Title: Predicting Sales Based on Advertising Spend

Overview:

In today's competitive market, companies are constantly looking for ways to optimize their advertising budgets to maximize sales. Understanding the relationship between advertising expenditure across various media channels and sales performance is crucial for making informed marketing decisions. This dataset contains data on advertising expenditures across three channels: TV, Radio, and Newspaper, along with the resulting sales figures.

Objective:

The objective of this analysis is to build a predictive model that can accurately forecast sales based on the amount of money spent on TV, Radio, and Newspaper advertisements. This model will help in understanding the impact of each advertising channel on sales and in making strategic decisions regarding budget allocation to maximize sales.

1. Problem Definition

Objective: Clearly define the problem you want to solve.

2. Data Collection

Objective: Gather relevant data that will be used to train and evaluate the model. Tasks: Identify data sources, collect data, and ensure it is in a usable format.

3. Data Preparation

Objective: Clean and preprocess the data to make it suitable for modeling. Tasks: Handle missing values, remove duplicates, perform feature scaling, encode categorical variables, and split the data into training and testing sets.

4. Exploratory Data Analysis (EDA)

Objective: Understand the data and its underlying patterns. Tasks: Perform statistical analysis, visualize data distributions, and identify correlations or outliers.

5. Feature Engineering

Objective: Create new features or modify existing ones to improve model performance. Tasks: Transform features, create interaction terms, perform dimensionality reduction if necessary.

6. Model Selection

Objective: Choose the appropriate machine learning algorithms for the problem. Tasks: Compare different models, consider the nature of the problem (classification, regression, clustering, etc.), and select candidate models.

7. Model Training

Objective: Train the chosen models on the training dataset. Tasks: Configure model parameters, train models, and evaluate initial performance.

8. Model Evaluation

Objective: Assess the performance of the trained models using the testing dataset. Tasks: Use appropriate metrics (accuracy, precision, recall, F1-score, RMSE, etc.), perform cross-validation, and compare model performance.

9. Model Tuning

Objective: Optimize model performance by fine-tuning hyperparameters. Tasks: Use techniques like grid search, random search, or Bayesian optimization to find the best hyperparameters.

10. Model Deployment

Objective: Make the model available for use in a production environment. Tasks: Choose a deployment method (API, embedded system, cloud service), implement the deployment, and monitor the model in production.

Dataset Description:

The dataset consists of the following columns:

1. TV: Amount of money spent on TV advertising (in thousands of dollars).
2. Radio: Amount of money spent on Radio advertising (in thousands of dollars).
3. Newspaper: Amount of money spent on Newspaper advertising (in thousands of dollars).
4. Sales: Sales of the product (in thousands of units).

Tools and Technologies

Data Analysis and Visualization:

```
Python (Pandas, NumPy, Matplotlib, Seaborn)
```

In [1]:
```
#Data Analysis and Visualization:
#Python (Pandas, NumPy, Matplotlib, Seaborn)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as plt
```

In [2]:
```
# Remove the warnings
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
# Read and Visualize the dataset
df=pd.read_csv('advertising.csv')
df
```

Out[3]:

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

The dataset has contain 200 Rows and 4 Columns

In [4]:
```
# By using info()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

By using info method to information about the dataset. so as the dataset has taken 6.4 KB. it contains four attributes, they are TV, Radio ,Newspaper and Sales

In [5]:
```
# By using head()
df.head()
```

Out[5]:

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

Using the head method to view the top values of the data can help visualize the dataset.

In [6]:
```
# By using tail()
df.tail()
```

Out[6]:

	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

Using the head method to view the last(Bottom) values of the data can help visualize the dataset.

EDA(Exploratory Data Analysis)

Distribution of Data

In [7]:
```
# Describe method
df.describe()
```

Out[7]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

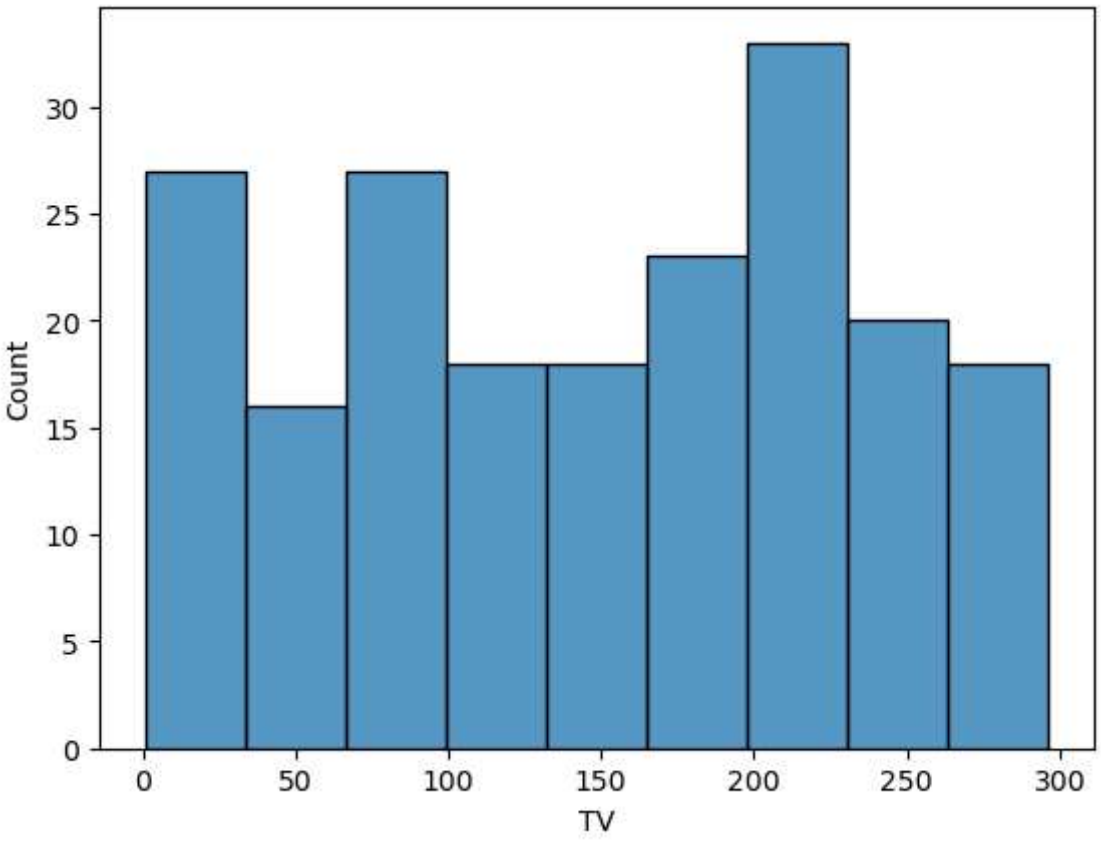
The describe method provides the Count, Mean, Standard deviation, Minimum, and Maximum values of each column.

Univariate Analysis

Univariate analysis focuses on a single variable to understand its internal structure. It is primarily concerned with describing the data and finding patterns existing in a single feature. This sort of evaluation makes a speciality of analyzing character variables inside the records set. It involves summarizing and visualizing a unmarried variable at a time to understand its distribution, relevant tendency, unfold, and different applicable records.

In [8]:
```
#histograms
sns.histplot(df,x='TV')
```

Out[8]:
```
<Axes: xlabel='TV', ylabel='Count'>
```

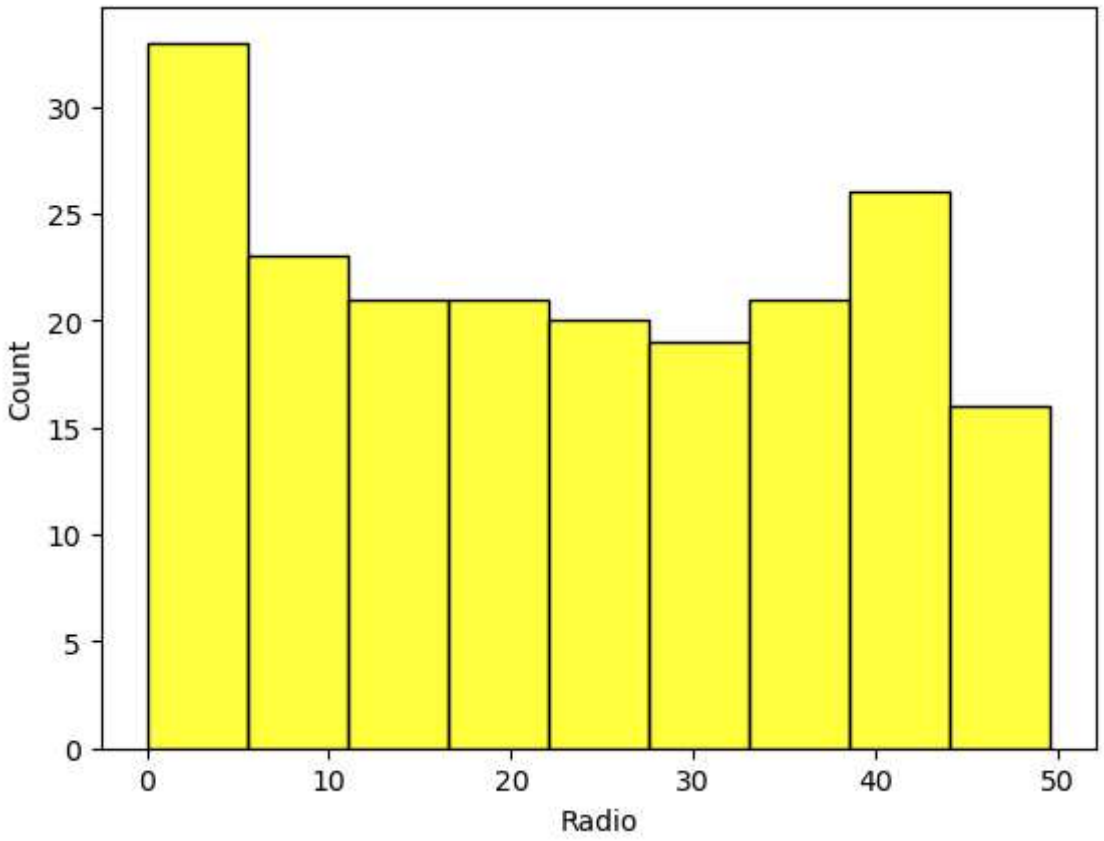


Histogram: Used to visualize the distribution of a variable, it takes the data on the x-axis and defaults to displaying the count on the y-axis.

sns.histplot(df,x='TV') here, to take the x-axis on TV and defaults to displaying the count on the y-axis.

In [9]:
```
#histograms
sns.histplot(df,x='Radio',color='yellow')
```

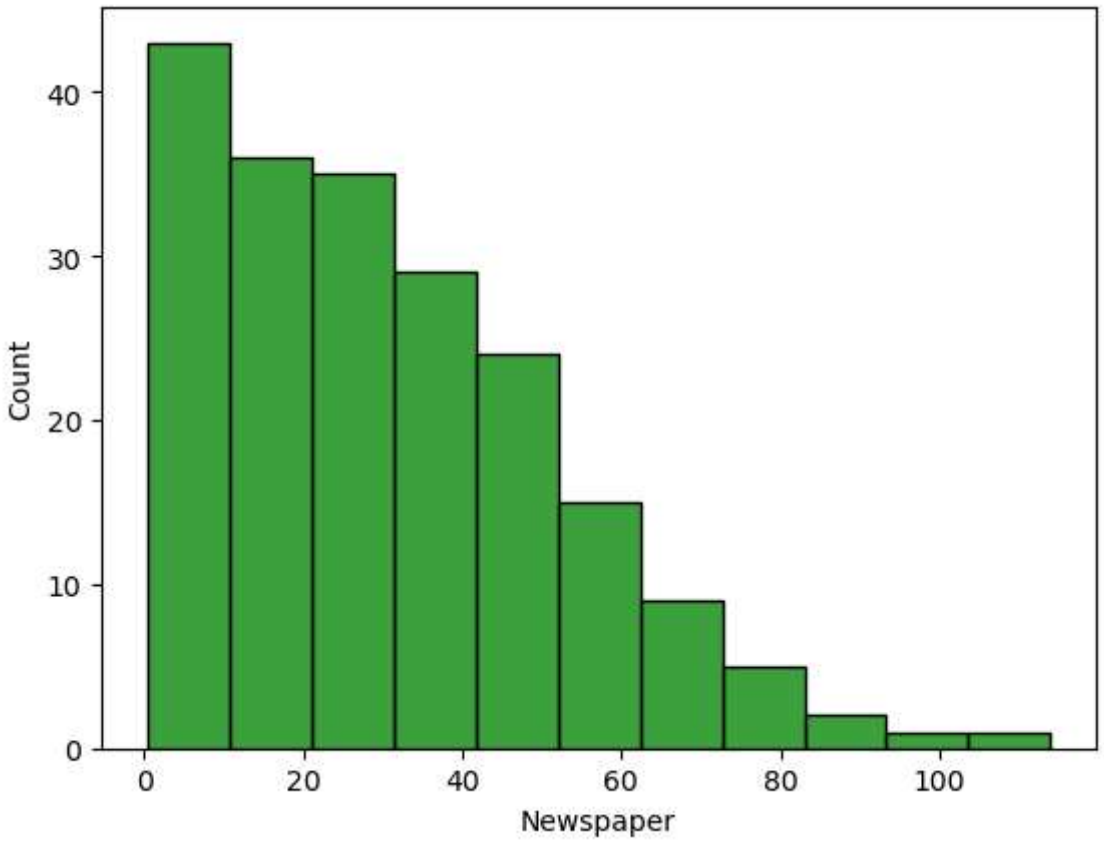
Out[9]:
```
<Axes: xlabel='Radio', ylabel='Count'>
```



sns.histplot(df,x='Radio') here, to take the x-axis on Radio and defaults to displaying the count on the y-axis.

In [10]:
```
#histograms
sns.histplot(df,x='Newspaper',color='green')
```

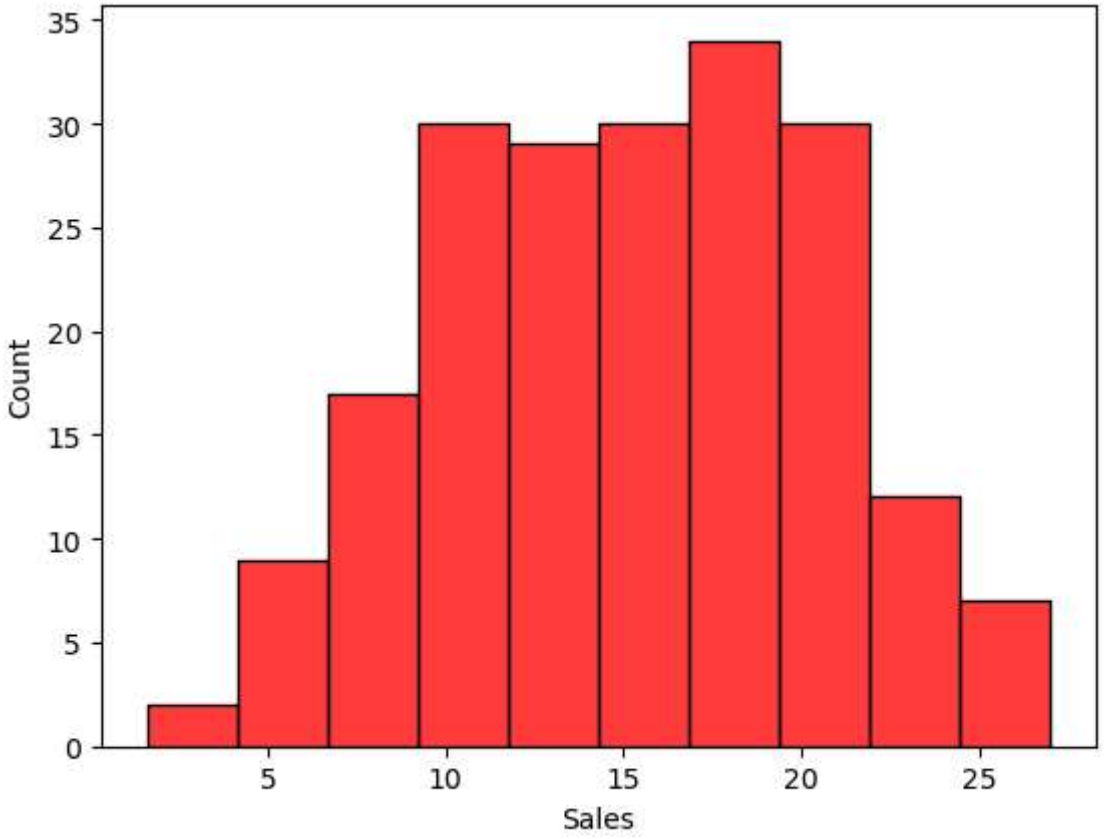
Out[10]:
```
<Axes: xlabel='Newspaper', ylabel='Count'>
```



sns.histplot(df,x='TV') here, to take the x-axis on Newspaper and defaults to displaying the count on the y-axis.

In [11]:
```
#histogram
sns.histplot(df,x='Sales',color='red')
```

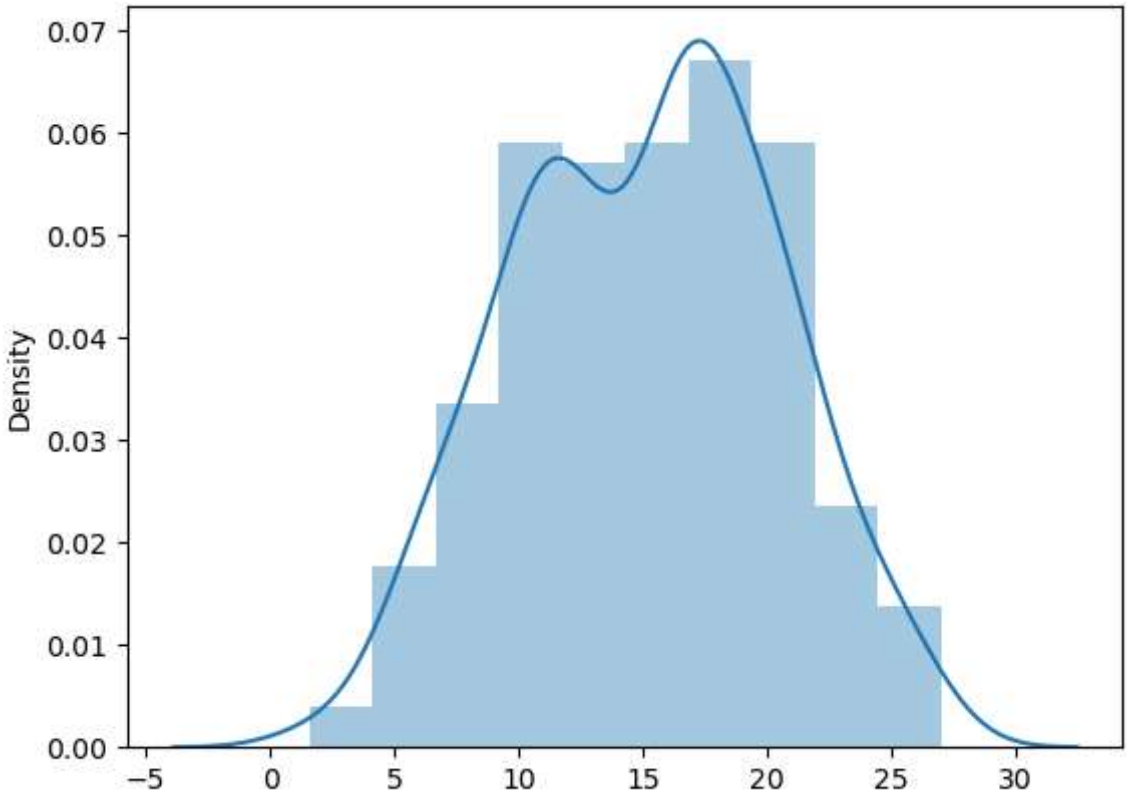
Out[11]:
```
<Axes: xlabel='Sales', ylabel='Count'>
```



Mmaximum sale of the production is 18 to 19
most of sales near to 34

In [12]:
```
sns.distplot(df,x=df['Sales'])
```

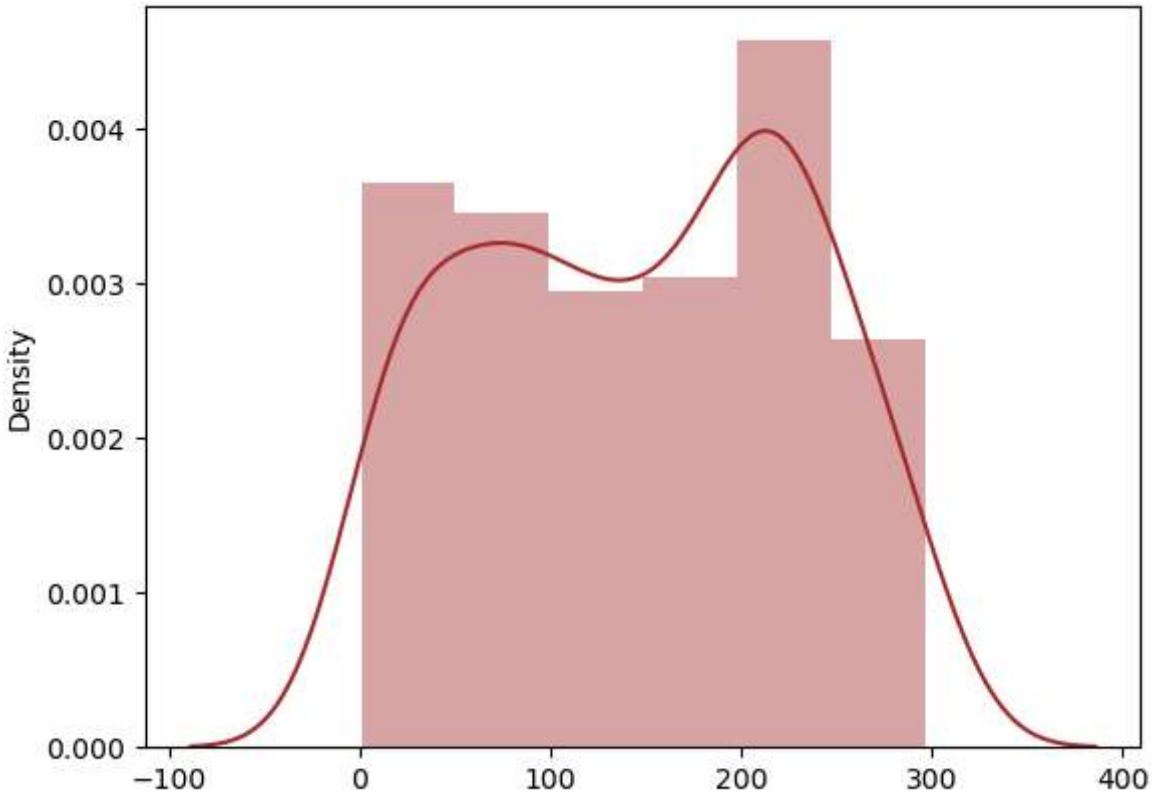
Out[12]:
```
<Axes: ylabel='Density'>
```



sns.histplot(df,x='Sales') here, to take the x-axis on sales and defaults to displaying the count on the y-axis.

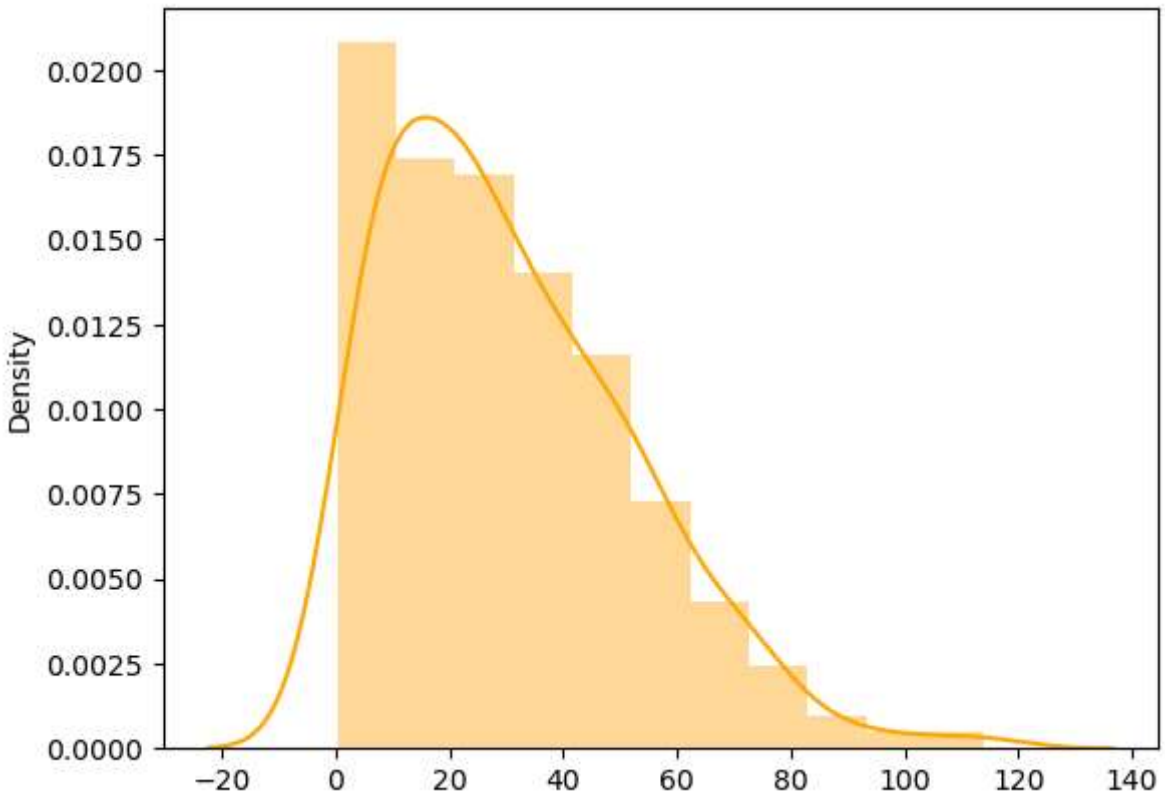
In [13]:
```
sns.distplot(df,x=df['TV'],color='brown')
```

Out[13]:
```
<Axes: ylabel='Density'>
```



In [14]:
```
sns.distplot(df,x=df['Newspaper'],color="orange")
```

Out[14]: <Axes: ylabel='Density'>

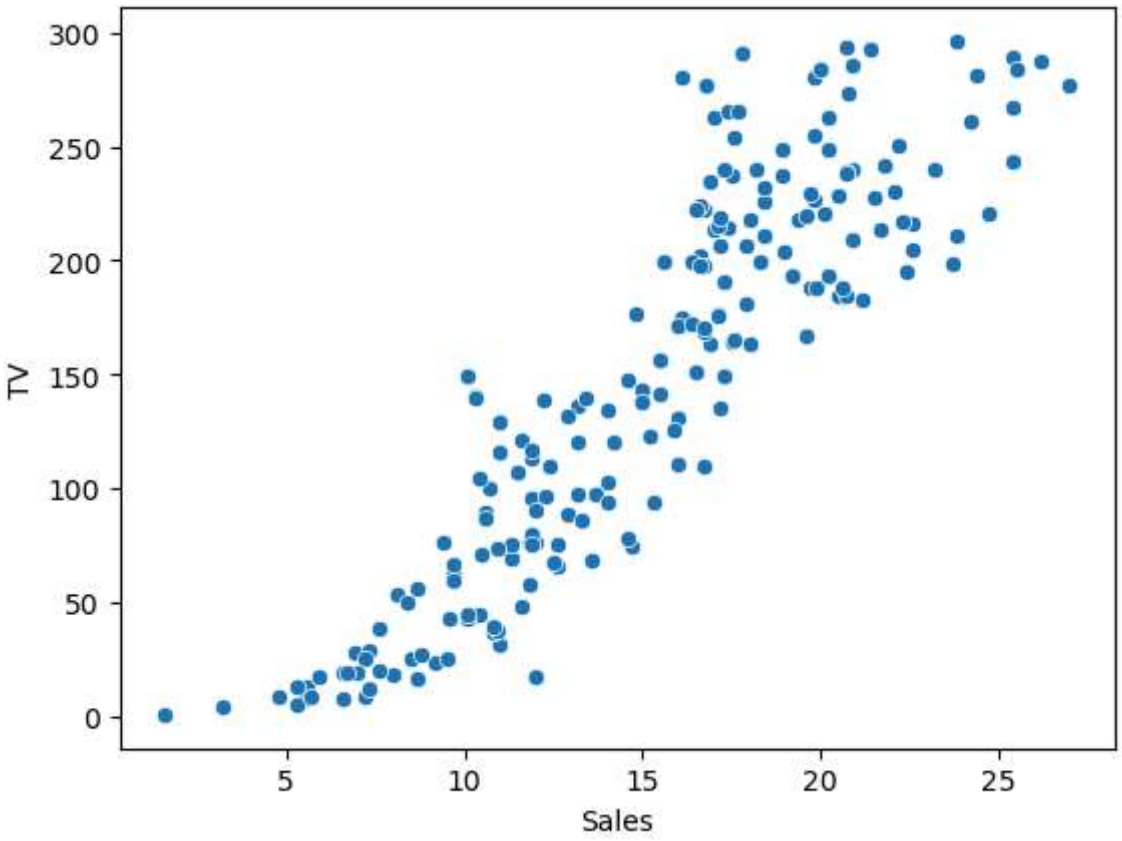


Bivariate analysis

A bivariate plot graphs the relationship between two variables that have been measured on a single sample of subjects. Such a plot permits you to see at a glance the degree and pattern of relation between the two variables

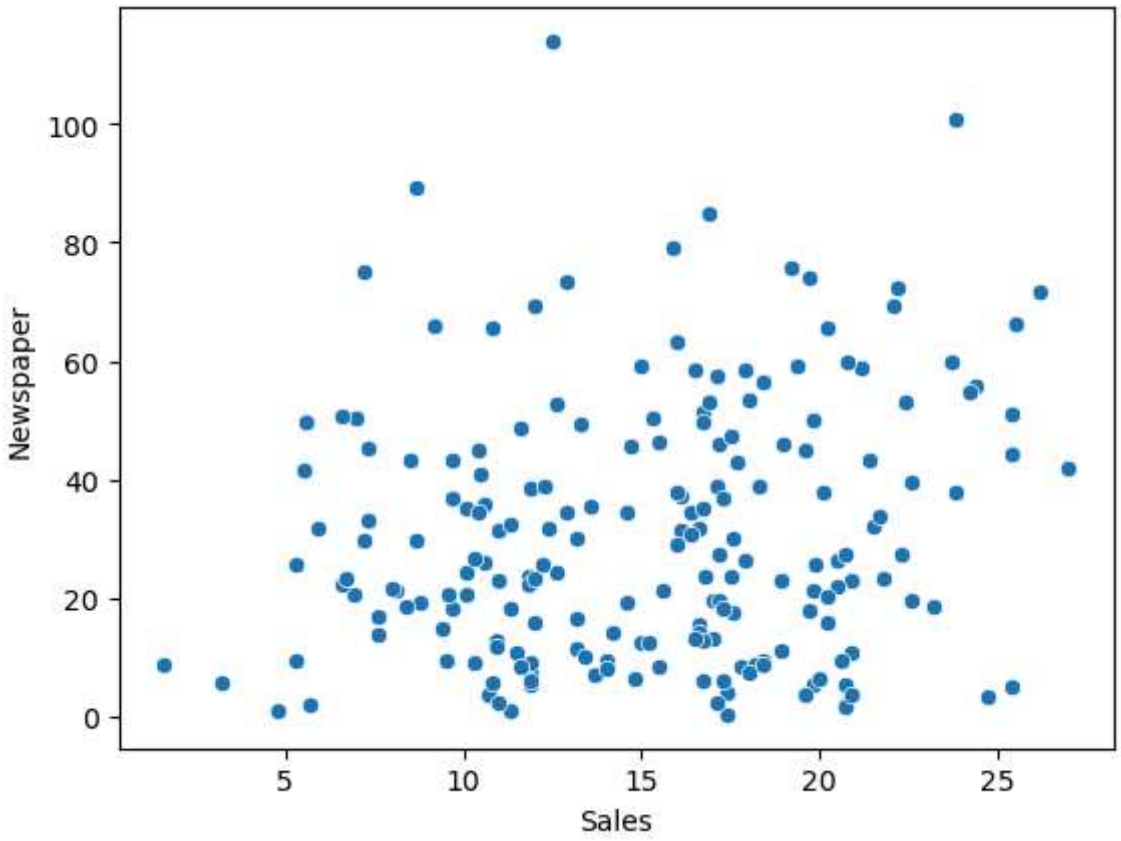
In [15]:
```
#Scatter plot
sns.scatterplot(df,x='Sales',y='TV')
```

Out[15]:
```
<Axes: xlabel='Sales', ylabel='TV'>
```



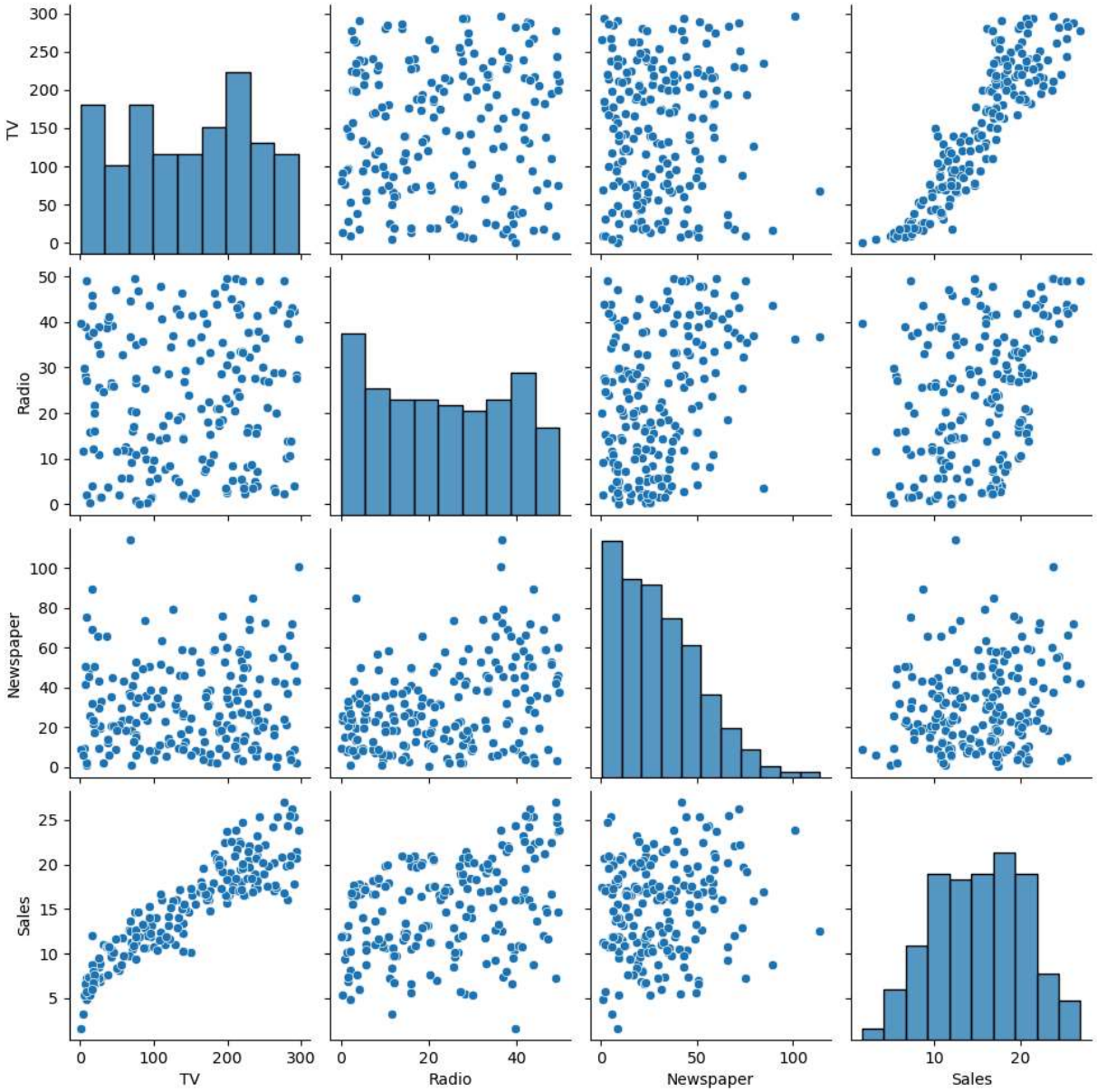
In [16]:
```
#scatterplot
sns.scatterplot(df,x='Sales',y="Newspaper")
```

Out[16]:
```
<Axes: xlabel='Sales', ylabel='Newspaper'>
```



In [17]:
```
sns.pairplot(df)
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x2e3882f7c90>
```



Split the data into (train,test)

A correlation is the statistical summary of the relationship between two sets of variables.

correlation

In [18]:
```
df.corr()
```

Out[18]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.901208
Radio	0.054809	1.000000	0.354104	0.349631
Newspaper	0.056648	0.354104	1.000000	0.157960
Sales	0.901208	0.349631	0.157960	1.000000

In [19]:
```
x=df[['TV','Newspaper','Radio']]
y=df['Sales']
```

Model evaluation

In [20]:
```
from sklearn.model_selection import train_test_split
```

The train_test_split function is a powerful tool in Scikit-learn's arsenal, primarily used to divide datasets into training and testing subsets. This function is part of the sklearn. model_selection module, which contains utilities for splitting data.

split the data into (X_train,X_test,y_train,y_test)

In [21]:
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [22]:
```
X_test.head()
```

Out[22]:

	TV	Newspaper	Radio
173	168.4	12.8	7.1
75	16.9	89.4	43.7
105	137.9	59.0	46.4
11	214.7	4.0	24.0
88	88.3	73.4	25.5

In [23]:
```
#shape
X_test.shape
```

Out[23]:
```
(40, 3)
```

In [24]:
```
X_train.shape
```

Out[24]:
```
(160, 3)
```

In [25]:
```
y_train.shape
```

Out[25]: (160,)

In [26]:
```
#shape
y_test.shape
```

Out[26]: (40,)

Import model

Simple Linear Regression

Problem Statement:

You have been provided with a dataset containing information about students' academic performance in their degree programs and their corresponding Job placement outcomes in terms of the packages received in lakhs per annum. Your task is to analyze the relationship between the percentages students recieve on their degree and the packages they receive as part of their job placements. Using simple linear regression, you aim to build a predictive model to estimate the packages received by students based on their academic performance percentages

In [32]:
```
from sklearn.linear_model import LinearRegression
```

In [33]:
```
L=LinearRegression()
```

fit()

In [34]:
```
L.fit(X_train,y_train)
```

Out[34]:
```
LinearRegression()
```

coefficient

The coefficient's value ranges between -1.0 and 1.0 while a calculated number larger than 1.0 indicates an error in the function. A coefficient of -1.0 shows a perfect negative correlation and 1.0 a perfect positive correlation

In [35]:
```
L.coef_
```

Out[35]:
```
array([0.05490744, 0.00070109, 0.10940849])
```

interception

Intercepts allows you to intercept function calls in Python and handle them in any manner you choose. For example, you can pre-process the inputs to a function, or apply post-processing on its output. Intercepts also allows you to completely replace a function with a custom implementation.

In [40]:
```
L.intercept_
```

Out[40]:
```
5.202208501844554
```

In [42]:
```
y_predicted=L.predict(X_test)
```

In [45]:
```
y_predicted
The precision of a numerical quantity is a measure of the detail in which the quant
```

Out[45]:
```
array([14.75910798, 10.25165305, 16.97749347, 18.91737055, 12.22986341,
       19.36244209, 20.68507437, 20.965965  ,  5.8683092 , 17.23366948,
        7.7737537 ,  8.16903024, 17.0666073 , 18.77347246, 17.00015017,
       15.95457579, 21.89913413, 20.15430524, 23.12195092, 19.35900062,
       16.88463129, 18.63786898,  9.87687722,  6.77705279, 20.64200537,
       16.57603388,  8.95519522, 18.27776123,  7.5522362 ,  8.24308659,
       19.95890333, 11.94272958,  9.40683816, 19.36421514, 19.88984598,
        9.20737844, 23.86771179, 20.44743136, 21.27468907, 17.19457921])
```

Mean Absolut Error

In [48]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [51]:
```
y_pred = L.predict(X_test)
```

In [52]:
```
y_test.values
```

Out[52]:
```
array([16.7,  8.7, 15. , 17.4, 12.9, 17. , 21.7, 22.3,  5.3, 16.5,  6.6,
        8.1, 20.2, 19.2, 20.5, 16.7, 20.2, 22.4, 24.2, 18. , 19.9, 19.8,
        8.8,  7.3, 17.6, 17.2,  8.4, 18.4,  7.6,  7. , 16.8, 12. ,  7.3,
       17.7, 19.4, 12.6, 24.4, 18.9, 20.2, 20.6])
```

MSE

The Mean Square Error (MSE) is a crucial metric for evaluating the performance of predictive models. It measures the average squared difference between the predicted and the actual

target values within a dataset.

In [53]:
```
mse=ae=mean_absolute_error(y_test,y_pred)
```

In [54]:
```
print("Mean Squared Error (MSE):",mse)
```

```
Mean Squared Error (MSE): 1.4485471984748728
```

RMSE

The Root Mean Squared Error (RMSE) is one of the two main performance indicators for a regression model. It measures the average difference between values predicted by a model and the actual values. It provides an estimation of how well the model is able to predict the target value (accuracy).It is a popular metric used in machine learning and statistics to measure the accuracy of a predictive model. It quantifies the differences between predicted values and actual values, squaring the errors, taking the mean, and then finding the square root.

In [55]:
```
rmse=mse**0.5
data_rmse = {'Actual (y_test)':y_test, 'Predicted (y_pred)':y_pred}
df_rmse = pd.DataFrame(data_rmse)
df_rmse.head()
```

Out[55]:

	Actual (y_test)	Predicted (y_pred)
173	16.7	14.759108
75	8.7	10.251653
105	15.0	16.977493
11	17.4	18.917371
88	12.9	12.229863

R2S

In [59]:
```
r2s=r2_score(y_test,y_pred)
```

In [60]:
```
print("R Square Score(R2S):",r2s)
```

```
R Square Score(R2S): 0.8953363556675344
```

11. Monitoring and Maintenance

Objective: Ensure the model remains effective over time. The data can be split into test and train data, and also be used for fitting the coefficients of the model

12. Documentation and Reporting

Objective: Document the process, findings, and results of the project. Tasks: Create comprehensive reports, include visualizations and key metrics, and document the code and methodology.

Introduction

```
Exploratory Data Analysis (EDA) is a crucial step in the data
science process. It involves summarizing the main characteristics
of a dataset, often using visual methods. EDA helps in
understanding the data distribution, identifying patterns,
detecting anomalies, and testing hypotheses.
```

Dataset Overview

```
1.Source: [advertising.csv]
2.Description: ["TV","Radio","Newspaper","Sales"]:
3.Number of observations: [Total rows:"200"]
4.Number of features: [Total columns:"4"]
```

Conclusion

```
The EDA provides a comprehensive understanding of the dataset,
which is crucial for feature engineering, model selection, and
further analysis. It helps in making informed decisions throughout
the machine learning workflow.
How does EDA behave in machine learning, and what is the importance
and benefit of EDA in ML (Machine Learning)
To learn how to split the data, predict data, and overwrite the
data using Data Visualization.like as
-describe function
-head function
-tail function
-is_null.sum function
-info function
-plots
-univariate analysis
-Bivariate analysis
```

Machine learning:

```
Import the model
-coefficient
```

```
-intercept
-fit
-model
-accuracy
-predict the data
```

The all the information can be learn to use the above the project that can be solve the real world problems

In [ ]: