**Phase 5 : Project Documentation & Submission – Final Submission**



**Name: C.Arthi**

**Register Number : 312621243003**

**College Name : Thangavelu Engineering College**

**Project 3 : Future Sales Prediction**

**Project 3: Future Sales Prediction**

**Objective:**

The objective is to create a tool that enables the company to optimize inventory management and make informed business decisions based on datadriven sales predictions In this part we understand the problem statement and we created a document on what have we understood and we proceeded ahead with solving the problem. The problem is to develop a predictive model that uses historical sales data to forecast future sales for a retail company.

**Problem Definition:**

The problem is to develop a predictive model that uses historical sales data to forecast future sales for a retail company. This project involves data preprocessing, feature engineering, model selection, training, and evaluation. This model will predict sales on a certain day after being provided with a certain set of inputs.

**Design Thinking:**

**Data Source**:

* Utilize a dataset containing historical sales data. Here a csv file is converted to a DataFrame and the pandas object is used.
* The head() method returns a specified number of rows, string from the top. The head() method returns the first 5 rows if a number is not specified.
* The describe() method returns description of the data in the DataFrame. We calculate and print the summary statistics of the dataset using df.describe() function . The describe() method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains these information for each column such as count , mean , std , min , 25% , 50% , 75% , max .

**Data Cleaning and Preprocessing:**

* In order to check missing values in Pandas DataFrame, we use a function isnull() and notnull(). Both function help in checking whether a value is NaN or not. These function can also be used in Pandas Series in order to find null values in a series.
* Now we set some range for each variable and performs the range checks. It is performed between each columns of the test dataset.
* Now we are going to check wheather the given datas entered are correct or not by checking the non negative values in the data.
* The drop\_duplicates() method removes duplicate rows. Use the subset parameter if only some specified columns should be considered when looking for duplicates.
* Outliers are data points that significantly differ from the rest of the observations in a dataset. They can be unusually high or low values compared to the majority of the data. In statistical terms, outliers are observations that fall outside of the typical range of values.Outliers can arise due to various reasons, such as errors in data collection, measurement variability, or the presence of rare events. They have the potential to skew statistical analyses and machine learning models, leading to misleading or inaccurate results.Detecting and handling outliers is an important step in data preprocessing and analysis to ensure that the insights drawn from the data are robust and representative of the underlying patterns.
* Now here we are extracting the dependent variable and independent variables .The ‘sales’ is a dependent variable and the other are the independent variables .
* Now here the binning method is to smooth or handle noisy data. First, the data is sorted then, and then the sorted values are separated and stored in the form of bins.
* A correlation matrix is a table containing correlation coefficients for many variables. Each cell in the table represents the correlation between two variables. The value might range between -1 and 1.
* StandardScaler removes the mean and scales each feature/variable to unit variance. This operation is performed feature-wise in an independent way. StandardScaler can be influenced by outliers (if they exist in the dataset) since it involves the estimation of the empirical mean and standard deviation of each feature.
* K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.  groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

**Adding Features:**

Here we add some features to the dataset for some accuracy

* Here we have created a new column Total\_Spent in our DataFrame df by summing up the expenses from 'TV', 'Radio', and 'Newspaper'. This can be a useful feature for our sales prediction model, as it captures the total advertising expenditure.
* Here we have added a new column called 'Previous\_Sales' to our DataFrame by shifting the 'Sales' column by one position. This creates a lagged version of the sales data.
* Now we have created a new column called TV\_Radio\_Interact in our DataFrame df by multiplying the 'TV' and 'Radio' columns. This is an example of feature engineering, which can potentially improve the performance of our predictive model.
* Here we have added a new feature named 'TV\_log' which represents the logarithm (base) of the 'TV' column. This transformation can be useful if the relationship between 'TV' and the target variable is non-linear.T he code we provided will calculate the natural logarithm of the 'TV' column and assign it to the new 'TV\_log' column in our DataFrame ‘df’ .Keep in mind that this transformation may help linearize the relationship between 'TV' and the target variable, which can potentially improve the performance of our linear regression model. However, it's always a good idea to evaluate the impact of this transformation on our model's performance using appropriate evaluation metrics.

**Model Selection and Training:**

Here the Linear regression is selected as the model and trained . This model is selected here for the prediction and accuracy .

# Linear Regression:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables . Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

**Import the necessary libraries:**Certainly! To run the code, we'll need to import the necessary libraries.

**Load and preprocess our dataset:**We have successfully created the features (X) and target variable (y) and then split the data into training and testing sets using the train\_test\_split function. This is a common and essential step in machine learning workflows.Here's a quick summary of what we've done:

* We selected the features 'TV', 'Radio', and 'Newspaper' from our DataFrame df and assigned them to X. This will be used as input to train our model.
* We selected the 'Sales' column from our DataFrame df and assigned it to y. This will be our target variable that we want to predict.
* We used the train\_test\_split function to split our data into training and testing sets.
* The training set (X\_train and y\_train) will be used to train the model, and the testing set (X\_test and y\_test) will be used to evaluate the model's performance.
* The test\_size=0.2 argument indicates that 20% of the data will be used for testing, while 80% will be used for training.
* The random\_state=42 argument ensures that the data is split in a reproducible manner (the same split will occur every time we run the code), which can be important for debugging and comparing different models.

**Initialize and train the Linear Regression model:**We have correctly initialized and trained a Linear Regression model by the appropriate code and algorithm .

**Predict using the model:**We have successfully used your trained model to make predictions on the test set (X\_test). The predicted values are stored in the variable y\_pred.This will display the array of predicted values for the test set. Each element in the array corresponds to the predicted value for a specific instance in your test set.

Now we create some scatter plot for this model . the below code will create a scatter plot where the x-axis represents the actual sales values (y\_test) and the y-axis represents the predicted sales values (y\_pred). If the model predictions are accurate, the points should fall close to a diagonal line.The visualization is effective for understanding the relationship between actual and predicted values in a regression model.

**Evaluate the model:**

* Mean Squared Error (MSE) measures the amount of error in a statistical model. Evaluate the mean squared difference between observed and predicted values. If the model has no errors, the MSE is zero. Its value increases as the model error increases.
* absolute error refers to the magnitude of difference between the prediction of an observation and the true value of that observation.The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables.
* The root mean square error (RMSE) measures the average difference between a statistical model's predicted values and the actual values. Mathematically, it is the standard deviation of the residuals. Residuals represent the distance between the regression line and the data points.
* R2 is a measure of the goodness of fit of a model. In regression, the R2 coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points. An R2 of 1 indicates that the regression predictions perfectly fit the data.

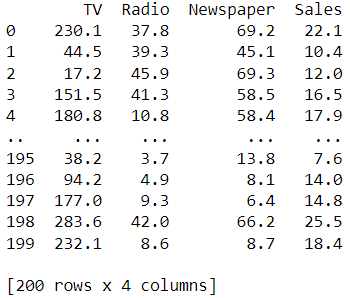
Finally Here printing the coefficients of the data model

**Code and Explanation :**

import pandas as pd

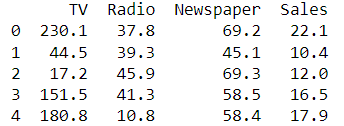
df=pd.read\_csv(r'Sales.csv')

print(df)



# Print the first few rows of the DataFrame

print(df.head())

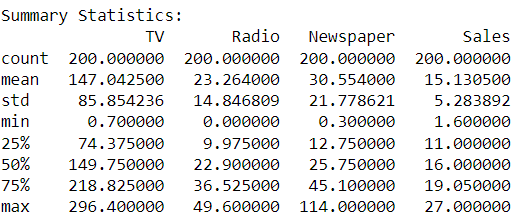


# Check basic statistics

summary\_stats = df.describe()

print("\nSummary Statistics:")

print(summary\_stats)



#checking wheather the data is available or not recorded.

missing\_values = df.isnull().sum()

# Print the missing values (if any)

print("Missing Values:")

print(missing\_values)

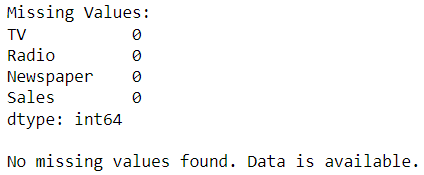
# You can also print a message based on the result

if missing\_values.sum() == 0:

print("\nNo missing values found. Data is available.")

else:

print("\nMissing values found. Data may be incomplete or recorded incorrectly.")



#Define reasonable ranges for each variable

reasonable\_ranges = {

'TV': (0, 1000), # Example: TV budget should be between 0 and 1000

'Radio': (0, 100), # Example: Radio budget should be between 0 and 100

'Newspaper': (0, 200), # Example: Newspaper budget should be between 0 and 200

'Sales': (0, 100) # Example: Sales should be between 0 and 100

}

# Perform range checks

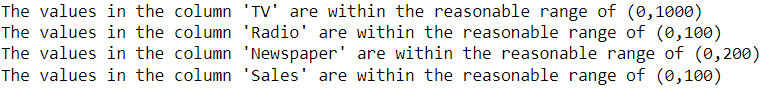
for column, (min\_val, max\_val) in reasonable\_ranges.items():

if ((df[column] < min\_val) | (df[column] > max\_val)).any():

print(f"Warning: Values in column '{column}' are not within the reasonable range of ({min\_val}, {max\_val}). Please verify the data.")

else:

print(f"The values in the column '{column}' are within the reasonable range of ({min\_val},{max\_val})")



#To check whether the data entered is correct or not.

# Checking if sales values are non-negative

if (df['Sales'] < 0).any():

print("Warning: There are negative sales values. Please verify the data.")

else:

print("The Data contains non negative sales values")



# Check for negative or unrealistic values

if (df < 0).any().any():

print(f"Warning: There are negative values in the dataset. Please verify the data.")

else:

print("No negative values found.")



# Checking if advertising budgets are non-negative

if (df[['TV', 'Radio', 'Newspaper']] < 0).any().any():

print("Warning: There are negative advertising budget values. Please verify the data.")

else:

print("The Data contains non negative advertising budget values")



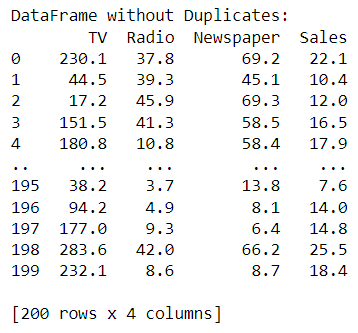
#to remove duplicate values

df\_dup = df.drop\_duplicates()

# Display the DataFrame without duplicates

print("DataFrame without Duplicates:")

print(df\_dup)



#Check for outliers and decide whether to remove them or not.

import numpy as np

# Define a function to detect outliers using IQR method

def detect\_outliers(data):

Q1 = data.quantile(0.25)

Q3 = data.quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

return (data < lower\_bound) | (data > upper\_bound)

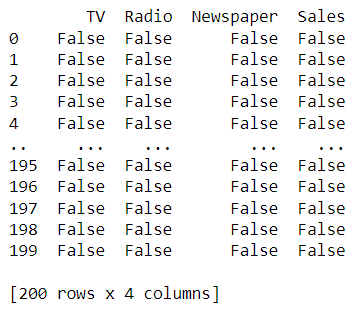
# Select the columns you want to check for outliers

columns\_to\_check = ['TV', 'Radio', 'Newspaper', 'Sales']

# Check for outliers in each column

outliers = df[columns\_to\_check].apply(detect\_outliers)

print(outliers)



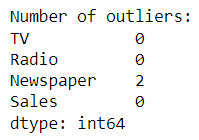
# Count the number of outliers in each column

num\_outliers = outliers.sum()

# Display the number of outliers for each column

print("Number of outliers:")

print(num\_outliers)

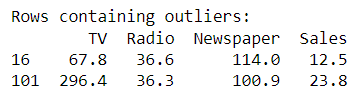


# Display the rows containing outliers

outliers\_rows = df[outliers.any(axis=1)]

print("\nRows containing outliers:")

print(outliers\_rows)

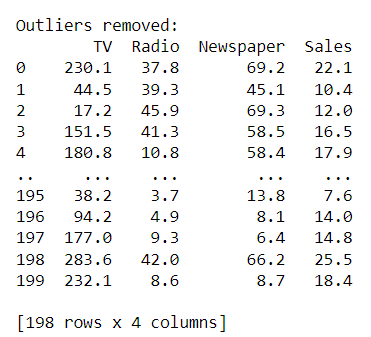


# To remove outliers, you can use something like this:

df\_cleaned = df[~outliers.any(axis=1)]

print("\nOutliers removed:")

print(df\_cleaned)

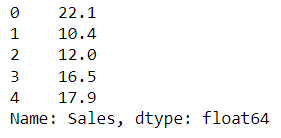


# Extracting the dependent variable 'Sales'

y = df['Sales']

# Printing the first few rows of 'Sales' to verify

print(y.head())



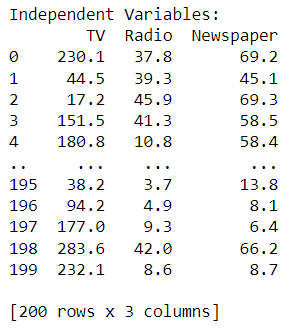
# Extracting the independent variables into a DataFrame

independent\_variables = df[['TV', 'Radio', 'Newspaper']]

# Printing the extracted independent variables

print("Independent Variables:")

print(independent\_variables)



#Bining the data

# Adjust the bin\_edges and bin\_labels as per your specific requirements

bin\_edges = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

bin\_labels = ['0-10', '10-20', '20-30', '30-40', '40-50', '50-60', '60-70', '70-80', '80-90', '90-100']

#Apply binning to a specific column (e.g., 'Sales')

df['Sales\_bin'] = pd.cut(df['Sales'], bins=bin\_edges, labels=bin\_labels, include\_lowest=True)

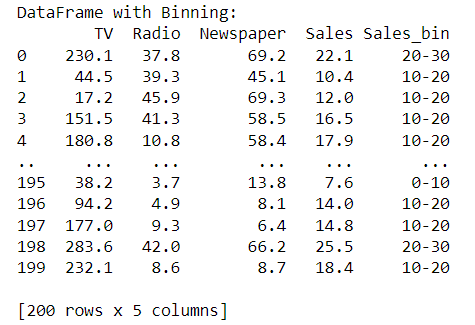
# You can choose a different column

#or adjust bin\_edges and bin\_labels based on your specific requirements.

# Display the DataFrame with the new binning column

print("DataFrame with Binning:")

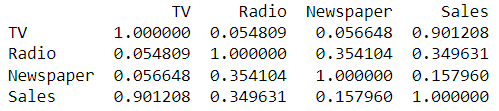
print(df)



# Correlation matrix

correlation\_matrix = df.corr()

print(correlation\_matrix)



from sklearn.preprocessing import StandardScaler

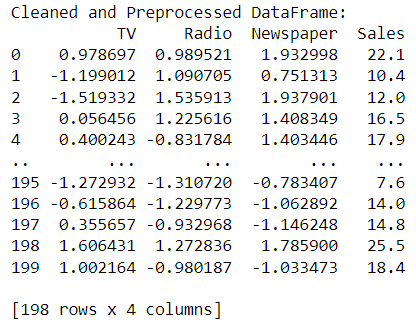
scaler = StandardScaler()

numerical\_features = ['TV', 'Radio', 'Newspaper']

df\_cleaned[numerical\_features] = scaler.fit\_transform(df\_cleaned[numerical\_features])

print("Cleaned and Preprocessed DataFrame:")

print(df\_cleaned)



from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

#Select the features for clustering (e.g., 'TV', 'Radio', 'Newspaper')

features = df[['TV', 'Radio', 'Newspaper']]

#Choose the number of clusters (k)

k = 3 # Adjust based on your specific analysis

#Perform K-Means clustering

kmeans = KMeans(n\_clusters=k, random\_state=0)

df['Cluster'] = kmeans.fit\_predict(features)

# Visualize the clusters (for 2D visualization)

plt.scatter(df['TV'], df['Radio'], c=df['Cluster'], cmap='viridis')

plt.xlabel('TV')

plt.ylabel('Radio')

plt.title('K-Means Clustering')

plt.show()

# The above visualization assumes 'TV' and 'Radio' as features. You can adjust based on your specific features.

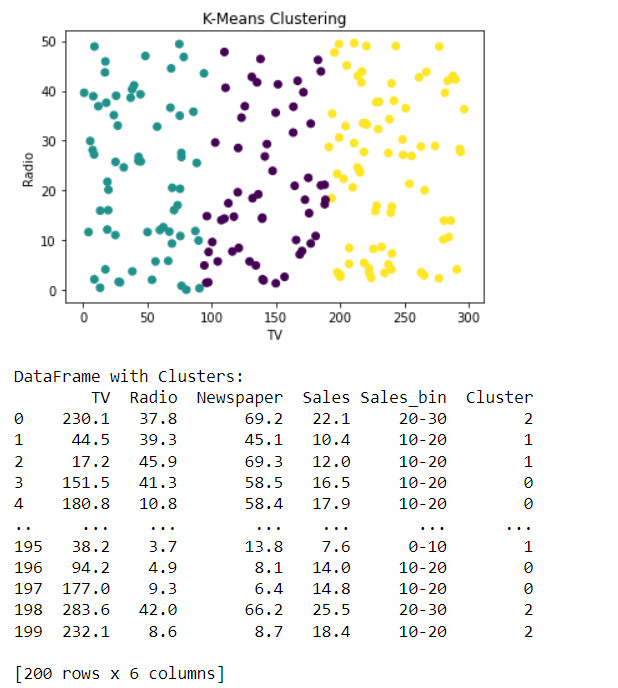
#Optional - Perform further analysis on the clusters

# You can analyze the clusters further to understand their characteristics.

# Display the DataFrame with cluster assignments

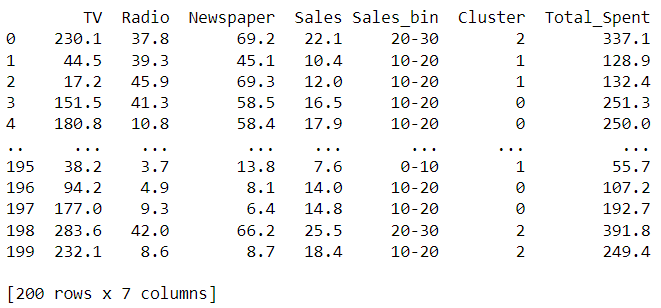
print("DataFrame with Clusters:")

print(df)



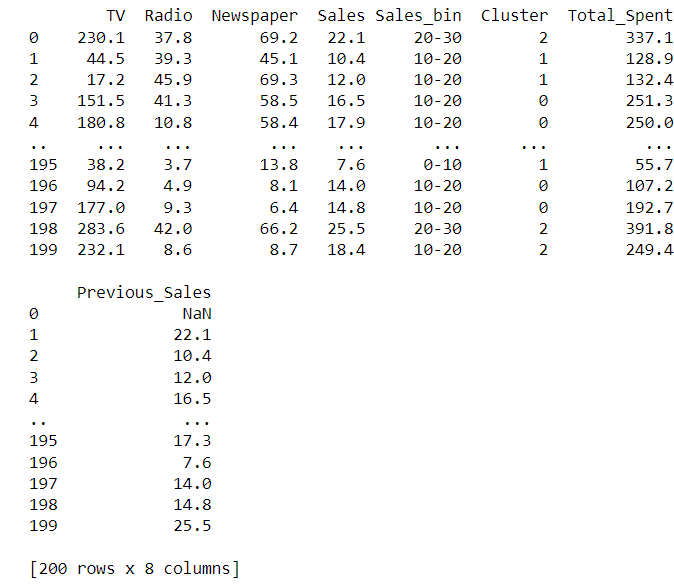
df['Total\_Spent'] = df['TV'] + df['Radio'] + df['Newspaper']

print(df)

****

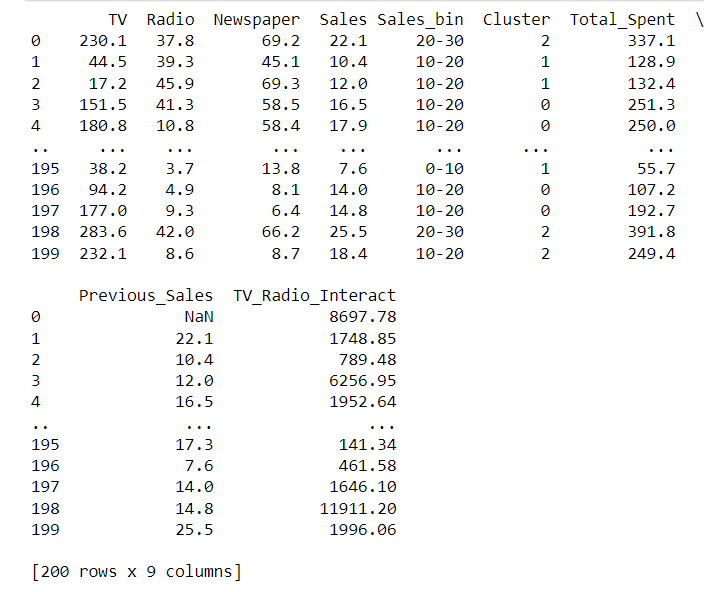
df['Previous\_Sales'] = df['Sales'].shift(1) # Lagged sales

print(df)



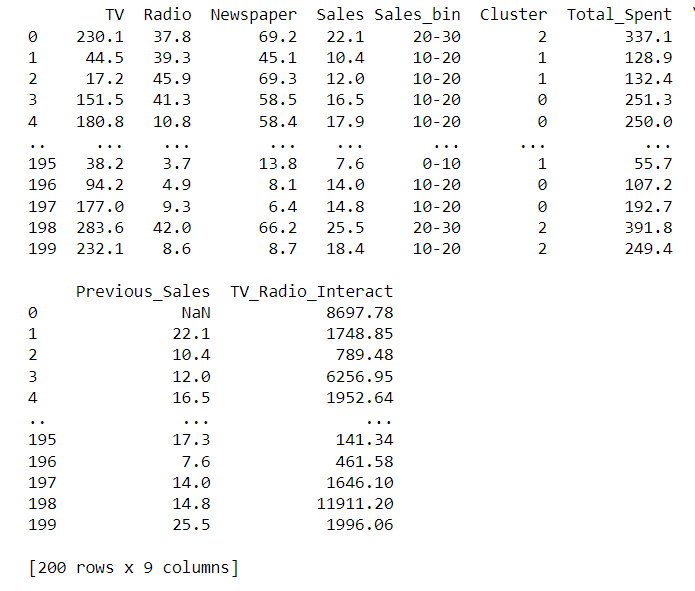
df['TV\_Radio\_Interact'] = df['TV'] \* df['Radio']

print(df)



df['TV\_Radio\_Interact'] = df['TV'] \* df['Radio']

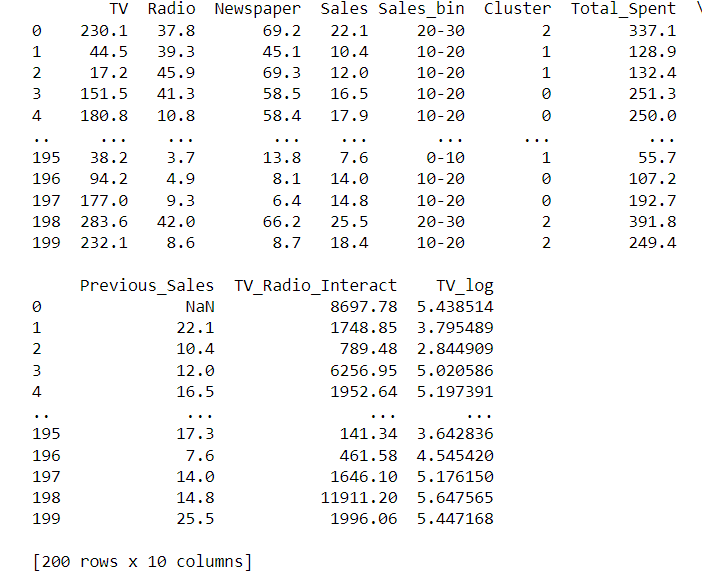
print(df)



import numpy as np

df['TV\_log'] = np.log(df['TV'])

print(df)



Import the necessary libraries:

from sklearn.linear\_model import LinearRegression

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

import pandas as pd

Load and preprocess your dataset:

X = df[['TV', 'Radio', 'Newspaper']] # Features

y = df['Sales'] # Target variable

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Initialize and train the Linear Regression model:

# Initialize the Linear Regression model

model = LinearRegression()

# Train the model on the training data

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

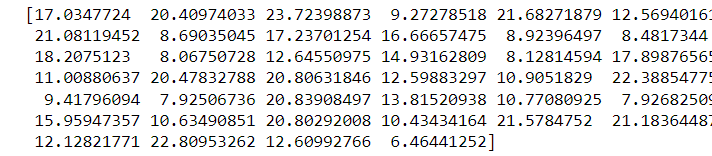


Predict using the model:

# Predict on the test set

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

print(y\_pred)



import matplotlib.pyplot as plt

# Assuming 'y\_test' and 'y\_pred' are your actual and predicted values, respectively

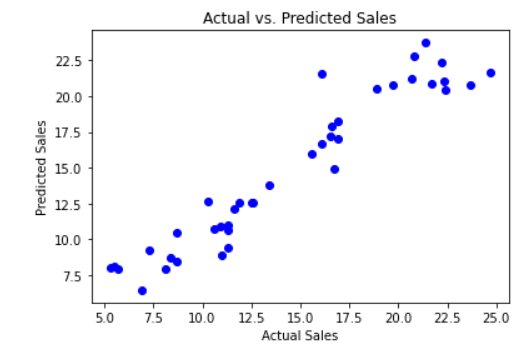
plt.scatter(y\_test, y\_pred, color='blue')

plt.xlabel('Actual Sales')

plt.ylabel('Predicted Sales')

plt.title('Actual vs. Predicted Sales')

plt.show()



from sklearn.metrics import mean\_squared\_error

# Calculate the Mean Squared Error (MSE)

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

print(f"Mean Squared Error: {mse}")



from sklearn.metrics import mean\_absolute\_error

# Calculate MAE

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

print(f"Mean Absolute Error: {mae}")



import numpy as np

# Calculate RMSE

rmse = np.sqrt(mse)

print(f"Root Mean Squared Error: {rmse}")



from sklearn.metrics import r2\_score

# Calculate R2 score

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

print(f"R-squared (R2) Score: {r2}")



# Print the coefficients

print("Coefficients:")

for feature, coef in zip(X.columns, model.coef\_):

print(f"{feature}: {coef}")

print(f"Intercept: {model.intercept\_}")

