**Future Sales Prediction with Machine Learning**

**Code Snippet #1: Code Snippet of Importing Libraries and Loading Data**

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

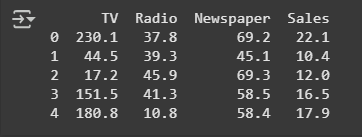
data = pd.read\_csv("advertising.csv")

print(data.head())

**Function:**

This code imports the necessary libraries for data manipulation, numerical operations, and machine learning. pandas is imported as pd to handle and manipulate tabular data, specifically loading the CSV file into a DataFrame. numpy is imported as np to work with numerical data and arrays. The train\_test\_split function from sklearn.model\_selection is used for splitting data into training and testing sets, and LinearRegression from sklearn.linear\_model is imported to build and train a linear regression model. The data variable loads the data from the advertising.csv file, and the print(data.head()) command displays the first five rows of the dataset to give an initial look at its structure.

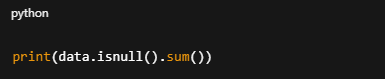
**Figure#1:**



**Description:**

After executing this code, the necessary libraries are imported, making it possible to manipulate the data, perform numerical operations, and build a linear regression model. The advertising.csv dataset is loaded into the data DataFrame, and the print(data.head()) function displays the first five rows of the dataset. This helps you quickly understand the structure and contents of the dataset before proceeding with further analysis or modeling tasks.

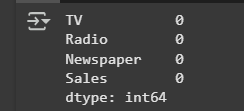
**Code Snippet #2: Code Snippet of Checking for Missing Values**



**Function:**

This code checks for any missing values (null values) in the data DataFrame. The isnull() function returns a DataFrame of the same shape as data, where each entry is a boolean indicating whether the corresponding value is missing (True for missing, False for not missing). The sum() function then adds up the True values (which are treated as 1), showing the total number of missing values for each column in the dataset.

**Figure#2:**



**Description:**

The output will display the total number of missing values (if any) for each column in the dataset. This allows you to quickly identify which features in the dataset have missing data, which is an important step in data cleaning. If any columns have missing values, you can take appropriate actions like filling or dropping those values before proceeding with analysis or model training.

**Code Snippet #3: Code Snippet of Scatter Plot with Trendline**

import plotly.express as px

import plotly.graph\_objects as go

figure = px.scatter(data\_frame = data, x="Sales",

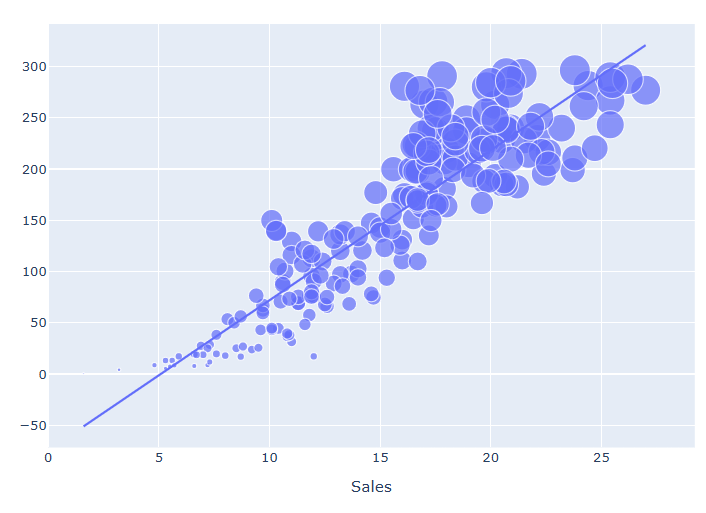
                    y="TV", size="TV", trendline="ols")

figure.show()

**Function:**

This code generates an interactive scatter plot using the plotly.express library. It uses the px.scatter() function to plot the Sales column on the x-axis and the TV column on the y-axis. The size of each data point is determined by the values in the TV column. Additionally, a trendline is added using the "ols" (Ordinary Least Squares) method to visually represent the linear relationship between Sales and TV. The plot is then displayed using figure.show().

**Figure#3:**



**Description:**

The output will be an interactive scatter plot that shows the relationship between TV advertising spending (TV) and product sales (Sales). The size of each point is proportional to the amount spent on TV advertising. The plot will also display a linear trendline, which highlights the overall correlation between sales and TV advertising. This visualization helps to quickly identify patterns, trends, or potential outliers in the relationship between these two variables.

**Code Snippet #4: Code Snippet of Scatter Plot with Trendline for Newspaper vs Sales**

figure = px.scatter(data\_frame = data, x="Sales",

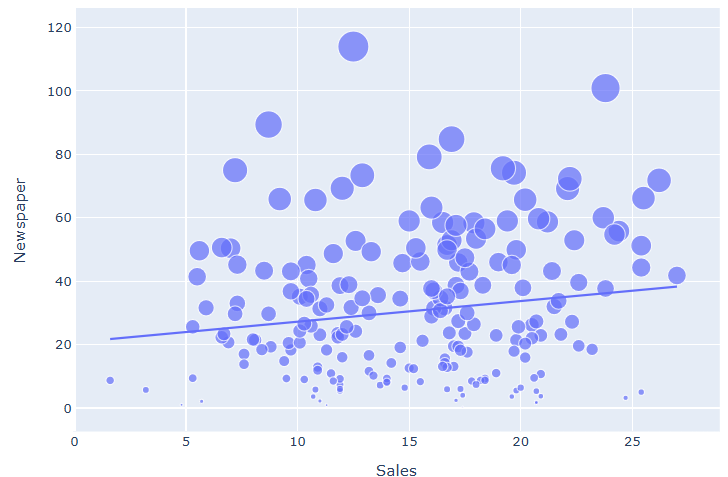
                    y="Newspaper", size="Newspaper", trendline="ols")

figure.show()

**Function:**

This code generates an interactive scatter plot using the plotly.express library. It plots the Sales column on the x-axis and the Newspaper column on the y-axis. The size of each data point is determined by the values in the Newspaper column, indicating the amount spent on newspaper advertising. A trendline is added using the "ols" (Ordinary Least Squares) method, which visually represents the linear relationship between Sales and Newspaper. The plot is displayed using figure.show().

**Figure#4:**



**Description:**

The output will be an interactive scatter plot showing the relationship between sales and newspaper advertising spend. Each point will represent a data record, with its size proportional to the amount spent on newspaper ads. A linear trendline will also be included, helping to identify the correlation between these two variables. The plot allows for easy exploration of how newspaper advertising influences sales, and its interactive nature lets you hover over points for more detailed information.

**Code Snippet #5: Code Snippet of Scatter Plot with Trendline for Radio vs Sales**

figure = px.scatter(data\_frame = data, x="Sales",

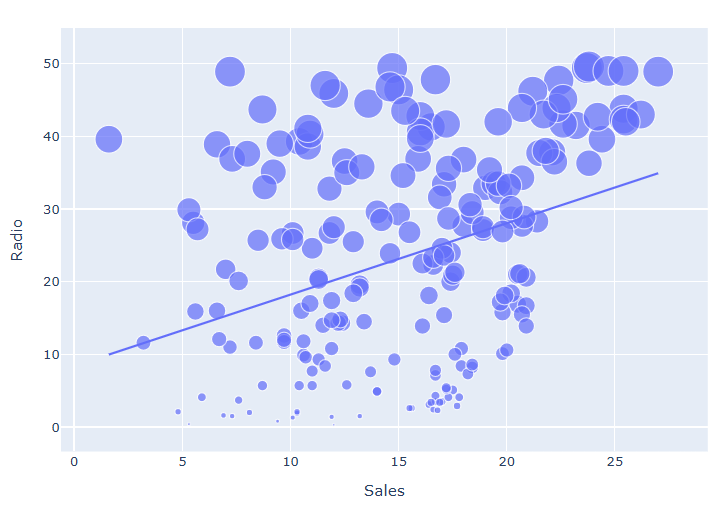
                    y="Radio", size="Radio", trendline="ols")

figure.show()

**Function:**

This code creates an interactive scatter plot using the plotly.express library. The plot maps the Sales column to the x-axis and the Radio column to the y-axis. The size of each point is proportional to the Radio values, which likely represent the amount spent on radio advertising. A linear trendline is added to the plot using the "ols" (Ordinary Least Squares) method, which fits a line to the data points to show the relationship between Sales and Radio. The plot is displayed using figure.show().

**Figure#5:**



**Description:**

The output is an interactive scatter plot showing how radio advertising spend (Radio) is related to product sales (Sales). The size of each point reflects the amount spent on radio ads. The trendline highlights the overall linear relationship between the two variables. This visualization helps identify trends, such as whether increases in radio advertising are associated with higher sales. The interactive nature of the plot allows you to hover over points for more detailed information, making it easier to explore the data.

**Code Snippet #6: Code Snippet of Correlation Calculation for Sales**

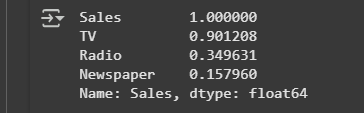
correlation = data.corr()

print(correlation["Sales"].sort\_values(ascending=False))

**Function:**

This code calculates the correlation matrix of the data DataFrame using the corr() function. The correlation matrix shows the relationships between all numerical columns in the dataset. Then, it selects the correlation values for the Sales column and sorts them in descending order using sort\_values(ascending=False). The result is printed, showing how each numerical feature correlates with Sales, from the highest to the lowest correlation.

**Figure#6:**

****

**Description:**

The output displays the correlation of the Sales column with every other numerical column in the dataset, sorted from the highest to the lowest correlation. Positive values indicate a positive correlation, meaning that as one variable increases, so does the other. Negative values indicate a negative correlation, where one variable decreases as the other increases. This helps you understand which factors, like TV, Newspaper, and Radio, have the strongest linear relationships with Sales, and can guide further analysis or model-building decisions.

**Future Sales Prediction Model**

**Code Snippet #7: Code Snippet of Preparing Data for Model Training**

x = np.array(data.drop(["Sales"], axis=1))

y = np.array(data["Sales"])

xtrain, xtest, ytrain, ytest = train\_test\_split(x, y,

                                                test\_size=0.2,

                                                random\_state=42)

model = LinearRegression()

model.fit(xtrain, ytrain)

print(model.score(xtest, ytest))

**Function:**

This code first separates the dataset into features (x) and the target variable (y). It removes the Sales column from the data DataFrame to create x and uses Sales as y. Then, the data is split into training and testing sets with 80% of the data used for training and 20% for testing using the train\_test\_split() function. After the split, a linear regression model is instantiated and trained on the training data (xtrain, ytrain) using the fit() method. Finally, the model's performance is evaluated by calculating the R² score on the test set (xtest, ytest) with the score() method. This score tells us how well the model's predictions match the actual values in the test set.

**Figure#7:**



**Description:**

The output, 0.9059011844150826, is the R² score of the trained linear regression model. This score indicates that approximately 90.59% of the variance in Sales can be explained by the features used in the model (such as TV, Radio, and Newspaper). In simpler terms, the model is quite good at predicting Sales based on the provided features. A value close to 1 means the model has a strong predictive power, and this score suggests that the model has a high degree of accuracy when predicting Sales on the test data.

**Code Snippet #8: Code Snippet of Making Predictions with the Trained Model**

#features = [[TV, Radio, Newspaper]]

features = np.array([[230.1, 37.8, 69.2]])

print(model.predict(features))

**Function:**

In this code, the trained linear regression model is used to predict the Sales value based on a specific set of input features: TV, Radio, and Newspaper advertising spends. The input features are provided as a 2D NumPy array (features), where each row represents a set of values for the input variables. The model.predict() method is then used to compute the predicted Sales value based on these input values, and the result is printed.

**Figure#8:**



**Description:**

The output of [21.37254028] represents the model’s predicted sales value for the given set of advertising spend values: $230.1 on TV, $37.8 on Radio, and $69.2 on Newspaper. This means that with these specific advertising spends, the model predicts approximately 21.37 units of sales. The prediction reflects the relationship the model has learned between advertising spend and sales, based on the historical data it was trained on.