APPLIED DATA SCIENCE GROUP-2

FUTURE SALES PREDICTION (PHASE - 5)

DEVELOPMENT & SUBMISSION

PROBLEM DEFINITION:

The problem is to develop a predictive model that uses historical sales data to forecast future sales for a retail company. The objective is to create a tool that enables the company to optimize inventory management and make informed business decisions based on data driven sales predictions. This project involves data preprocessing, feature engineering, model selection, training, and evaluation.

DESIGN THINKING

Design thinking is a creative problem-solving approach that can be applied to various domains, including sales prediction. To design a system for future sales prediction using design thinking, follow these steps:

1. Empathize:

Understand the problem by gathering insights from various stakeholders, such as sales teams, customers, and data analysts. Identify their needs, pain points, and expectations regarding sales prediction.

2. Define:

Clearly define the problem statement and goals.

3. Ideate:

Brainstorm potential solutions and concepts for sales prediction. Encourage diverse perspectives from cross-functional teams and leverage various sources of data, including historical sales data, market trends, customer behavior, and external factors.

4. Prototype:

Create a prototype of your sales prediction model. This could be a preliminary version of the algorithm, a visualization tool, or a dashboard.

5. Test:

Pilot your prototype by applying it to a subset of historical data and evaluate its accuracy. Collect feedback from end-users, such as sales managers and analysts. Refine your prototype based on the feedback.

<u>PHASES OF DEVELOPMENT</u>

There are 5 phases of development in Future Sales Prediction:

Phase 1 – Problem Definition and Design Thinking

- Introduction
- Problem Statement
- Data Collection
- Data Exploration and Feature Engineering
- Model Evaluation
- Deployment
- Monitoring and Maintenance
- Result

Phase 2 – Innovation

- Introduction
- Dataset details
- Libraries
- How to train and test
- Process flow

Phase 3 – Development part -1

- Loading the dataset
- Preprocessing the dataset

Phase 4 – Development part -2

- Model training
- Feature engineering
- Evaluation

Phase 5 – Project Documentation & Submission

- Documentation
- Submission

DATASET:

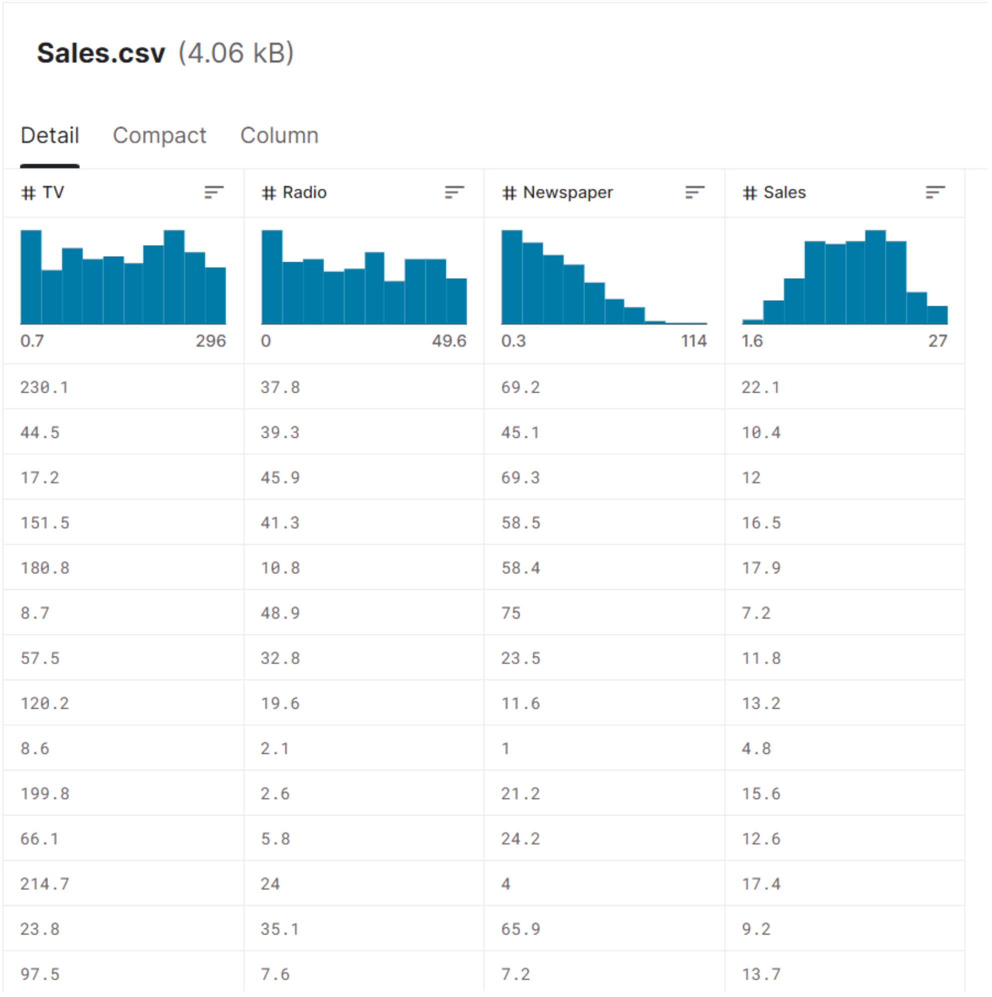
Dataset: www.kaggle.com

Dataset name: future-sales-prediction

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

DESCRIPTION ABOUT THE DATASET:

It consist of 4 columns and every column consist of 200 data's



# TV	# Radio	# Newspaper	# Sales
0.7 – 296	0 – 49.6	0.3 – 114	1.6 – 27
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

TV: this feature represents the amount of advertising budget spent on television media for a product or service in a certain period, for example in thousands of dollars (USD).

Radio: this feature represents the amount of advertising budget spent on radio media in the same period as TV.

Newspaper: this feature represents the amount of advertising budget spent in newspapers or print media in the same period as TV and Radio.

Sales: This feature represents product or service sales data in the same period as advertising expenditure on TV, Radio and Newspaper.

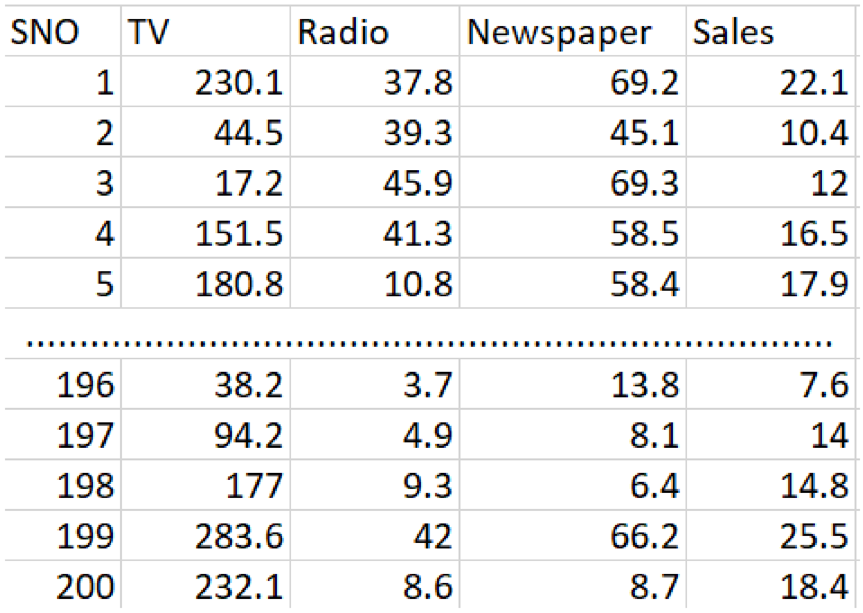
DATA PREPROCESSING STEPS:

1. Acquire the dataset
2. Import all the crucial libraries
3. Import the dataset
4. Identifying and handling the missing values
5. Encoding the categorical data
6. Splitting the dataset
7. Feature scaling

1.Acquire the dataset:

Dataset link:

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



SNO	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	12
4	151.5	41.3	58.5	16.5
5	180.8	10.8	58.4	17.9
....				
196	38.2	3.7	13.8	7.6
197	94.2	4.9	8.1	14
198	177	9.3	6.4	14.8
199	283.6	42	66.2	25.5
200	232.1	8.6	8.7	18.4

2.Import all the crucial libraries

Numpy

```
import numpy as np
```

Pandas

```
import pandas as pd
```

Scikit-learn

```
from sklearn.metrics import mean_squared_error, r2_score

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import StandardScaler
```

Matplotlib

```
import matplotlib.pyplot as plt
```

Seaborn

```
import seaborn as sns
```

Plotly

```
import plotly.express as px
```

3.Import the dataset

We are using pandas library as It provides data structures like DataFrames for working with structured data, such as tabular data in CSV or Excel files. Copy the dataset link from the file and paste it in this “pd.read_csv” area

```
import pandas as pd
# Import dataset
data = pd.read_csv("F:\AIML DATASET\Sales.csv")
```

4.Identifying and handling the missing values

Handling the missing values:

Handling missing values is an important step in data preprocessing to ensure that your data is clean and suitable for analysis or modeling.

```
# Check for missing values
print(data.isnull().sum())
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

5.Encoding the categorical data:

In our dataset, there is no explicit encoding of categorical data because the dataset appears to consist of numerical features. Encoding categorical data is typically necessary when dealing with non-numeric data, such as text or categorical labels .Since the code is working with numerical features, we don't need to perform categorical encoding.

6.Splitting the dataset:

```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
```

Once we have pre-processed the data into a format that's ready to be used by the model, we need to split up the data into train and test sets. This is because the machine learning algorithm will use the data in the training set to learn what it needs to know.

It will then make a prediction about the data in the test set, using what it has learned. We can then compare this prediction against the actual target variables in the test set in order to see how accurate the model is.

We will do the train/test split in proportions. The larger portion of the data split will be the train set and the smaller portion will be the test set. This will help to ensure that we are using enough data to accurately train the model.In general, we carry out the train-test split with an 80:20 ratio(also 70:30 ratio)

7.Feature scaling

There are many feature engineering technique but we use “scaling” in our project.

Scaling is a technique used to transform numerical variables to have a similar scale, so that they can be compared more easily. The most common scaling techniques are standardization and normalization. Standardization scales the variable so that it has zero mean and unit variance.

"Feature Scaling" is performed using the `StandardScaler` from scikit-learn. Feature scaling is a preprocessing step in machine learning that standardizes or normalizes the features (independent variables) to ensure that they have the same scale. This is important because many machine learning algorithms are sensitive to the scale of input features.

```
from sklearn.preprocessing import StandardScaler
# Create a StandardScaler instance
scaler = StandardScaler()
```

MODEL TRAINING

Model training is the primary step in machine learning, resulting in a working model that can then be validated, tested, and deployed. The model's performance during training will eventually determine how well it will work when it is eventually put into an application for the end-users.

Step 1: Data preprocessing

Load the dataset and check for any missing values

```
data = pd.read_csv("F:\AIML DATASET\Sales.csv")
print(data.isnull().sum())
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

Step 2: Data splitting

The first step in model training is to prepare our data. Split the dataset into training and test sets using the train_test_split function from the scikit-learn library. This ensures that we have separate data for training and evaluating our model.

```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
```

Step 3: Train the Model

Next, select an appropriate model for the task and train it using the training set. Let us train a logistic regression model using scikit-learn:

```
from sklearn.linear_model import LinearRegression
# Create a instance Linear Regression model
model = LinearRegression()
# Fit the model on the scaled training data
model.fit(xtrain_scaled, ytrain)
```

Step 4: Evaluate on the Test Set

Now, it's time to evaluate our model on the test set. Use the trained model to make predictions on the test data and compare them to the actual labels.

```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
# Make predictions on the scaled test data
ypred_scaled = model.predict(xtest_scaled)
# Evaluate the model's performance on scaled data
mse_scaled = mean_squared_error(ytest, ypred_scaled)
r2_scaled = r2_score(ytest, ypred_scaled)
```

Step 5: Perform Regression metrics

Metrics refer to quantitative measures used to assess the performance, quality, or characteristics of a model. Metrics are essential for making data-driven decisions, comparing different models and determining whether the data science project is successful.

Mean Squared Error (MSE): The average of the squared differences between predicted and actual values.

```
mse_scaled = mean_squared_error(ytest, ypred_scaled)
```

Root Mean Squared Error (RMSE): The square root of the MSE, which provides a measure of the error in the original units.

```
r2_scaled = r2_score(ytest, ypred_scaled)
```

Step 6: Assess Model's Performance

Analyze the evaluation metrics obtained from the previous steps to assess the model's performance. Consider the context of the problem and compare the results against the desired performance level or any baseline models. This analysis will provide insights into the strengths and weaknesses of the model.

Step 7: Iterate and Improve (if needed)

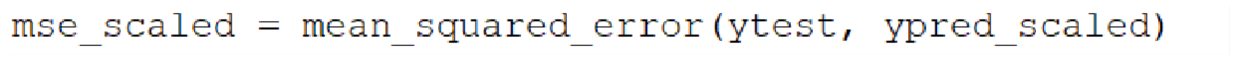
Based on the assessment, we may need to iterate and improve the model. Consider collecting more data, refining features, trying different algorithms, or tuning hyperparameters. Repeat the evaluation process until we achieve the desired performance.

EVALUATION:

Evaluation in data science refers to the process of assessing the performance, accuracy, and effectiveness of a predictive model, algorithm, or analytical approach applied to a dataset. It is a critical step in the data science workflow that helps determine how well the model or analysis meets the goals and objectives of a given project.

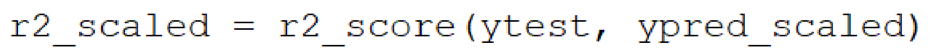
MSE AND R-Squared value:

Mean Squared Error (MSE): The average of the squared differences between predicted and actual values.

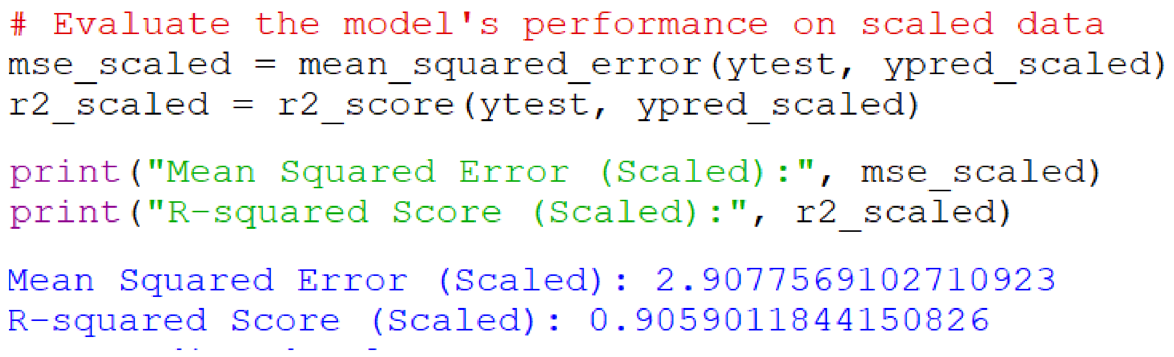


```
mse_scaled = mean_squared_error(ytest, ypred_scaled)
```

Root Mean Squared Error (RMSE): The square root of the MSE, which provides a measure of the error in the original units.



```
r2_scaled = r2_score(ytest, ypred_scaled)
```



```
# Evaluate the model's performance on scaled data
mse_scaled = mean_squared_error(ytest, ypred_scaled)
r2_scaled = r2_score(ytest, ypred_scaled)

print("Mean Squared Error (Scaled):", mse_scaled)
print("R-squared Score (Scaled):", r2_scaled)
```

```
Mean Squared Error (Scaled): 2.9077569102710923
R-squared Score (Scaled): 0.9059011844150826
```

DATA VIRTUALIZATION:

Data visualization is the graphical representation of information and data in a pictorial or graphical format(Example: charts, graphs, and maps). Data visualization tools provide an accessible way to see and understand trends, patterns in data, and outliers. Data visualization tools and technologies are essential to analyzing massive amounts of information and making data-driven decisions. The concept of using pictures is to understand data that has been used for centuries. General types of data visualization are Charts, Tables, Graphs, Maps, Dashboards.

Plotly:

The Plotly library is an interactive open-source library. This can be a very helpful tool for data visualization and understanding the data simply and easily. plotly graph objects are a high-level interface to plotly which are easy to use. It can plot various types of graphs and charts like scatter plots, line charts, bar charts, box plots, histograms, pie charts, etc.

```
import plotly.express as px
```

Representing the TV,Radio,Nwespaper using plotly:

```
# Plot representation of TV
figure = px.scatter(data_frame=data, x="TV", y="Sales", size="TV", trendline="ols")
figure.show()
```

```
# Plot representation of Newspaper
figure = px.scatter(data_frame=data, x="Newspaper", y="Sales", size="Newspaper", trendline="ols")
figure.show()
```

```
# Plot representation of Radio
figure = px.scatter(data_frame=data, x="Radio", y="Sales", size="Radio", trendline="ols")
figure.show()
```

Matplotlib:

Matplotlib is a powerful tool for executing a variety of tasks. It is able to create different types of visualization reports like line plots, scatter plots, histograms, bar charts, pie charts, box plots, and many more different plots. This library also supports 3-dimensional plotting.

```
import matplotlib.pyplot as plt
```

```
plt.bar(index, ytest, bar_width, label='Actual Sales', color='orange', align='center')
plt.bar(index + bar_width, ypred_scaled, bar_width, label='Predicted Sales', color='b', align='center')
plt.xlabel("Data Points")
plt.ylabel("Sales")
plt.title("Actual vs. Predicted Sales")
plt.legend()
plt.xticks(index + bar_width / 2, index)  # Set x-axis labels
plt.show()
```

Python code for predicting future sales:

```
import numpy as np

import pandas as pd

from sklearn.metrics import mean_squared_error, r2_score

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

import plotly.express as px

from sklearn.preprocessing import StandardScaler

# Import dataset

data =  pd.read_csv("F:\AIML  DATASET\Sales.csv")

# Display data

print(data)

# Shape of the dataset (rows, columns)

print(data.shape)

# Information about the dataset

print(data.info())

# Description of the dataset

print(data.describe())

# Check for missing values

print(data.isnull().sum())

# Plot representation of TV

figure = px.scatter(data_frame=data, x="TV", y="Sales", size="TV",
trendline="ols")
```

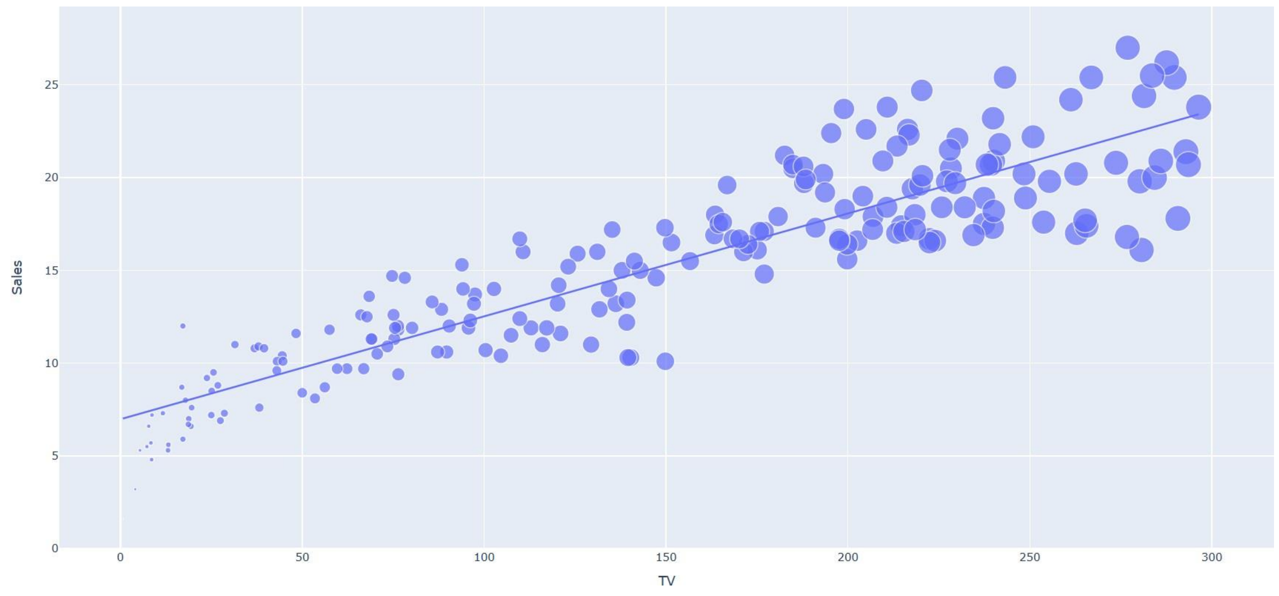
```
figure.show()

# Plot representation of Newspaper

figure = px.scatter(data_frame=data, x="Newspaper", y="Sales",
size="Newspaper", trendline="ols")

figure.show()

# Plot representation of Radio

figure = px.scatter(data_frame=data, x="Radio", y="Sales", size="Radio",
trendline="ols")

figure.show()

# Heatmap representation

sns.heatmap(data.corr(), cmap="YlGnBu", annot=True)

plt.show()

data.hist(bins=20)

# Split the data into features (x) and target (y)

x = data.drop("Sales", axis=1)  # Drop the "Sales" column

y = data["Sales"]

# Split the data into training and testing sets

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)

# Create a StandardScaler instance

scaler = StandardScaler()

# Fit the scaler on the training data and transform it

xtrain_scaled = scaler.fit_transform(xtrain)

# Transform the test data using the same scaler

xtest_scaled = scaler.transform(xtest)

# Create a Linear Regression model
```

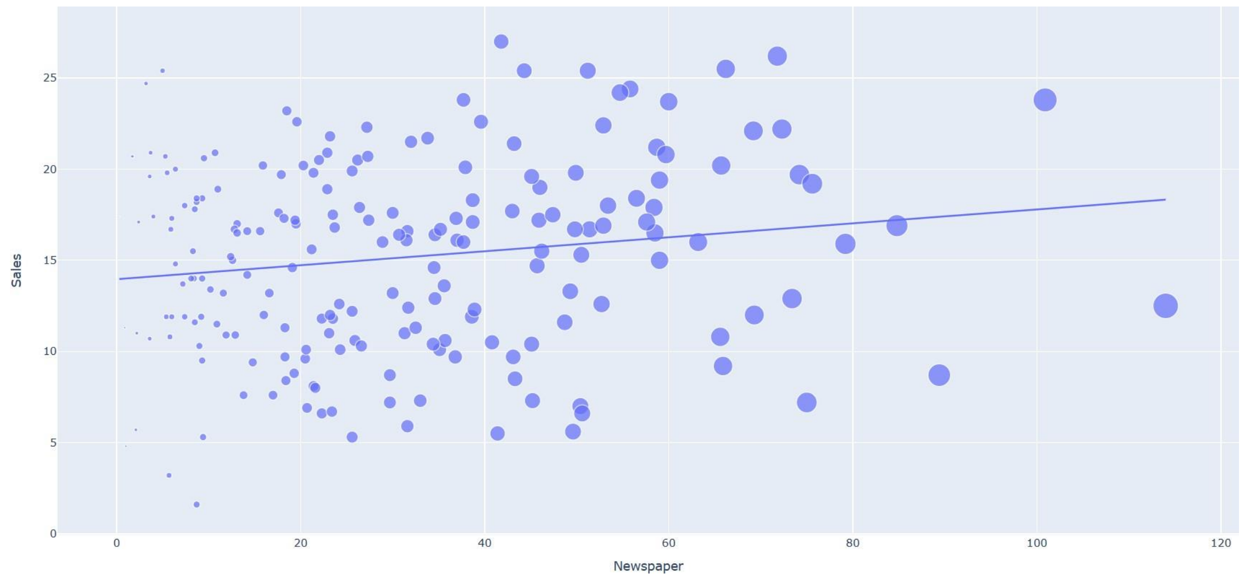
```
model = LinearRegression()
# Fit the model on the scaled training data
model.fit(xtrain_scaled, ytrain)
# Make predictions on the scaled test data
ypred_scaled = model.predict(xtest_scaled)
# Evaluate the model's performance on scaled data
mse_scaled = mean_squared_error(ytest, ypred_scaled)
r2_scaled = r2_score(ytest, ypred_scaled)
print("Mean Squared Error (Scaled):", mse_scaled)
print("R-squared Score (Scaled):", r2_scaled)
result = pd.DataFrame(data={"Predicted Sales": ypred_scaled})
print(result)
# Create a line graph for actual and predicted Sales
plt.figure(figsize=(10, 50))  # Set the figure size
plt.plot(ytest.values, label='Actual Sales', color='orange', marker='o')
plt.plot(ypred_scaled, label='Predicted Sales', color='b', marker='o')
plt.xlabel("Data Points")
plt.ylabel("Sales")
plt.title("Actual vs. Predicted Sales")
plt.legend()
plt.show()
```

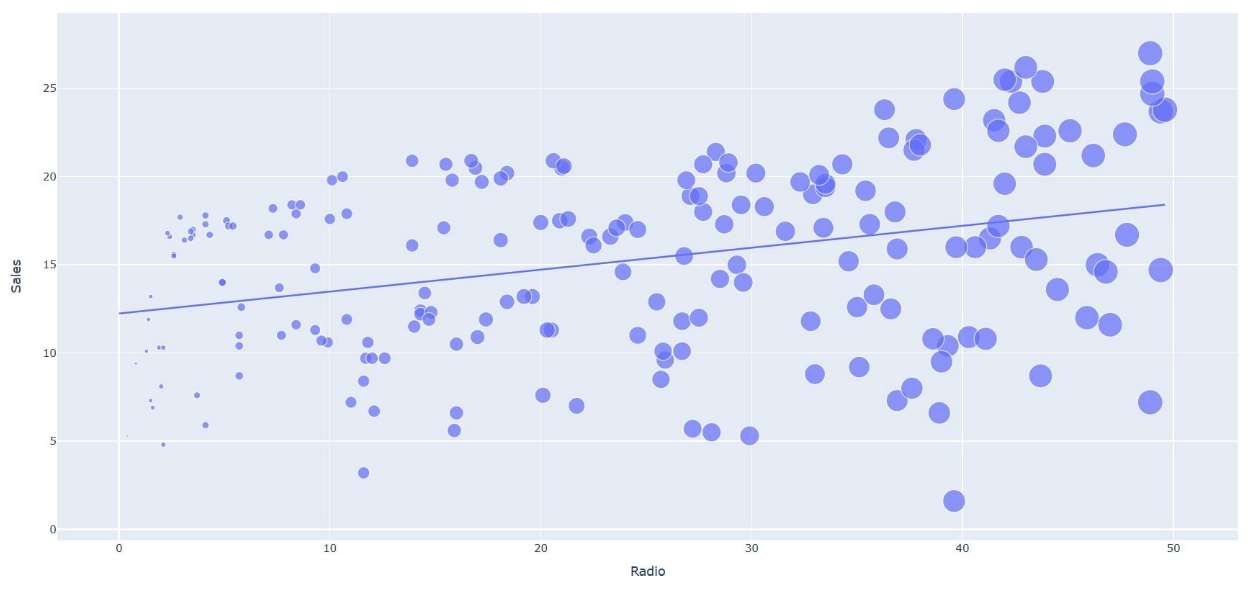
Output:

```
        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
..     ...    ...        ...    ...
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 x 4 columns]

(200, 4)

<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
None
               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
```

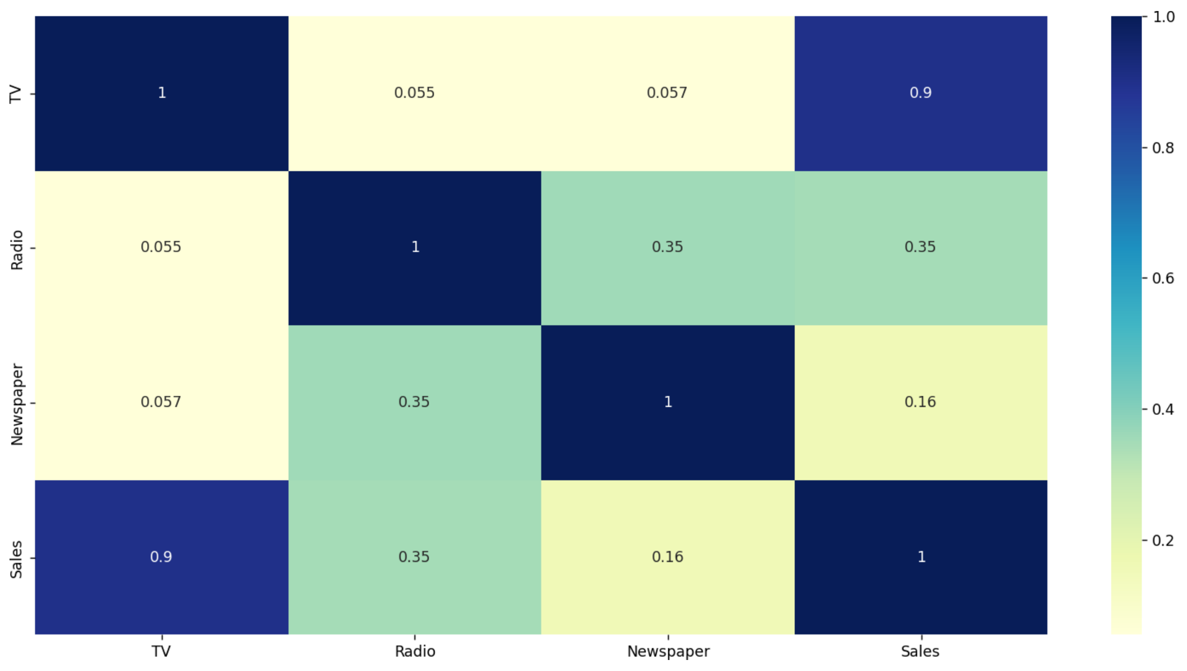
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```



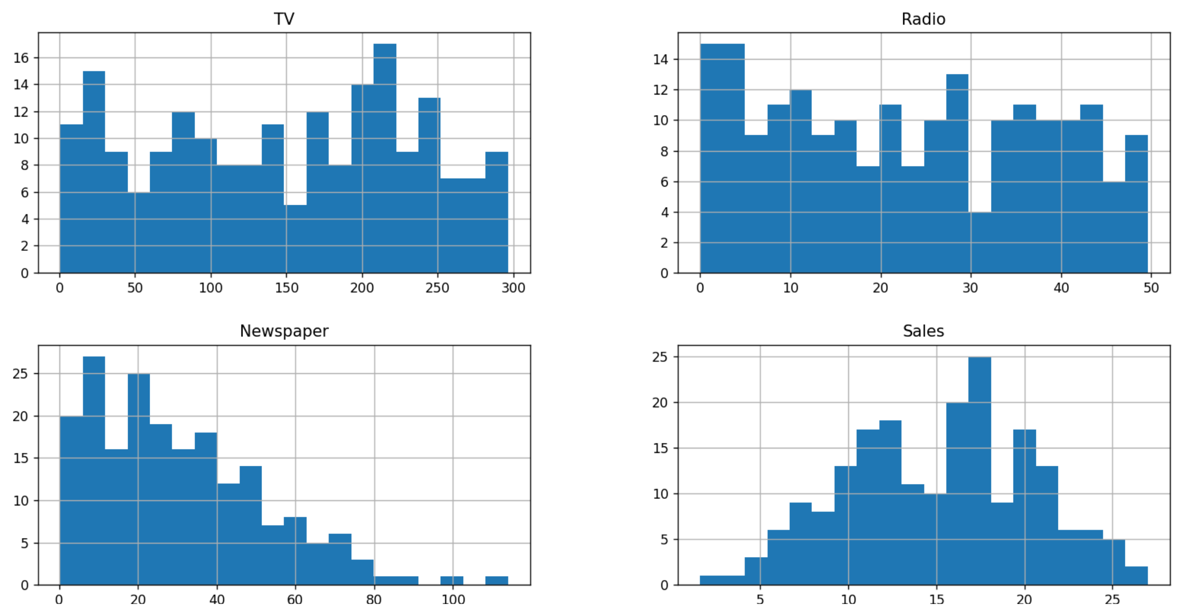




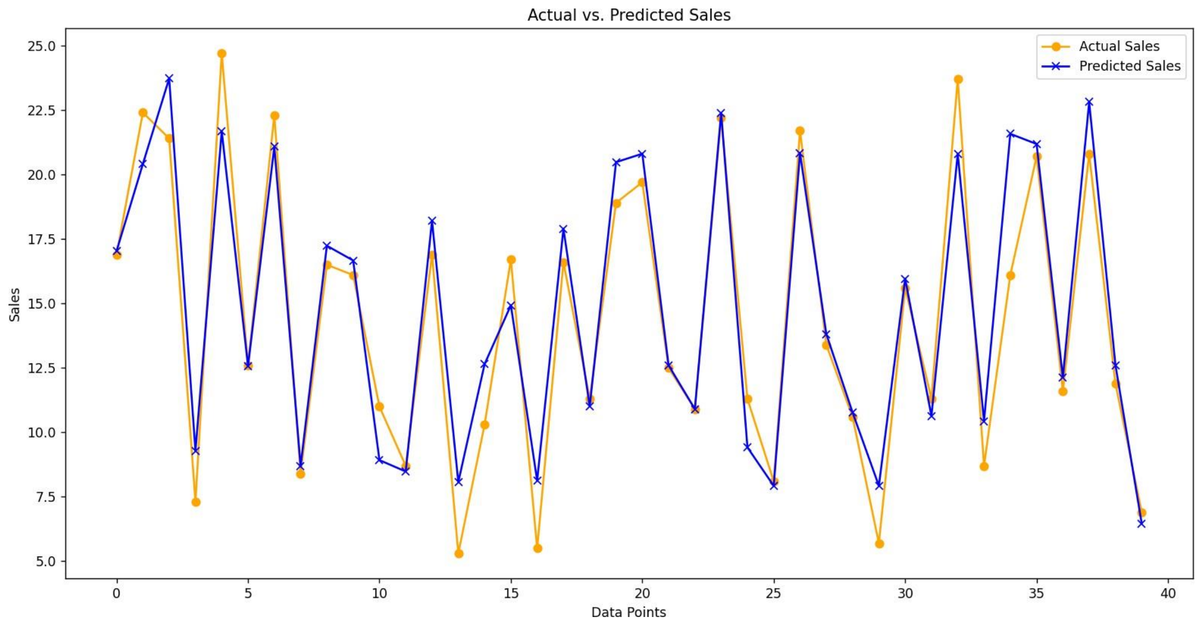
Sales
Radio
0
5
10
15
20
25
30
40
50



TV
Radio
Newspaper
Sales
1
0.055
0.057
0.9
0.35
0.16
1.0
0.8
0.6
0.4
0.2



```
Mean Squared Error (Scaled): 2.9077569102710923
R-squared Score (Scaled): 0.9059011844150826
```



```
    Predicted Sales
0         17.034772
1         20.409740
2         23.723989
3          9.272785
4         21.682719
5         12.569402
6         21.081195
7          8.690350
8         17.237013
9         16.666575
10         8.923965
11         8.481734
12        18.207512
13         8.067507
14        12.645510
15        14.931628
16         8.128146
17        17.898766
18        11.008806
19        20.478328
20        20.806318
21        12.598833
22        10.905183
23        22.388548
24         9.417961
25         7.925067
26        20.839085
27        13.815209
28        10.770809
29         7.926825
30        15.959474
31        10.634909
32        20.802920
33        10.434342
34        21.578475
35        21.183645
36        12.128218
37        22.809533
38        12.609928
39         6.464413
```

CONCLUSION:

So this is how we can train a machine learning model to predict the future sales of a product. Predicting the future sales of a product helps a business manage the manufacturing and advertising cost of the product. Sales forecasting is done by analyzing customer purchasing behaviour and it plays an important role in modern business intelligence. Forecasting future sales demand is key to business and business planning activities. Forecasting helps business organizations to make improvements, to make changes to business plans and to provide a stock storage solution. Forecast is determined by the use of data or information from past works and the consideration of recognized feature in future. Sales forecasting plays a vital role in strategic planning and market strategy for every company to assess past and present sales statistics and predict potential results.