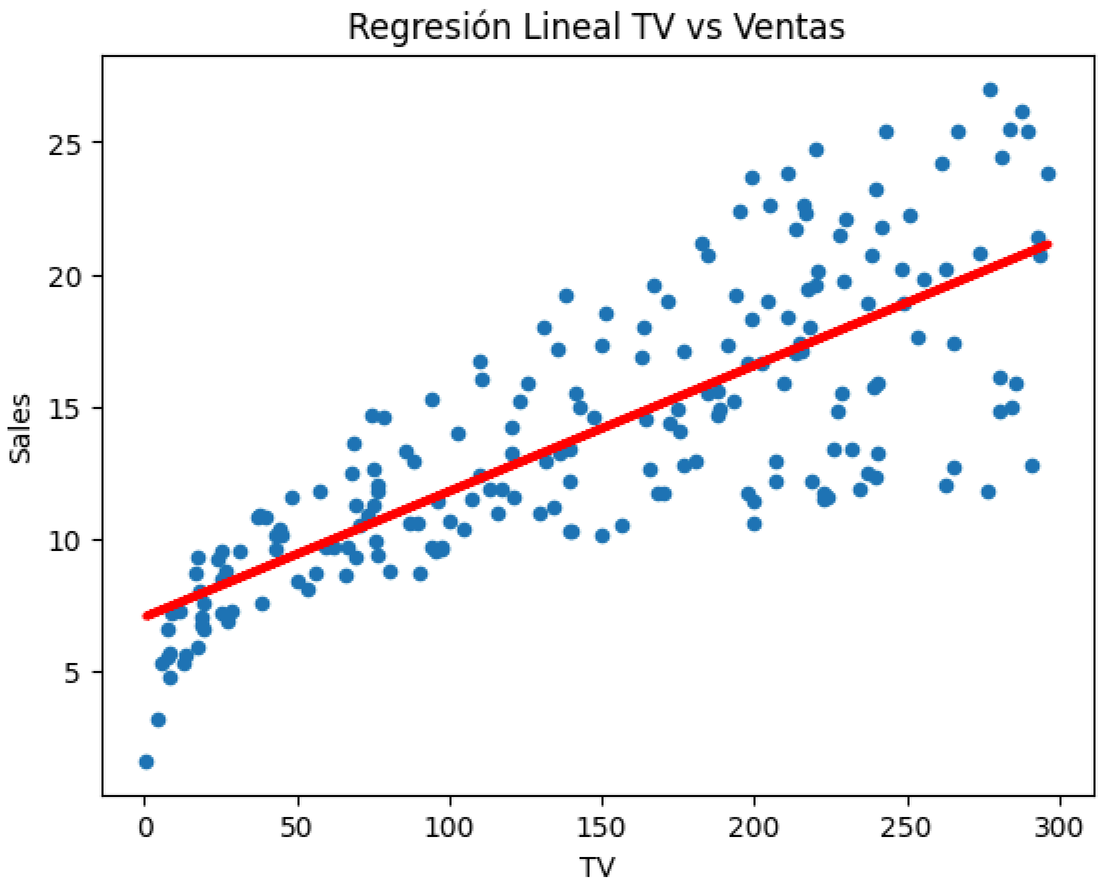
Out[ ]: 14.0225

In [ ]:
```
# Histograma de los errores
plt.hist(data['Sales'] - data['SalesTV'])
plt.show()
```



In [ ]:
```
#Graficar la regresión lineal
import matplotlib.pyplot as plt
```

In [ ]:
```
%matplotlib inline
data.plot(kind = "scatter", x = "TV", y="Sales")
plt.plot(pd.DataFrame(data["TV"]), sales_pred, c = "red", linewidth= 3)
plt.title("Regresión Lineal TV vs Ventas")
plt.show()
```



Relationship of sales with the Radio variable

In [ ]:
```
# Relacionamos la variable independiente TV con la variable dependiente Ventas (Sal
lm = smf.ols(formula="Sales~Radio", data = data).fit()
```

In [ ]:
```
lm.params

# Nota el valor de predicción sería Sales = 7.032594 + 0.047537 * Radio
```

Out[ ]:
```
Intercept    9.311638
Radio        0.202496
dtype: float64
```

In [ ]:
```
# Determinar el valor del coeficiente de predictibilidad
lm.pvalues
```

Out[ ]:
```
Intercept    3.561071e-39
Radio        4.354966e-19
dtype: float64
```

In [ ]:
```
lm.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.332
Model:	OLS	**Adj. R-squared:**	0.329
Method:	Least Squares	**F-statistic:**	98.42
Date:	Sun, 28 Apr 2024	**Prob (F-statistic):**	4.35e-19
Time:	17:30:14	**Log-Likelihood:**	-573.34
No. Observations:	200	**AIC:**	1151.
Df Residuals:	198	**BIC:**	1157.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	9.3116	0.563	16.542	0.000	8.202	10.422
Radio	0.2025	0.020	9.921	0.000	0.162	0.243

Omnibus:	19.358	**Durbin-Watson:**	1.946
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	21.910
Skew:	-0.764	**Prob(JB):**	1.75e-05
Kurtosis:	3.544	**Cond. No.**	51.4

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```python
# Error de predictibilidad
sales_pred = lm.predict(pd.DataFrame(data["Radio"]))
sales_pred
```

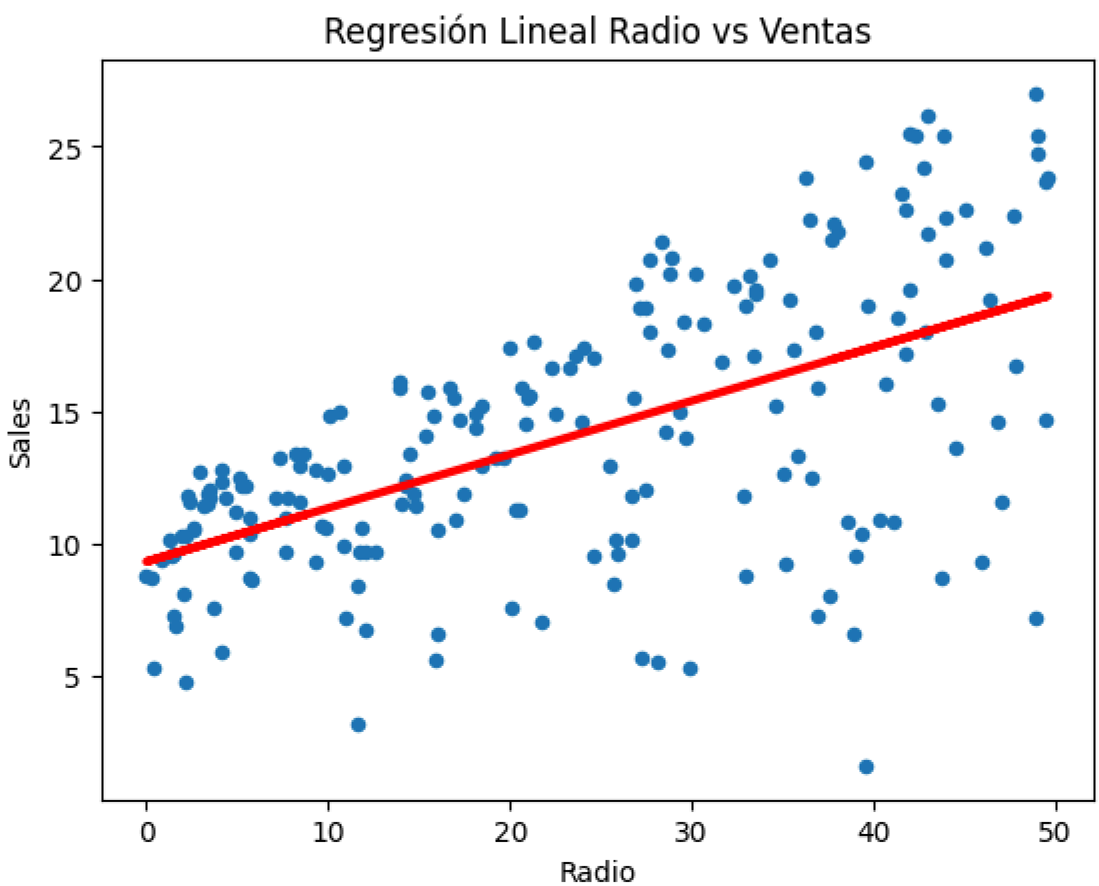
Out[ ]:
```
0      16.965979
1      17.269722
2      18.606195
3      17.674714
4      11.498593
         ...
195    10.060872
196    10.303867
197    11.194849
198    17.816461
199    11.053102
Length: 200, dtype: float64
```

In [ ]:
```
# Para agregar una columna con las predicciones de ventas generadas por el modelo c
# Agregamos al dataset una columna de las predicciones
data['SalesRadio'] = 9.3116 + 0.563 * data['Radio']
# Agregamos el error residual
data['RSE_SalesRadio'] = (data['Sales'] - data['SalesRadio'])**2
SSD = sum(data['RSE_SalesRadio'])
SSD
```

Out[ ]:
```
23386.789700819994
```

In [ ]:
```
#Graficar la regresión lineal
import matplotlib.pyplot as plt
```

In [ ]:
```
%matplotlib inline
data.plot(kind = "scatter", x = "Radio", y="Sales")
plt.plot(pd.DataFrame(data["Radio"]), sales_pred, c = "red", linewidth= 3)
plt.title("Regresión Lineal Radio vs Ventas")
plt.show()
```



Relationship of sales with the Newspaper variable

In [ ]:
```
# Relacionamos la variable independiente Newspaper con la variable dependiente Vent
lm = smf.ols(formula="Sales~Newspaper", data = data).fit()
```

In [ ]:
```
lm.params
```

```
# Nota el valor de predicción sería Sales = 7.032594 + 0.047537 * TV
```

Out[ ]:
```
Intercept    12.351407
Newspaper     0.054693
dtype: float64
```

In [ ]:
```
# Determinar el valor del coeficiente de predictibilidad
lm.pvalues
```

Out[ ]:
```
Intercept    4.713507e-49
Newspaper    1.148196e-03
dtype: float64
```

In [ ]:
```
lm.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.052
Model:	OLS	**Adj. R-squared:**	0.047
Method:	Least Squares	**F-statistic:**	10.89
Date:	Sun, 28 Apr 2024	**Prob (F-statistic):**	0.00115
Time:	17:30:15	**Log-Likelihood:**	-608.34
No. Observations:	200	**AIC:**	1221.
Df Residuals:	198	**BIC:**	1227.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	12.3514	0.621	19.876	0.000	11.126	13.577
Newspaper	0.0547	0.017	3.300	0.001	0.022	0.087

Omnibus:	6.231	**Durbin-Watson:**	1.983
Prob(Omnibus):	0.044	**Jarque-Bera (JB):**	5.483
Skew:	0.330	**Prob(JB):**	0.0645
Kurtosis:	2.527	**Cond. No.**	64.7

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```
# Error de predictibilidad
sales_pred = lm.predict(pd.DataFrame(data["Newspaper"]))
sales_pred
```

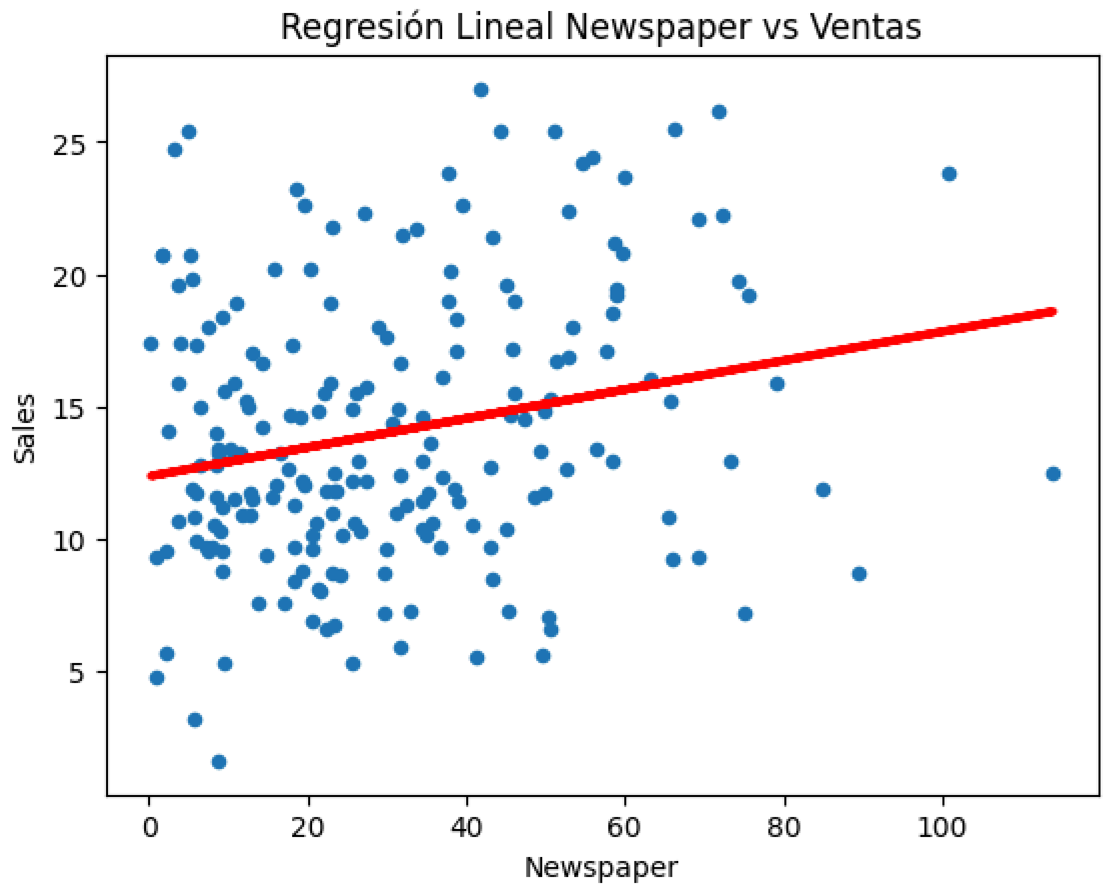
Out[ ]:
```
0      16.136169
1      14.818066
2      16.141639
3      15.550953
4      15.545484
         ...
195    13.106172
196    12.794421
197    12.701443
198    15.972090
199    12.827237
Length: 200, dtype: float64
```

In [ ]:
```
# Para agregar una columna con las predicciones de ventas generadas por el modelo c
# Agregamos al dataset una columna de las predicciones
data['SalesNewspaper'] = 12.3514 + 0.621 * data['Newspaper']
# Agregamos el error residual
data['RSE_SalesNewspaper'] = (data['Sales'] - data['SalesNewspaper'])**2
SSD = sum(data['RSE_SalesNewspaper'])
SSD
```

Out[ ]:
```
95283.46586138
```

In [ ]:
```
#Graficar la regresión lineal
import matplotlib.pyplot as plt
```

In [ ]:
```
%matplotlib inline
data.plot(kind = "scatter", x = "Newspaper", y="Sales")
plt.plot(pd.DataFrame(data["Newspaper"]), sales_pred, c = "red", linewidth= 3)
plt.title("Regresión Lineal Newspaper vs Ventas")
plt.show()
```



Activities

From here create cells and add your results

Simple linnear Regression

In [ ]:
```
import tkinter as tk
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Crear una ventana
root = tk.Tk()
root.geometry("600x400")
root.title("Inversión en publicidad")

# Solicitar el monto de inversión en publicidad
msg_inversion = tk.Label(root, text="¿Cuánto se invertirá en publicidad?")
msg_inversion.pack()

entry_inversion = tk.Entry(root)
entry_inversion.pack()

# Solicitar en qué se va invertir
msg_opcion = tk.Label(root, text="¿En qué se invertirá?")
msg_opcion.pack()
```

```
# Crear una lista de opciones
options = ["TV", "Radio", "Newspaper"]
variable = tk.StringVar(root)
variable.set(options[0])

# Crear un menú desplegable
dropdown = tk.OptionMenu(root, variable, *options)
dropdown.pack()

# Crear una función para obtener la inversión
def get_inversion():
    inversion = float(entry_inversion.get())
    opcion = variable.get()
    print("La inversión en publicidad es de:", inversion)
    print("La inversión será en:", opcion)

    # Ajustar un modelo de regresión lineal
    X = sm.add_constant(data[opcion])
    y = data['Sales']
    model = sm.OLS(y, X).fit()
    print(model.summary())

    # Predecir las ventas
    sales_pred = model.predict(X)
    prediction = model.predict([1, inversion])

    # Calcular las ganancias
    # ganancia = prediction - inversion

    print(f"Se ganó {prediction} de la inversión.")

    # Visualizar la regresión lineal
    plt.scatter(data[opcion], data['Sales'])
    plt.plot(data[opcion], sales_pred, color='red')
    plt.scatter(inversion, prediction, color='green')
    plt.xlabel(opcion)
    plt.ylabel('Sales')
    plt.title(f'Simple linnear Regression {opcion} vs Ventas')
    plt.show()

button = tk.Button(root, text="Enviar", command=get_inversion)
button.pack()

# Mostrar la ventana
root.mainloop()
```

How the development of this activity has helped me to improve my skills in the domain of machine learning algorithms?

I learned how to do linear regressions in python, which helps me predict values in the future. Engaging in the realm of machine learning algorithms has been a transformative journey, enriching my skill set in multifaceted ways. Through the continuous development of this activity, I've witnessed a profound evolution in my proficiency and understanding of machine learning concepts. One pivotal aspect of this journey has been the enhancement of my problem-solving skills. Confronted with diverse datasets and complex challenges, I've learned to dissect problems methodically, devising innovative solutions through the application of various algorithms. This iterative process has honed my analytical abilities, allowing me to navigate intricate problems with precision and efficiency. Furthermore, the development of this activity has fostered a deep comprehension of algorithmic principles. Experimentation with different models, ranging from classical techniques to cutting-edge neural networks, has broadened my understanding of algorithmic paradigms and their applications across diverse domains. As I delve deeper into the intricacies of machine learning, I've cultivated a nuanced appreciation for the strengths and limitations of different algorithms, empowering me to make informed decisions in model selection and optimization. Moreover, engaging in this activity has nurtured my creativity and adaptability. Exploring novel approaches and refining existing methodologies have instilled in me a sense of resilience and resourcefulness, crucial attributes in the dynamic landscape of machine learning. In essence, the development of this activity has served as a crucible for my growth as a machine learning practitioner. It has not only expanded my technical proficiency but also instilled in me a profound passion for exploring the boundless possibilities of artificial intelligence. As I continue on this journey, I am excited by the prospect of further honing my skills and contributing to the advancement of this transformative field.

TV + Newspaper

In [ ]:

```
# Regresion múltiple

# Añadir el Newspaper al modelo existente
lm2 = smf.ols(formula="Sales~TV+Newspaper", data=data).fit()
# Revisemos los parámetros
lm2.params
```

Out[ ]:

```
Intercept    5.774948
TV           0.046901
Newspaper    0.044219
dtype: float64
```

In [ ]:

```
lm2.pvalues
```

Out[ ]:

```
Intercept    3.145860e-22
TV           5.507584e-44
Newspaper    2.217084e-05
dtype: float64
```

In [ ]:

```
lm2.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.646
Model:	OLS	**Adj. R-squared:**	0.642
Method:	Least Squares	**F-statistic:**	179.6
Date:	Sun, 28 Apr 2024	**Prob (F-statistic):**	3.95e-45
Time:	17:30:15	**Log-Likelihood:**	-509.89
No. Observations:	200	**AIC:**	1026.
Df Residuals:	197	**BIC:**	1036.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	5.7749	0.525	10.993	0.000	4.739	6.811
TV	0.0469	0.003	18.173	0.000	0.042	0.052
Newspaper	0.0442	0.010	4.346	0.000	0.024	0.064

Omnibus:	0.658	**Durbin-Watson:**	1.969
Prob(Omnibus):	0.720	**Jarque-Bera (JB):**	0.415
Skew:	-0.093	**Prob(JB):**	0.813
Kurtosis:	3.122	**Cond. No.**	410.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```
# La ecuación del modelo
#Sales = 5.774948 + 0.046901TV+ 0.044219Newspaper
lm2.rsquared
```

Out[ ]: 0.6458354938293271

In [ ]:
```
lm2.rsquared_adj
```

Out[ ]: 0.6422399150864777

In [ ]:
```
# Hacemos algunas predicciones
sales_pred = lm2.predict(data[['TV', 'Newspaper']])
```

In [ ]:
```
data.head()
```

Out[ ]:

	TV	Radio	Newspaper	Sales	SalesTV	RSE_SalesTV	SalesRadio	RSE_SalesRadio	S
0	230.1	37.8	69.2	22.1	17.970858	17.049816	30.5930	72.131049	
1	44.5	39.3	45.1	10.4	9.147990	1.567528	31.4375	442.576406	
2	17.2	45.9	69.3	9.3	7.850230	2.101832	35.1533	668.393121	
3	151.5	41.3	58.5	18.5	14.234450	18.194921	32.5635	197.782032	
4	180.8	10.8	58.4	12.9	15.627284	7.438076	15.3920	6.210064	

In [ ]:

```
# Desviación estandar de los residuos
# Para agregar una columna con las predicciones de ventas generadas por el modelo c
# Agregamos al dataset una columna de las predicciones
data['SalesTVNewspaper'] = 5.774948 + 0.046901 * data['TV'] + 0.044219 * data['News
# Agregamos el error residual
RSE_SalesTVNewspaper = (data['Sales'] - data['SalesTVNewspaper'])**2
# Suma de los cuadrados de los errores o de las diferencias
SSD = sum(RSE_SalesTVNewspaper)
SSD
# SSD = sum((data['Sales'] -sales_pred)**2)
# SSD
```

Out[ ]: 1918.5618123786915

In [ ]:

```
data
```

Out[ ]:

	TV	Radio	Newspaper	Sales	SalesTV	RSE_SalesTV	SalesRadio	RSE_SalesRadio
0	230.1	37.8	69.2	22.1	17.970858	17.049816	30.5930	72.131049
1	44.5	39.3	45.1	10.4	9.147990	1.567528	31.4375	442.576406
2	17.2	45.9	69.3	9.3	7.850230	2.101832	35.1533	668.393121
3	151.5	41.3	58.5	18.5	14.234450	18.194921	32.5635	197.782032
4	180.8	10.8	58.4	12.9	15.627284	7.438076	15.3920	6.210064
...	...	...	...	...	...	...	...	...
195	38.2	3.7	13.8	7.6	8.848507	1.558771	11.3947	14.399748
196	94.2	4.9	8.1	9.7	11.510579	3.278198	12.0703	5.618322
197	177.0	9.3	6.4	12.8	15.446643	7.004719	14.5475	3.053756
198	283.6	42.0	66.2	25.5	20.514087	24.859326	32.9576	55.615798
199	232.1	8.6	8.7	13.4	18.065932	21.770919	14.1534	0.567612

200 rows × 11 columns

In [ ]:
```
# Error residual
RSE = np.sqrt(SSD/(len(data)-3))
RSE
```

Out[ ]: 3.1207198606447837

In [ ]:
```
# Comprobar con respecto al promedio
# Calcular el error promedio del modelo
error = RSE / sales_m
error
```

Out[ ]: 0.222550890400769

In [ ]:
```
lm2.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.646
Model:	OLS	**Adj. R-squared:**	0.642
Method:	Least Squares	**F-statistic:**	179.6
Date:	Sun, 28 Apr 2024	**Prob (F-statistic):**	3.95e-45
Time:	17:30:15	**Log-Likelihood:**	-509.89
No. Observations:	200	**AIC:**	1026.
Df Residuals:	197	**BIC:**	1036.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	5.7749	0.525	10.993	0.000	4.739	6.811
TV	0.0469	0.003	18.173	0.000	0.042	0.052
Newspaper	0.0442	0.010	4.346	0.000	0.024	0.064

Omnibus:	0.658	**Durbin-Watson:**	1.969
Prob(Omnibus):	0.720	**Jarque-Bera (JB):**	0.415
Skew:	-0.093	**Prob(JB):**	0.813
Kurtosis:	3.122	**Cond. No.**	410.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

TV + Radio

In [ ]:
```
# Añadir el Radio al modelo existente
lm3 = smf.ols(formula="Sales~TV+Radio", data=data).fit()
```

In [ ]:
```
# Revisamos los parámetros
lm3.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.897
Model:	OLS	**Adj. R-squared:**	0.896
Method:	Least Squares	**F-statistic:**	859.6
Date:	Sun, 28 Apr 2024	**Prob (F-statistic):**	4.83e-98
Time:	17:30:15	**Log-Likelihood:**	-386.20
No. Observations:	200	**AIC:**	778.4
Df Residuals:	197	**BIC:**	788.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.9211	0.294	9.919	0.000	2.340	3.502
TV	0.0458	0.001	32.909	0.000	0.043	0.048
Radio	0.1880	0.008	23.382	0.000	0.172	0.204

Omnibus:	60.022	**Durbin-Watson:**	2.081
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	148.679
Skew:	-1.323	**Prob(JB):**	5.19e-33
Kurtosis:	6.292	**Cond. No.**	425.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```
lm3.params
```

Out[ ]:
```
Intercept    2.921100
TV           0.045755
Radio        0.187994
dtype: float64
```

In [ ]:
```
lm3.rsquared
```

Out[ ]: 0.8971942610828956

In [ ]:
```
lm3.rsquared_adj
```

Out[ ]: 0.8961505479974428

In [ ]:
```
# Hacemos algunas predicciones
sales_pred = lm3.predict(data[['TV', 'Radio']])
sales_pred
```

Out[ ]:
```
0      20.555465
1      12.345362
2      12.337018
3      17.617116
4      13.223908
         ...
195     5.364512
196     8.152375
197    12.768048
198    23.792923
199    15.157543
Length: 200, dtype: float64
```

In [ ]:
```
data.head()
```

Out[ ]:

	TV	Radio	Newspaper	Sales	SalesTV	RSE_SalesTV	SalesRadio	RSE_SalesRadio	S
0	230.1	37.8	69.2	22.1	17.970858	17.049816	30.5930	72.131049	
1	44.5	39.3	45.1	10.4	9.147990	1.567528	31.4375	442.576406	
2	17.2	45.9	69.3	9.3	7.850230	2.101832	35.1533	668.393121	
3	151.5	41.3	58.5	18.5	14.234450	18.194921	32.5635	197.782032	
4	180.8	10.8	58.4	12.9	15.627284	7.438076	15.3920	6.210064	

In [ ]:
```
# Desviación estandar de los residuos
# Para agregar una columna con las predicciones de ventas generadas por el modelo c
# Agregamos al dataset una columna de las predicciones
data['SalesTVRadio'] = 2.921100 + 0.045755 * data['TV'] + 0.187994 * data['Radio']
# Agregamos el error residual
RSE_SalesTVNewspaper = (data['Sales'] - data['SalesTVRadio'])**2
# Suma de los cuadrados de los errores o de las diferencias
SSD = sum(RSE_SalesTVNewspaper)
SSD

# SSD = sum((data['Sales'] -sales_pred)**2)
# SSD
```

Out[ ]: 556.9139802156839

In [ ]:
```
data
```

Out[ ]:

	TV	Radio	Newspaper	Sales	SalesTV	RSE_SalesTV	SalesRadio	RSE_SalesRadio
0	230.1	37.8	69.2	22.1	17.970858	17.049816	30.5930	72.131049
1	44.5	39.3	45.1	10.4	9.147990	1.567528	31.4375	442.576406
2	17.2	45.9	69.3	9.3	7.850230	2.101832	35.1533	668.393121
3	151.5	41.3	58.5	18.5	14.234450	18.194921	32.5635	197.782032
4	180.8	10.8	58.4	12.9	15.627284	7.438076	15.3920	6.210064
...	...	...	...	...	...	...	...	...
195	38.2	3.7	13.8	7.6	8.848507	1.558771	11.3947	14.399748
196	94.2	4.9	8.1	9.7	11.510579	3.278198	12.0703	5.618322
197	177.0	9.3	6.4	12.8	15.446643	7.004719	14.5475	3.053756
198	283.6	42.0	66.2	25.5	20.514087	24.859326	32.9576	55.615798
199	232.1	8.6	8.7	13.4	18.065932	21.770919	14.1534	0.567612

200 rows × 12 columns

In [ ]:
```
# Error residual
RSE = np.sqrt(SSD/(len(data)-2-1))
RSE
```

Out[ ]: 1.681360912731511

In [ ]:
```
# Error. TV+Radio es muy bueno
RSE/sales_m
```

Out[ ]: 0.11990450438448999

Radio + TV + Newspaper

In [ ]:
```
# Añadir el Newspaper al modelo existente
lm4 = smf.ols(formula="Sales~TV+Radio+Newspaper", data=data).fit()
```

In [ ]:
```
lm4.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.897
Model:	OLS	**Adj. R-squared:**	0.896
Method:	Least Squares	**F-statistic:**	570.3
Date:	Sun, 28 Apr 2024	**Prob (F-statistic):**	1.58e-96
Time:	17:30:16	**Log-Likelihood:**	-386.18
No. Observations:	200	**AIC:**	780.4
Df Residuals:	196	**BIC:**	793.6
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	2.9389	0.312	9.422	0.000	2.324	3.554
TV	0.0458	0.001	32.809	0.000	0.043	0.049
Radio	0.1885	0.009	21.893	0.000	0.172	0.206
Newspaper	-0.0010	0.006	-0.177	0.860	-0.013	0.011

Omnibus:	60.414	**Durbin-Watson:**	2.084
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	151.241
Skew:	-1.327	**Prob(JB):**	1.44e-33
Kurtosis:	6.332	**Cond. No.**	454.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
# Hacemos algunas predicciones
sales_pred = lm4.predict(data[['TV', 'Radio', 'Newspaper']])

# Crear un nuevo dato con los valores de las características
new_data_point = pd.DataFrame({'TV': [200], 'Radio': [100], 'Newspaper': [50]})

# Hacer la predicción para el nuevo dato
prediction = lm4.predict(new_data_point)

# Para agregar una columna con las predicciones de ventas generadas por el modelo c
# Agregamos al dataset una columna de las predicciones
data['SalesTVRadioNewspaper'] = 2.938889 + 0.045765 * data['TV'] + 0.188530 * data[
# Agregamos el error residual
RSE_SalesTVNewspaper = (data['Sales'] - data['SalesTVRadio'])**2
```

```
# Suma de los cuadrados de los errores o de las diferencias
SSD = sum(RSE_SalesTVNewspaper)
SSD
# SSD = sum((data['Sales'] -sales_pred)**2)
# RSE = np.sqrt(SSD/(len(data)-3-1))
```

Out[ ]:

```
556.9139802156839
```

In [ ]:

```
prediction.values[0]
```

Out[ ]:

```
30.892945500235573
```

In [ ]:

```
sales_pred
```

Out[ ]:

```
0      20.523974
1      12.337855
2      12.307671
3      17.597830
4      13.188672
         ...
195     5.370342
196     8.165312
197    12.785921
198    23.767321
199    15.173196
Length: 200, dtype: float64
```

In [ ]:

```
SSD
```

Out[ ]:

```
556.9139802156839
```

In [ ]:

```
RSE
```

Out[ ]:

```
1.681360912731511
```

In [ ]:

```
RSE / sales_m
```

Out[ ]:

```
0.11990450438448999
```

In [ ]:

```
lm4.params
```

Out[ ]:

```
Intercept    2.938889
TV           0.045765
Radio        0.188530
Newspaper   -0.001037
dtype: float64
```

In [ ]:

```
lm4.pvalues
```

Out[ ]:

```
Intercept    1.267295e-17
TV           1.509960e-81
Radio        1.505339e-54
Newspaper    8.599151e-01
dtype: float64
```

In [ ]:
```
lm4.rsquared
```

Out[ ]: 0.8972106381789522

In [ ]:
```
lm4.rsquared_adj
```

Out[ ]: 0.8956373316204668

In [ ]:
```
data
```

Out[ ]:

	TV	Radio	Newspaper	Sales	SalesTV	RSE_SalesTV	SalesRadio	RSE_SalesRadio
0	230.1	37.8	69.2	22.1	17.970858	17.049816	30.5930	72.131049
1	44.5	39.3	45.1	10.4	9.147990	1.567528	31.4375	442.576406
2	17.2	45.9	69.3	9.3	7.850230	2.101832	35.1533	668.393121
3	151.5	41.3	58.5	18.5	14.234450	18.194921	32.5635	197.782032
4	180.8	10.8	58.4	12.9	15.627284	7.438076	15.3920	6.210064
...	...	...	...	...	...	...	...	...
195	38.2	3.7	13.8	7.6	8.848507	1.558771	11.3947	14.399748
196	94.2	4.9	8.1	9.7	11.510579	3.278198	12.0703	5.618322
197	177.0	9.3	6.4	12.8	15.446643	7.004719	14.5475	3.053756
198	283.6	42.0	66.2	25.5	20.514087	24.859326	32.9576	55.615798
199	232.1	8.6	8.7	13.4	18.065932	21.770919	14.1534	0.567612

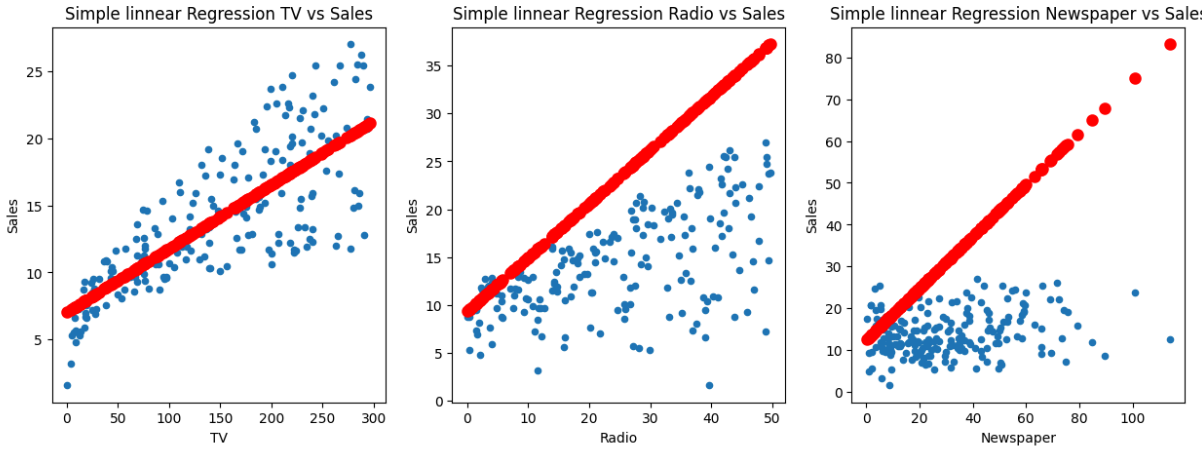
200 rows × 13 columns

Challenge

- Add a column to the dataset for each linear regression model.
- Select and generate a type of graph that clearly illustrates the differences between sales and the generated predictive models.

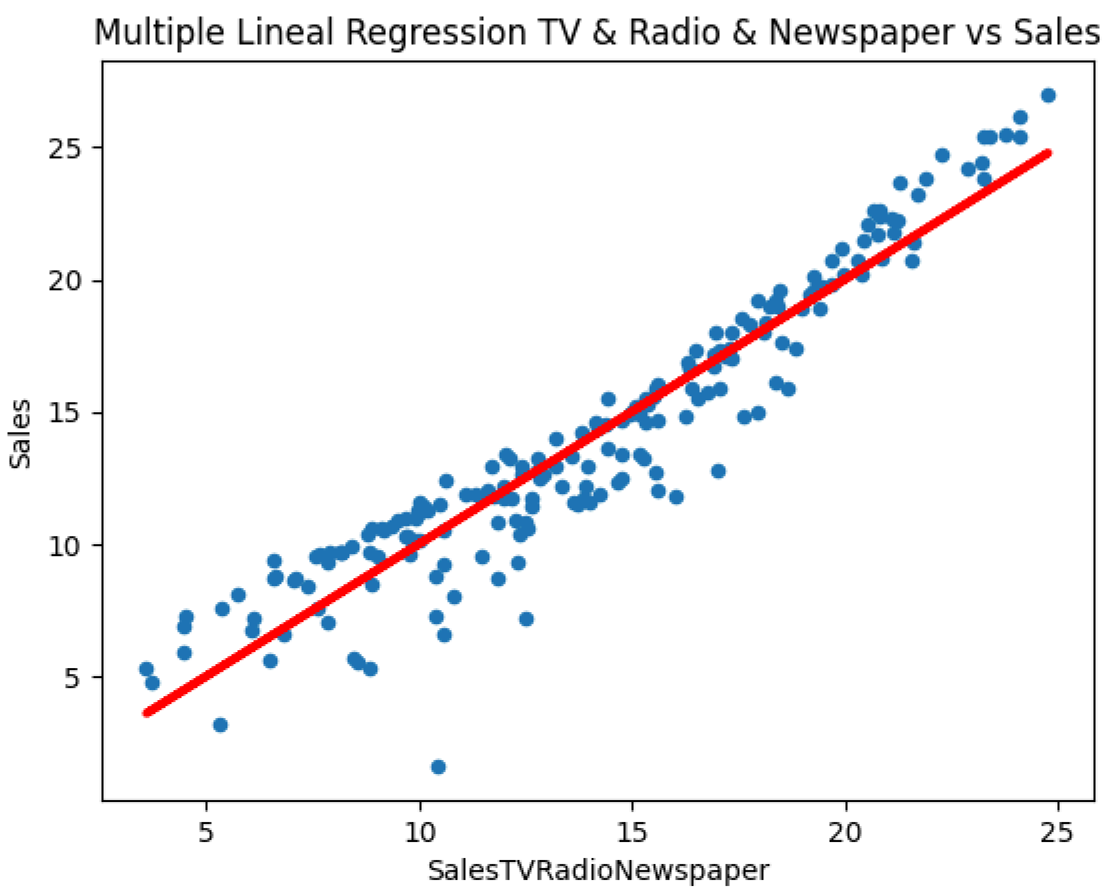
In [ ]:
```
# - Seleccionar y generar un tipo de gráfica que ilustre de manera clara las difere
# Modelos predictivos de una variable
fig, ax = plt.subplots(1, 3, figsize=(15, 5))
# TV
data.plot(kind="scatter", x="TV", y="Sales", ax=ax[0])
ax[0].scatter(data['TV'], data['SalesTV'], c="red", linewidth=3)
ax[0].set_title("Simple linnear Regression TV vs Sales")
# Radio
data.plot(kind="scatter", x="Radio", y="Sales", ax=ax[1])
ax[1].scatter(data['Radio'], data['SalesRadio'], c="red", linewidth=3)
ax[1].set_title("Simple linnear Regression Radio vs Sales")
```

```python
# Newspaper
data.plot(kind="scatter", x="Newspaper", y="Sales", ax=ax[2])
ax[2].scatter(data['Newspaper'], data['SalesNewspaper'], c="red", linewidth=3)
ax[2].set_title("Simple linnear Regression Newspaper vs Sales")
plt.show()
```



In [ ]:

```python
# - Seleccionar y generar un tipo de gráfica que ilustre de manera clara las difere
# Modelos predictivos multiples variables (TV , Newspaper y Radio)
%matplotlib inline
data.plot(kind = "scatter", x = "SalesTVRadioNewspaper", y="Sales")
plt.plot(pd.DataFrame(data["SalesTVRadioNewspaper"]), sales_pred, c = "red", linewi
plt.title("Multiple Lineal Regression TV & Radio & Newspaper vs Sales")
plt.show()
```

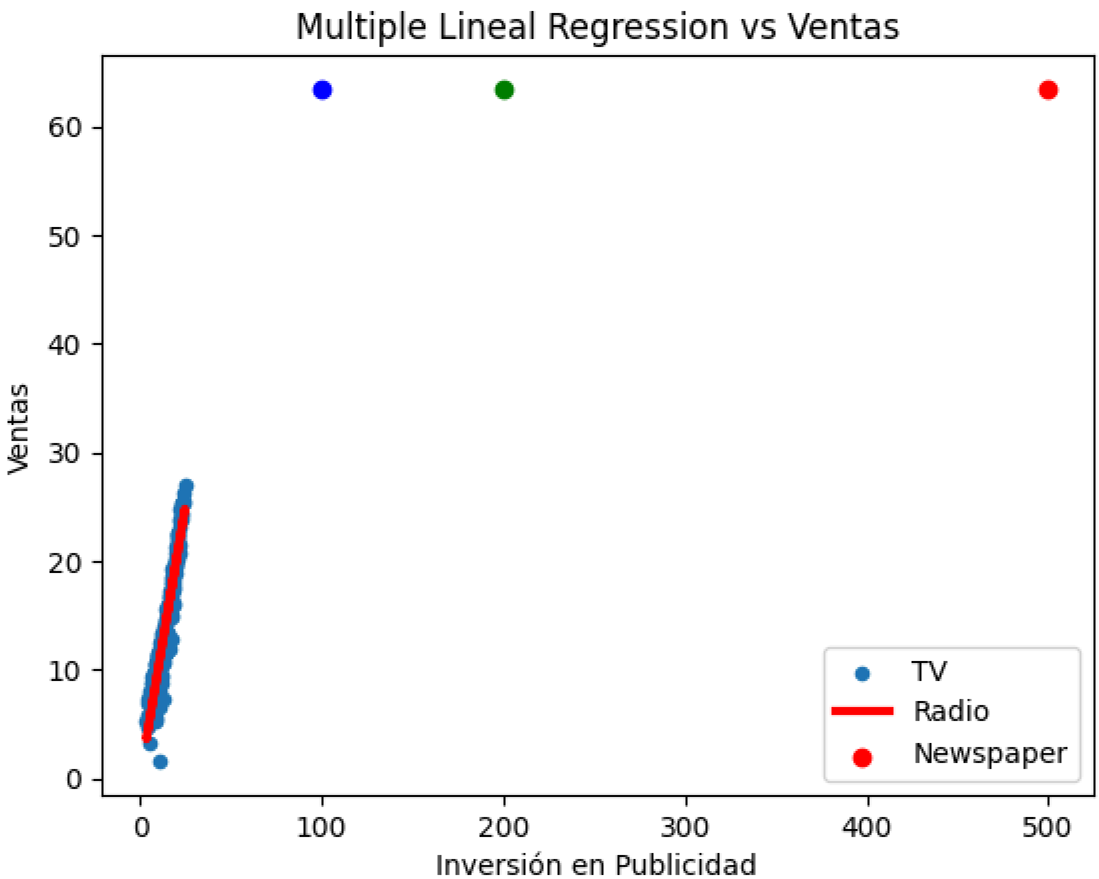


Multiple Lineal Regression

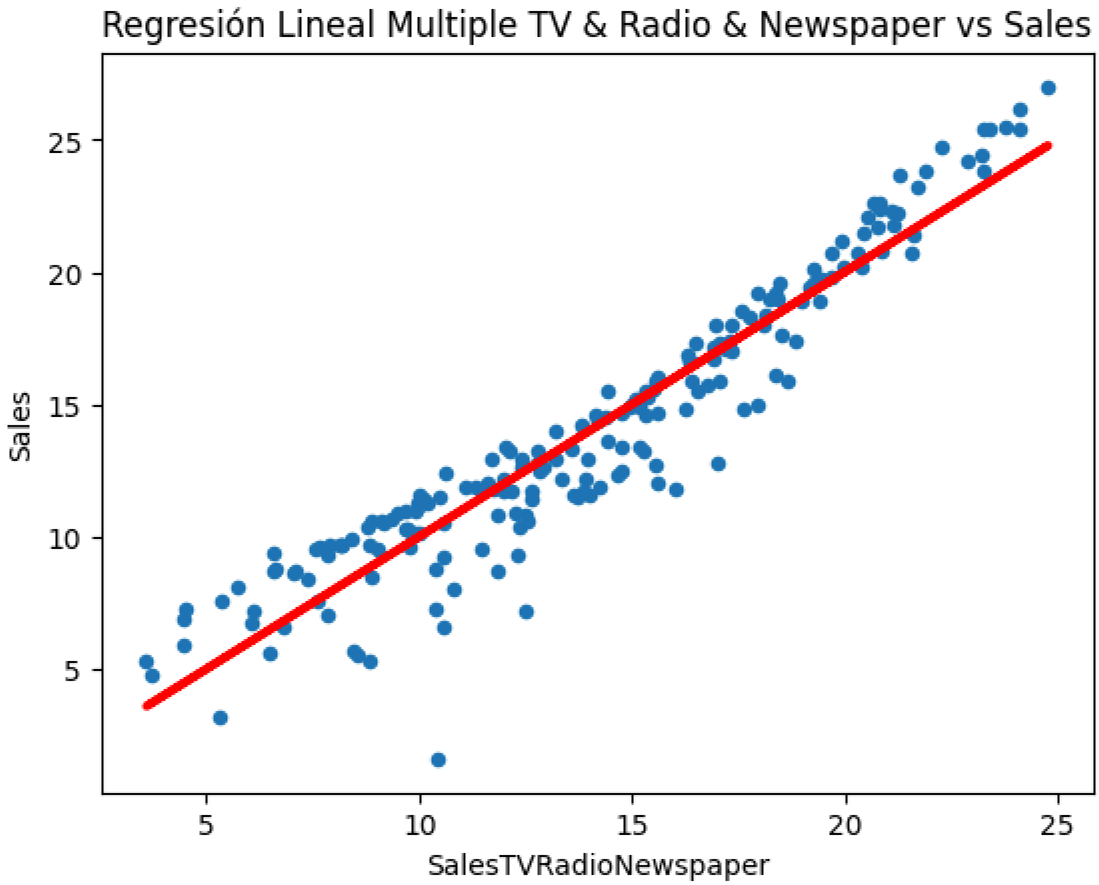
In [ ]:
```
import tkinter as tk
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Crear una ventana
root = tk.Tk()
root.geometry("600x400")
root.title("Inversión en publicidad")

# Solicitar el monto de inversión en publicidad en TV
msg_inversionTV = tk.Label(root, text="¿Cuánto se invertirá en publicidad en TV?")
msg_inversionTV.pack()
# Crear una entrada para el monto de inversión en TV
entry_inversionTV = tk.Entry(root)
entry_inversionTV.pack()

# Solicitar el monto de inversión en publicidad en Radio
msg_inversionRadio = tk.Label(root, text="¿Cuánto se invertirá en publicidad en Rad
msg_inversionRadio.pack()
# Crear una entrada para el monto de inversión en Radio
entry_inversionRadio = tk.Entry(root)
entry_inversionRadio.pack()

# Solicitar el monto de inversión en publicidad en Newspaper
msg_inversionNewspaper = tk.Label(root, text="¿Cuánto se invertirá en publicidad en
msg_inversionNewspaper.pack()
# Crear una entrada para el monto de inversión en Newspaper
entry_inversionNewspaper = tk.Entry(root)
entry_inversionNewspaper.pack()

# Crear una lista de opciones
options = ["TV", "Radio", "Newspaper"]

# Crear una función para obtener la inversión
# Crear una función para obtener la inversión
def get_inversion():
    inversionTV = float(entry_inversionTV.get())
    inversionRadio = float(entry_inversionRadio.get())
    inversionNewspaper = float(entry_inversionNewspaper.get())
    inversion = inversionTV + inversionRadio + inversionNewspaper
    print("La inversión en publicidad es de:", inversion)

    # Ajustar un modelo de regresión lineal múltiple
    # X = sm.add_constant(data[["TV", "Radio", "Newspaper"]])
    # y = data['Sales']
    # model = sm.OLS(y, X).fit()
    # print(model.summary())

    # Predecir las ventas
    # prediction = lm4.predict([[1, inversionTV, inversionRadio, inversionNewspaper
    # prediction = model.predict([1, inversionTV, inversionRadio, inversionNewspape
```

```
    # Crear un nuevo dato con los valores de las características
    new_data_point = pd.DataFrame({'TV': [inversionTV], 'Radio': [inversionRadio],

    # Hacer la predicción para el nuevo dato
    prediction = lm4.predict(new_data_point)
    indexPrediction = len(sales_pred)+1

    # Añadir la prediccion a las predicciones
    # sales_pred.append(prediction, ignore_index=True)

    # Calcular las ganancias
    # ganancia = prediction - inversion

    print(f"Se ganó {prediction.values[0]} de la inversión.")

    # Visualizar la regresión lineal
    # plt.scatter(data["TV"], data['Sales'])
    # plt.scatter(data["Radio"], data['Sales'])
    # plt.scatter(data["Newspaper"], data['Sales'])
    # plt.plot(data["TV"], model.predict(X), color='red')
    # plt.plot(data["Radio"], model.predict(X), color='green')
    # plt.plot(data["Newspaper"], model.predict(X), color='blue')
    %matplotlib inline
    data.plot(kind = "scatter", x = "SalesTVRadioNewspaper", y="Sales")
    plt.plot(pd.DataFrame(data["SalesTVRadioNewspaper"]), sales_pred, c = "red", li
    # Predicción
    # plt.plot(indexPrediction, prediction.values[0], color='green')

    plt.scatter(inversionTV, prediction, color='red')
    plt.scatter(inversionRadio, prediction, color='green')
    plt.scatter(inversionNewspaper, prediction, color='blue')
    plt.xlabel('Inversión en Publicidad')
    plt.ylabel('Ventas')
    plt.title('Multiple Lineal Regression vs Ventas')
    plt.legend(["TV", "Radio", "Newspaper"])
    plt.show()

button = tk.Button(root, text="Enviar", command=get_inversion)
button.pack()

# Mostrar la ventana
root.mainloop()
```

```
La inversión en publicidad es de: 800.0
Se ganó 63.42346617655149 de la inversión.
```



In [ ]:
```
%matplotlib inline
data.plot(kind = "scatter", x = "SalesTVRadioNewspaper", y="Sales")
plt.plot(pd.DataFrame(data["SalesTVRadioNewspaper"]), sales_pred, c = "red", linewi
plt.title("Regresión Lineal Multiple TV & Radio & Newspaper vs Sales")
plt.show()
```


Regresión Lineal Multiple TV & Radio & Newspaper vs Sales
Sales
SalesTVRadioNewspaper
5
10
15
20
25
5
10
15
20
25