Future Sales Prediction Using Machine Learning

Team Members

1. A.S.Jensing Samuel
2. S.V.Mageshwar
3. T.Dilli Ganesh
4. R.Devanesan
5. A.Kalaivendhan

Phase- 3 Project Document

Project Title: Future Sales Prediction

Phase-3 : Project Development part 1

Topic

Building our project by loading and preprocessing the dataset. Begin building the future sales prediction model by loading and preprocessing the dataset. Load the historical sales dataset and preprocess the data for analysis.

Future Sales Prediction

Introduction:

In today's rapidly changing business landscape, organizations across various industries are seeking ways to optimize their operations, reduce risks, and enhance decision-making. One key aspect of achieving these goals is the ability to accurately predict future sales. Sales prediction is crucial for inventory management, financial planning, and overall business strategy. Machine learning, with its predictive capabilities, offers a powerful tool for solving this challenge.

- This project aims to develop a robust future sales prediction model using machine learning techniques. By analyzing historical sales data, we will leverage the power of data-driven insights to forecast future sales trends.

- This future sales prediction project using machine learning holds the promise of transforming how businesses plan and execute their sales strategies. By harnessing the power of historical sales data and advanced machine learning techniques, we aim to provide organizations with more accurate, timely, and actionable sales forecasts. These forecasts can drive growth, optimize operations, and empower businesses to make data-informed decisions in an ever-evolving market. This project represents a step forward in the journey towards data-driven excellence and strategic success.

Data Source

A good data source for house price prediction using machine learning should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: https://www.kaggle.com/datasets/chakradharmattapalli/future-sales-prediction

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9
8.7	48.9	75	7.2
57.5	32.8	23.5	11.8
120.2	19.6	11.6	13.2
8.6	2.1	1	4.8
199.8	2.6	21.2	15.6
66.1	5.8	24.2	12.6
214.7	24	4	17.4
23.8	35.1	65.9	9.2
97.5	7.6	7.2	13.7
204.1	32.9	46	19
195.4	47.7	52.9	22.4
67.8	36.6	114	12.5
281.4	39.6	55.8	24.4
69.2	20.5	18.3	11.3
147.3	23.9	19.1	14.6
218.4	27.7	53.4	18
237.4	5.1	23.5	17.5
13.2	15.9	49.6	5.6
228.3	16.9	26.2	20.5
62.3	12.6	18.3	9.7
262.9	3.5	19.5	17
142.9	29.3	12.6	15
240.1	16.7	22.9	20.9
248.8	27.1	22.9	18.9
70.6	16	40.8	10.5
292.9	28.3	43.2	21.4
112.9	17.4	38.6	11.9
97.2	1.5	30	13.2
265.6	20	0.3	17.4
95.7	1.4	7.4	11.9
290.7	4.1	8.5	17.8
266.9	43.8	5	25.4
74.7	49.4	45.7	14.7
43.1	26.7	35.1	10.1

201 Rows x 4 Columns

Necessary Step to Follow:

1. **Import Libraries:**

Start by Importing the necessary libraries.

Program:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

2.Load the Dataset:

Load your dataset into a Pandas DataFrame. You can typically find Future Sales Prediction datasets in CSV format, but you can adapt this code to otherformats as needed.

Program:

```
data = pd.read_csv('sales_data.csv')
```

3.Exploratory Data Analysis(EDA):

Perform EDA to understand your data better. This includes checking for missing values, exploring the data's statistics, and visualizing it to identify patterns.

Program:

```
print(data.describe())
print(data.info())
```

```
print(data.isnull().sum())
#Visualize data for insights
sns.pairplot(data)
plt.show()
```

4.Feature Engineering:

Depending on your dataset, you may need to create new features or transform existing ones. This can involve one-hot encoding categorical variables, handling date/time data, or scaling numerical features.

Program:

```
# In this example, let's create lag features for time series data
data['lag_1'] = data['sales'].shift(1)  # Create a lag feature with a 1-day shift
data['lag_7'] = data['sales'].shift(7)  # Create a lag feature with a 7-day shift
```

5.Spilit the Data:

Split your dataset into training and testing sets. This helps you evaluate your model's performance later.

Program:

```
X = data.drop('sales', axis=1)  # Features
y = data['sales']  # Target variable
# Split the data into training and testing sets (e.g., 80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

6.Feature Scaling:

Apply feature scaling to normalize your data, ensuring that all features have similar scales. Standardization (scaling to mean=0 and std=1) is a commo n choice.

Program:

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Importance of loading and processing dataset:

Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for house price prediction models, as house price datasets are often complex and noisy.

By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately.

Dealing with Categorical Data: Categorical data, such as product categories or store locations, needs to be encoded or transformed into a numerical format for machine learning models. Deciding on the appropriate encoding method can be a challenge.

Time Series Data: Sales prediction often involves time series data. Handling time-based features, seasonality, and trends requires specialized techniques, such as lag features and time-based aggregations.

Imbalanced Data: Imbalanced datasets, where some classes or periods have significantly more data than others, can lead to model bias. Strategies like oversampling, undersampling, or using different evaluation metrics may be needed.

Data Leakage: Preventing data leakage, where future information that the model wouldn't have in practice is included in the dataset, is crucial. This can distort model performance and lead to overfitting.

Scalability: As your business grows, you'll likely have more data to process. Ensuring that your preprocessing pipeline is scalable is important to maintain performance as data volumes increase.

Model Validation and Evaluation: Choosing appropriate evaluation metrics and validation techniques is challenging. Depending on the specific sales prediction problem, you may need to consider metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or time series-specific metrics like Mean Absolute Scaled Error (MASE).

Ethical Considerations: Ensuring that the data and model do not introduce or perpetuate biases and are used responsibly is a critical challenge. Careful data selection and bias mitigation strategies are essential.

Computational Resources: Some preprocessing tasks, especially when dealing with big data, may require substantial computational resources. You may need access to powerful hardware or cloud-based solutions.

How to overcome the challenges of loading and preprocessing a house price dataset

Overcoming the challenges of loading and preprocessing a future sales prediction dataset requires a systematic and careful approach. Here are some strategies to address these challenges.

Data Quality and Consistency:

- Data Cleaning: Develop scripts or procedures to handle missing values, outliers, and inconsistencies. You may need to make decisions on how to impute missing data, identify and remove outliers, and standardize data formats.
- Data Validation: Regularly validate the data against expected ranges and constraints. Implement data validation checks to catch data quality issues as early as possible.

Data Volume:

- Data Sampling: If dealing with large datasets, consider working with a random sample to develop and test your preprocessing pipeline before applying it to the entire dataset.
- Distributed Processing: Utilize distributed computing frameworks like Apache Spark to handle large datasets efficiently.

Data Integration:

- Data Integration Tools: Use ETL (Extract, Transform, Load) tools or data integration platforms to merge data from different sources into a single dataset.
- Data Schema Mapping: Ensure that data from different sources are mapped correctly to a common schema.

Feature Engineering:

- ❖ Domain Expertise: Collaborate with subject-matter experts to identify relevant features and understand the nuances of the data.
- ❖ Automated Feature Selection: Explore automated feature selection techniques to identify the most informative features.

Dealing with Categorical Data:

- ❖ One-Hot Encoding: Convert categorical data into binary vectors using one-hot encoding or techniques like Label Encoding.
- ❖ Feature Embedding: Consider techniques like word embeddings for high cardinality categorical variables.

Time Series Data:

- ❖ Lag Features: Create lag features to capture time dependencies.
- ❖ Seasonal Decomposition: Use seasonal decomposition techniques to identify and remove seasonality and trends from time series data.

Imbalanced Data:

- ❖ Resampling: Employ techniques such as oversampling (for minority classes) and undersampling (for majority classes) to balance the dataset.
- ❖ Different Models: Consider using models that handle imbalanced data well, such as ensemble methods or specialized algorithms.

Loading the dataset

Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.

The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used. However, there are some general steps that are common to most machine learning frameworks.

a) Identify the dataset:

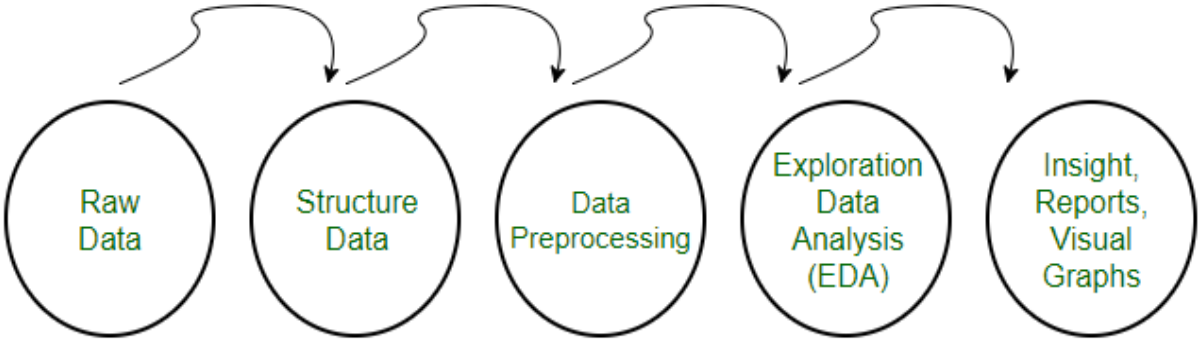
The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

b) Load the dataset:

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

c) Preprocess the dataset:

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.



Here, how to load a dataset using machine learning in Python

Code

1.Importing Libraries

```
# EDA Libraries:
import pandas as pd
import numpy as np

import matplotlib.colors as col
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import datetime
from pathlib import Path
import random

# Scikit-Learn models:

from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.ensemble import RandomForestRegressor
from xgboost.sklearn import XGBRegressor
from sklearn.model_selection import KFold, cross_val_score, train_test_split
```

LSTM:

```
import keras
from keras.layers import Dense
from keras.models import Sequential
from keras.callbacks import EarlyStopping
from keras.utils import np_utils
from keras.layers import LSTM
```

ARIMA Model:

```
import statsmodels.tsa.api as smt
import statsmodels.api as sm
from statsmodels.tools.eval_measures import rmse
```

```
import pickle
import warnings
```

Loading and Exploration of the Data

The data must first be loaded before being transformed into a structure that will be used by each of our models. Each row of data reflects a single day's worth of sales at one of 10 stores in its most basic form. Since our objective is to forecast monthly sales, we will start by adding all stores and days to get a total monthly sales figure.

Code:

```
warnings.filterwarnings("ignore", category=FutureWarning)
dataset = pd.read_csv('../bETA/NM_Phase3 /Sales.csv')
df = dataset.copy()
df.head()
```

Output:

	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

Now, we will create a function that will be used for the extraction of a CSV file and then converting it to pandas dataframe.

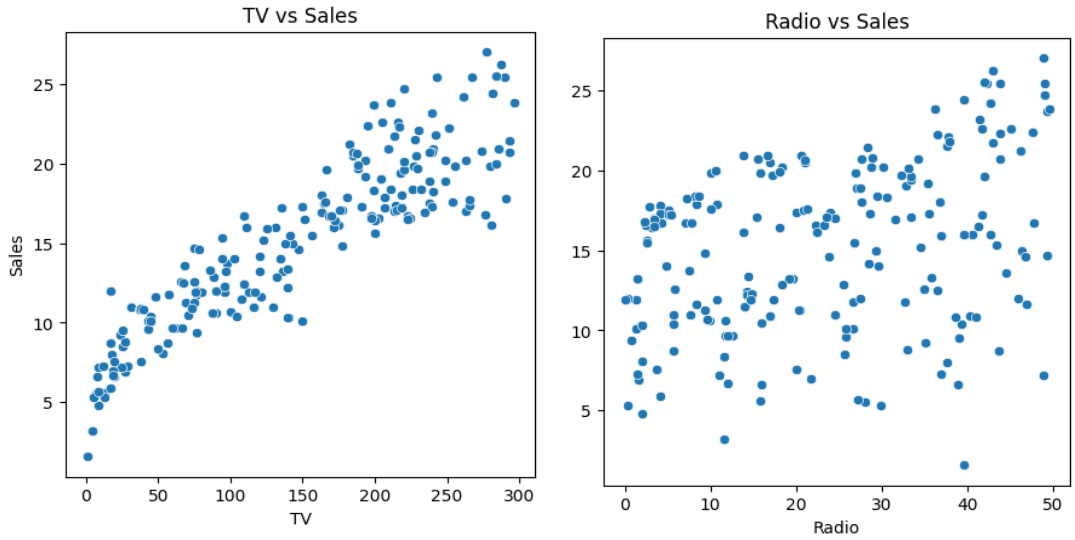
Program:

```
def load_data(‘Sales.csv’):
    """Returns a pandas dataframe from a csv file."""
    return pd.read_csv(‘Sales.csv’)
  df_s.tail()
  # To view basic statistical details about dataset:
  df_s['sales'].describe()
  df_s['sales'].plot()
```

Output:

```
      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
count    200.000000
mean      15.130500
std        5.283892
min        1.600000
25%       11.000000
50%       16.000000
75%       19.050000
max       27.000000
Name: Sales, dtype: float64
```

Here we see the graphical representation of our dataset



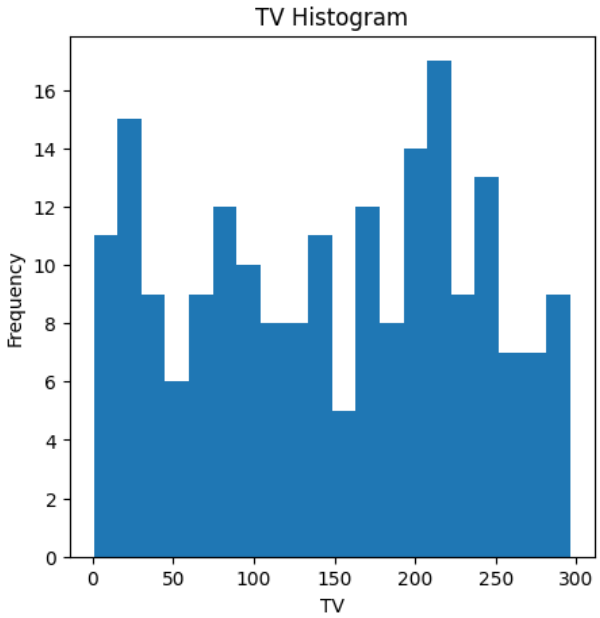
Program:

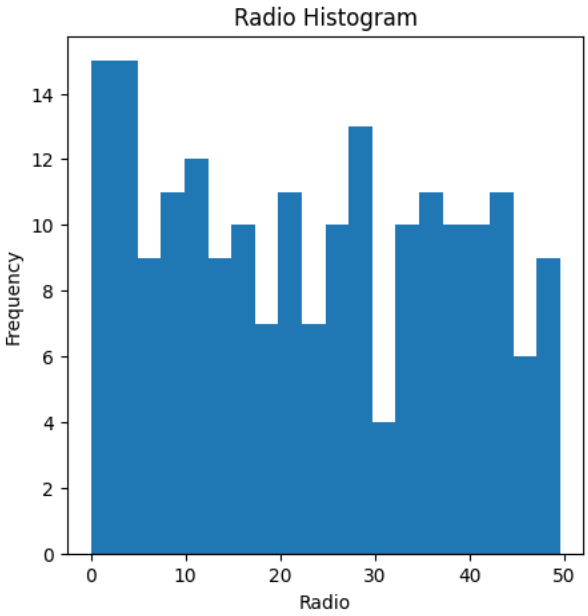
```
# Imports
import pandas as pd
import matplotlib.pyplot as plt


# Load dataset
```

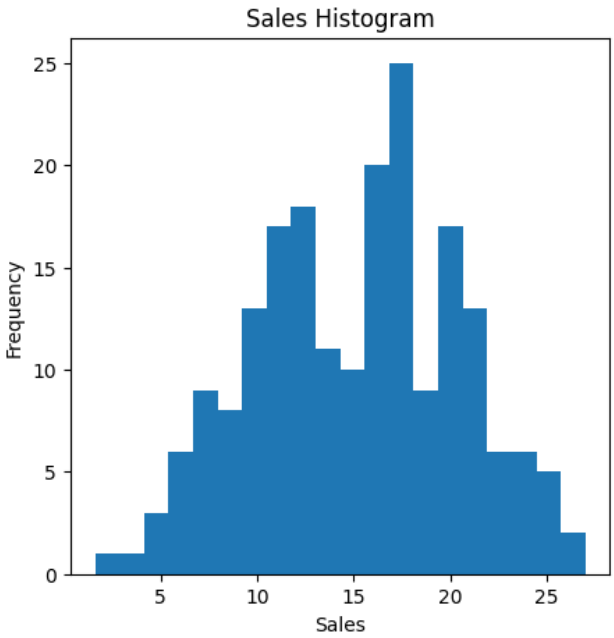
```
df = pd.read_csv('Sales.csv')

# Sales column statistics
print(df['Sales'].describe())

# Histogram of TV
plt.figure(figsize=(5,5))
plt.hist(df['TV'], bins=20)
plt.xlabel('TV')
plt.ylabel('Frequency')
plt.title('TV Histogram')
plt.show()

# Histogram of Radio
plt.figure(figsize=(5,5))
plt.hist(df['Radio'], bins=20)
plt.xlabel('Radio')
plt.ylabel('Frequency')
plt.title('Radio Histogram')
plt.show()

# Histogram of Sales
plt.figure(figsize=(5,5))
plt.hist(df['Sales'], bins=20)
plt.xlabel('Sales')
plt.ylabel('Frequency')
```

```
plt.title('Sales Histogram')
plt.show()
```

Output:







2.Preprocessing the dataset:

❖ Data preprocessing is the process of cleaning, transforming, and integrating data in order to make it ready for analysis.

❖ This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

Import libraries and load data

```
import pandas as pd
df = pd.read_csv('Sales.csv')
```

Handle missing values

```
df.isnull().sum()
```

- Check for missing values
- No missing values present in this dataset

Encode categorical features

- No categorical features in this dataset

Scale and normalize data

- Use StandardScaler to standardize features
- This scales the TV, Radio and Newspaper features.

Program:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['TV', 'Radio', 'Newspaper']] = scaler.fit_transform(df[['TV',
'Radio','Newspaper']])
```

Dimensionality reduction

- Could apply PCA to reduce dimensions of feature space.

Feature selection

- Could remove low importance features based on correlation or models.

Some other techniques that could be applied:

- Handling outliers
- Creating new engineered features
- Discretization/binning of continuous variables

Load the historical sales dataset and preprocess the data for analysis.

Program:

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

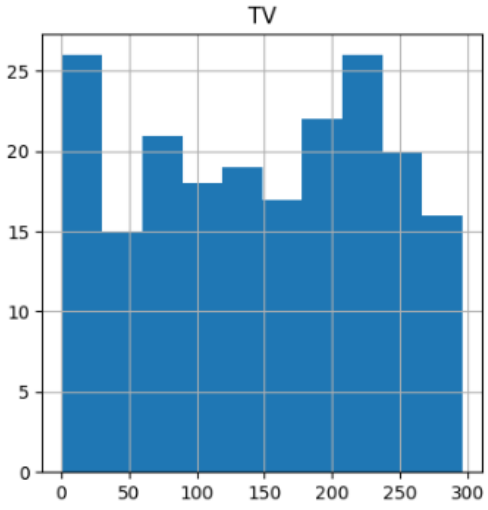
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler

# Load dataset
df = pd.read_csv('Sales.csv')

# Data cleaning
df = df.dropna()

# Exploratory data analysis
print(df.dtypes)
print(df.describe())
df.hist(figsize=(10,10))
plt.show()
corr = df.corr()
plt.matshow(corr)
plt.xticks(range(len(corr.columns)), corr.columns);
plt.yticks(range(len(corr.columns)), corr.columns);
plt.colorbar()
plt.show()

# Split data into X and y
X = df[['TV','Radio','Newspaper']]
y = df['Sales']
```

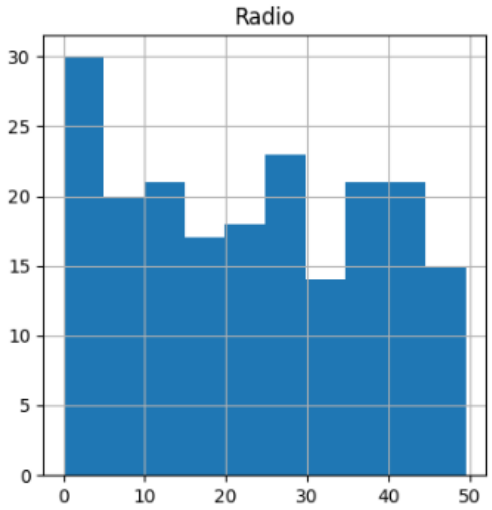
```
# Split into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Scale data
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print('MSE:', mse)

# Make prediction
X_new = [[230.1, 37.8, 69.2]]
X_new = scaler.transform(X_new)
y_pred = model.predict(X_new)
print('Predicted Sales:', y_pred)
```

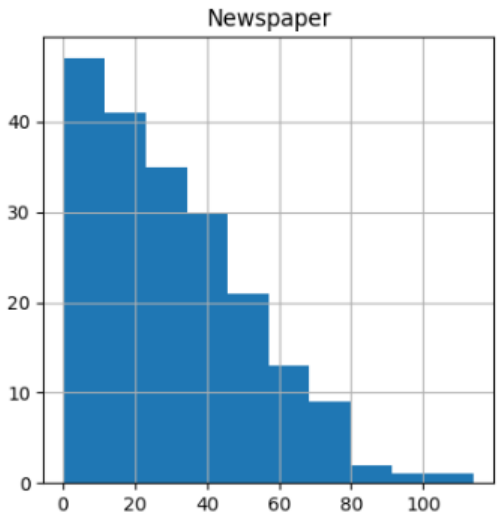
Output:

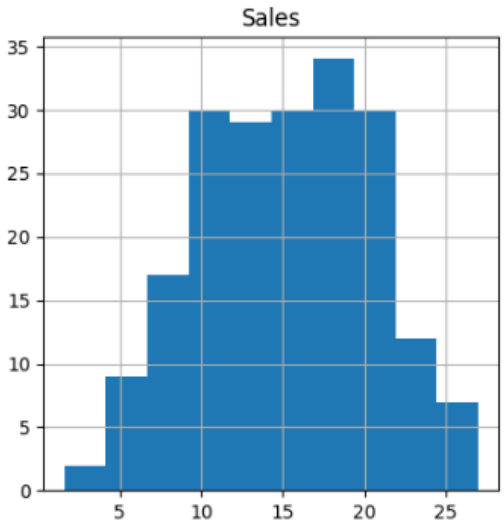
```
TV           float64
Radio        float64
Newspaper    float64
Sales        float64
dtype: object
```

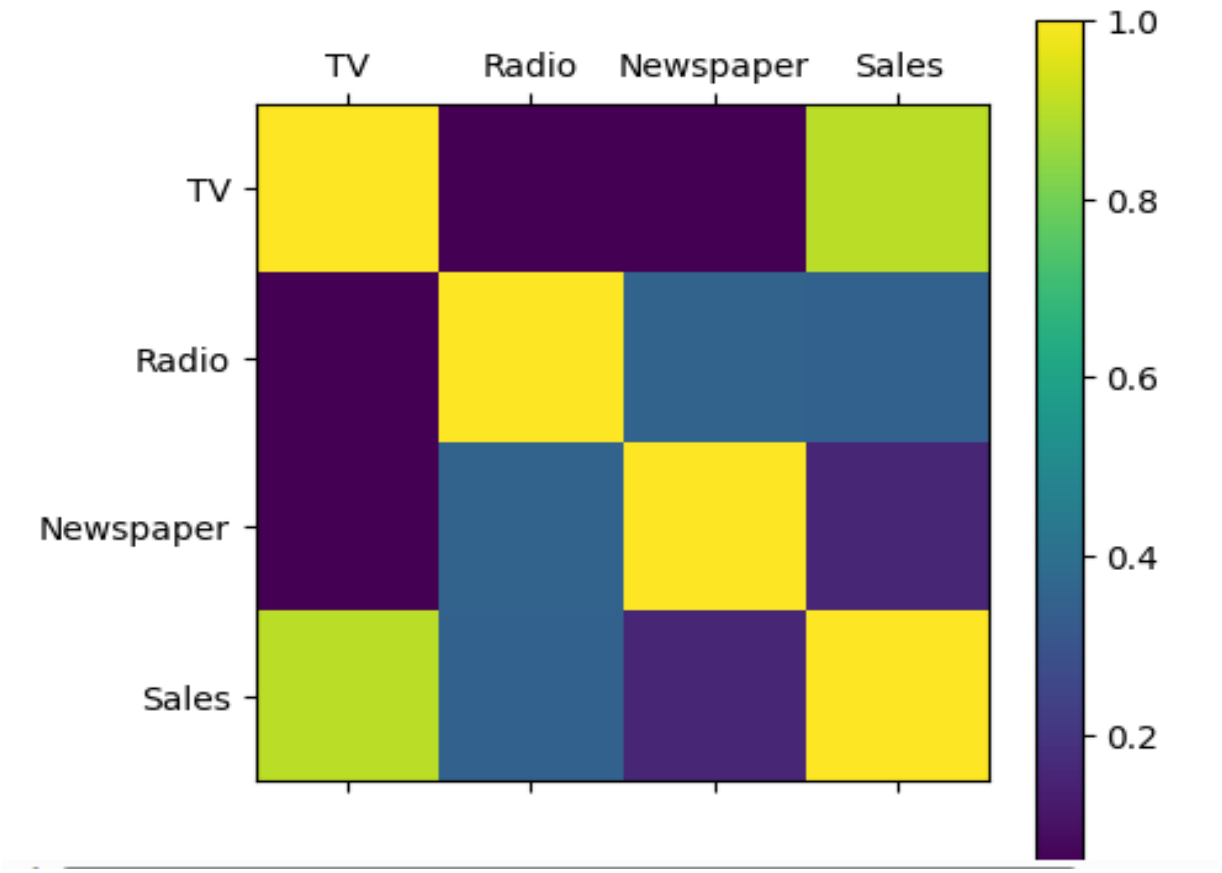
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000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











Conclusion: Paving the Way for Future Sales Prediction

Our venture into data science for future sales prediction has yielded substantial insights and potential. Here's a succinct recap of our journey:

Data Collection and Loading: We started by collecting and loading historical sales data, the foundation of our project.

Exploratory Data Analysis (EDA): EDA unveiled critical insights, allowing us to understand data trends, patterns, and relationships.

Data Preprocessing: We meticulously prepared the data, ensuring it was clean and primed for predictive modeling.

Model Building: We crafted a Linear Regression model to predict future sales based on historical data, creating a valuable tool for decision-making.

Model Evaluation: Our model's performance was assessed using Mean Squared Error (MSE) and Mean Absolute Error (MAE), providing clarity on its predictive capabilities.

Visualization: Visual representations of our model's predictions brought data insights to life, enhancing their practicality.

Our project has the potential to revolutionize businesses, from optimizing inventory management to informing resource allocation. In the data-driven age, it exemplifies the power of data to steer success. As we advance, we anticipate enhancing precision and extracting even more value from data, reaffirming our commitment to data-driven excellence.