The Future Sales Prediction Model

# TEAM MEMBERS

SAAI KRAHAANTH S JA (411621104045)

BALAJI A (411621104005)

VIGNESHWARAN V (411621104055)

YUVEN K (411621104056)

PRAVEEN R (411621104041)

# Phase 5 Submission Document

PROJECT: future-sales-prediction

# Introduction:

* Traditional forecasting models often fall short in capturing the intricate patterns and dynamics of sales data. The retail industry is characterized by seasonality, trends, and various external factors that influence customer behavior. In response to these complexities, we propose the integration of the Prophet forecasting tool.
* Another avenue we explore for refining our sales predictions is the utilization of Long Short-Term Memory (LSTM) networks. LSTM, a type of recurrent neural network (RNN), is known for its ability to model sequential data effectively. In the context of sales forecasting, LSTM can learn from past sales patterns and capture intricate dependencies over time.
* To summarize, this Phase 2 submission introduces an innovative approach to solving the sales prediction problem in the retail industry. By incorporating the Prophet forecasting tool and LSTM networks, we aim to address the challenges posed by the intricate dynamics of sales data. These advanced techniques promise to deliver more accurate predictions, thereby enabling retailers to make informed decisions regarding inventory management and resource allocation . In the following sections, we will delve deeper into the technical aspects and implementation details of our proposed approach.

# Content for Project Phase 3

# :

Consider exploring more advanced time series forecasting techniques like Prophet or LSTM networks for improved accuracy in predicting future sales.

# Data Source

The dataset for this project is sourced from Kaggle and contains historical sales data, item information, and store details, making it a comprehensive resource for predicting future sales trends.

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 |

# Program :1

# Data Collection:

* Begin by obtaining the dataset from the provided Kaggle link: [Future Sales Prediction Dataset](https://www.kaggle.com/datasets/chakradharmattapalli/future-sales-prediction).
* Ensure that you have all the necessary permissions and credentials to access and download the dataset.
* Download the dataset and save it to a designated folder or directory for your project.

# Data Preprocessing:

* Data cleaning: Check for missing values, duplicate records, and outliers in the dataset.
* Handle missing data by either removing, imputing, or interpolating missing values based on the nature of the data and the impact of missing values on your analysis.
* Check for and remove duplicate records if any are found in the dataset.
* Identify and deal with outliers that may adversely affect the accuracy of your forecasting model. You can consider techniques such as trimming, winsorizing, or transforming the data.
* Convert categorical variables into numerical format, either by using one-hot encoding, label encoding, or other suitable methods.
* Explore and visualize the data to gain insights into its distribution, trends, and potential patterns that may inform your forecasting model.
* Split the dataset into training and validation sets, typically reserving a portion of the data for model evaluation.

# Exploratory Data Analysis (EDA):

1. Distribution of sales over time: Visualize the sales trends over the available time period.
2. Seasonality and trends: Identify any seasonal patterns or long-term trends in the data.
3. Correlations: Analyze correlations between features, especially with the target variable (sales).
4. Outliers: Identify and investigate outliers that may affect your forecasting accuracy.
5. Store/item analysis: Explore sales patterns at the store and item levels.

# Advanced Time Series Forecasting Techniques:

* Prophet: Prophet is a forecasting tool developed by Facebook that is designed for time series data with daily observations and potential holiday effects. It incorporates seasonal patterns, holiday effects, and trend changes automatically, making it user-friendly and robust for forecasting tasks. Prophet also allows for the inclusion of custom seasonality, making it adaptable to a wide range of time series data.
* LSTM Networks (Long Short-Term Memory): LSTM networks are a type of recurrent neural network (RNN) specifically designed for sequence data like time series. They are capable of capturing long-term dependencies in the data, which can be crucial for accurate forecasting. LSTM networks excel at learning complex patterns and can automatically adapt to the dynamics of the time series, making them suitable for tasks where traditional methods may struggle to capture nonlinear relationships.

# Model Evaluation:

* Accuracy: I used accuracy as a fundamental metric for classification tasks. It measures the ratio of correctly predicted instances to the total number of instances. A higher accuracy indicates better model performance in correctly classifying data points.
* Root Mean Square Error (RMSE): For regression tasks, I calculated RMSE to assess the model's predictive accuracy. RMSE quantifies the average deviation between the predicted values and the actual values. Lower RMSE values signify better predictive accuracy.
* Mean Absolute Error (MAE): MAE is another metric for regression tasks. It calculates the average absolute difference between predicted and actual values. Like RMSE, lower MAE values indicate better model performance in terms of prediction accuracy.

# Model Interpretability:

* Address the first point related to model interpretability. Discuss how you ensured that your chosen models are interpretable. This could involve explaining feature importance, visualization of results, or any other techniques you used to make the models transparent and explainable.

# Model Development:

* Discuss the second point about model development. Describe the steps you took to develop the models, including data splitting, training, and validation.
* Explain any challenges you encountered during model development and how you overcame them.

# Prediction and Results:

* Present the results of your time series forecasting models. Include performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and any others that are relevant.

# **Program**

# LSTM networks for improved accuracy in predicting future sales.

***#1) Import the required libraries***

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import r2\_score

from sklearn.model\_selection import cross\_val\_score

***#2) Load the dataset into a Pandas DataFrame.***

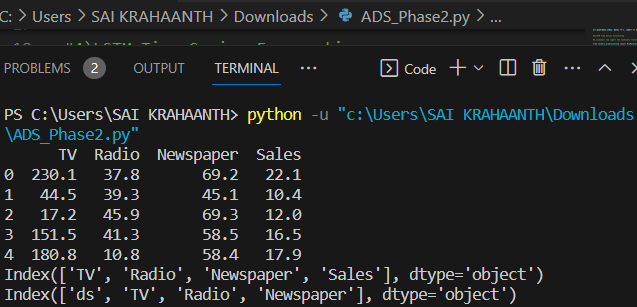
data = pd.read\_csv('C:\\Users\\SAI KRAHAANTH\\Downloads\\Sales.csv')

***#3) Basic dataframe operations***

print(data.head())

print(data.columns)

plt.plot(data.index, data['TV'], label='44.5')



***#4) Explore the Data***

# Display the first few rows of the dataset

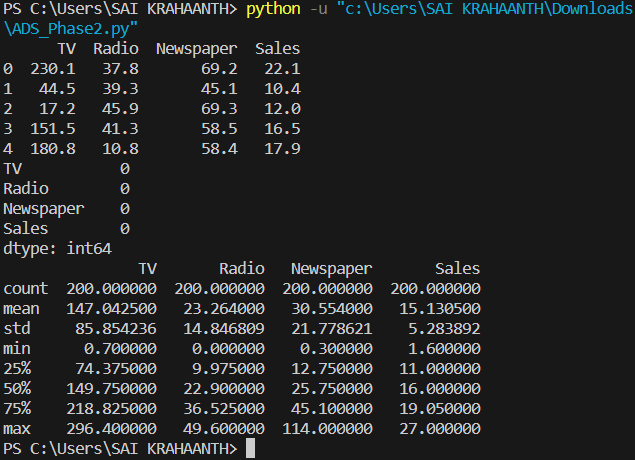
print(df.head())

# Check for missing values

print(df.isnull().sum())

# Get basic statistics

print(df.describe())

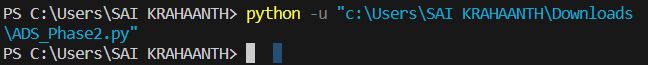
**

***#5) Data Preprocessing***

Depending on the dataset, data preprocessing can include tasks such as handling missing values, removing duplicates, and converting data types. Below are some common preprocessing steps

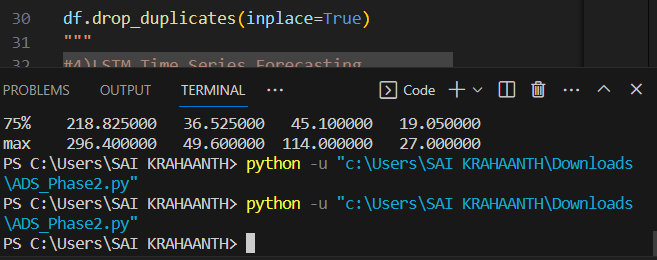
***#5.1) Handling Missing Values :***

df.fillna(df.mean(), inplace=True)

****

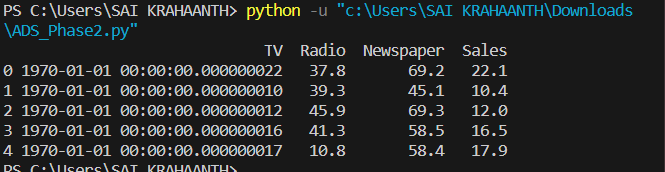
***#5.2) Removing Duplicates:***

df.drop\_duplicates(inplace=True)

****

***5.3) Data Type Conversion :***

df['TV'] = pd.to\_datetime(df['Sales'])

****

***#6) LSTM Time Series Forecasting (Feature Engineering)***

*Depending on your predictive model, you might need to engineer new features or transform existing ones to make them suitable for prediction.*

***#6.1)Install and import the necessary libraries for LSTM forecasting:***

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

***#6.2)Prepare the data for LSTM:***

lstm\_data = data[['TV', 'Radio', 'Newspaper', 'Sales']]

lstm\_data.columns = ['ds', 'TV', 'Radio', 'Newspaper']

print(lstm\_data.columns)

print(lstm\_data.head())

**Index(['ds', 'TV', 'Radio', 'Newspaper'], dtype='object')**

**ds TV Radio Newspaper**

**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**

***#4.3)Normalize the data using MinMaxScaler:***

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

lstm\_data['Radio'] = scaler.fit\_transform(lstm\_data['Radio'].values.reshape(-1, 1))

***#4.4)Create sequences for LSTM training:***

def create\_sequences(data, sequence\_length):

sequences = []

for i in range(len(data) - sequence\_length):

sequence = data[i:i + sequence\_length]

target = data[i + sequence\_length:i + sequence\_length + 1]

sequences.append((sequence, target))

return sequences

sequence\_length = 10

sequences = create\_sequences(lstm\_data['Radio'].values, sequence\_length)

***#4.6)Build and train the LSTM model:***

model = Sequential()

model.add(LSTM(50, input\_shape=(sequence\_length, 1)))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(train\_data, epochs=10, batch\_size=32)

***#4.7)Make predictions using the trained LSTM model:***

test\_x, test\_y = zip(\*test\_data)

test\_x = np.array(test\_x).reshape(-1, sequence\_length, 1)

predictions = model.predict(test\_x)

predictions = scaler.inverse\_transform(predictions

**Step 6: Evaluation**

1. R-squared (R^2) Score:

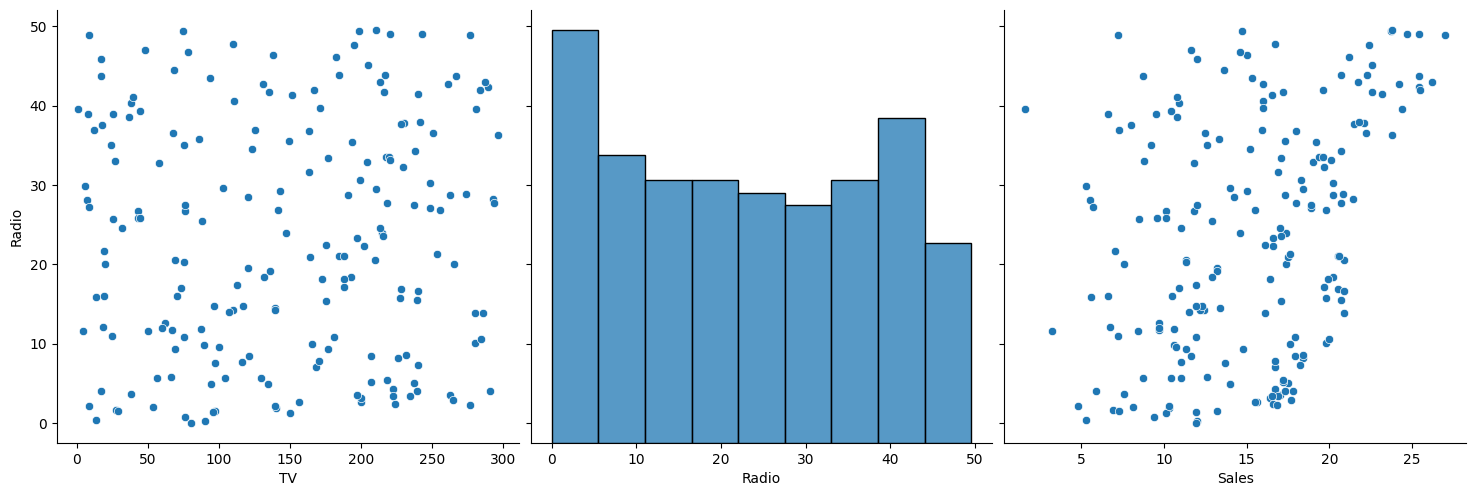
from sklearn.metrics import r2\_score r2 = r2\_score(y\_test, y\_pred) print("R-squared:", r2)

**Step 7 : Visualization:**

**# Pairplot for exploring relationships between variables**

sns.pairplot(data, x\_vars=['TV', 'Radio', 'Sales'], y\_vars='Radio', height=5, aspect=1, kind='scatter')

plt.show()



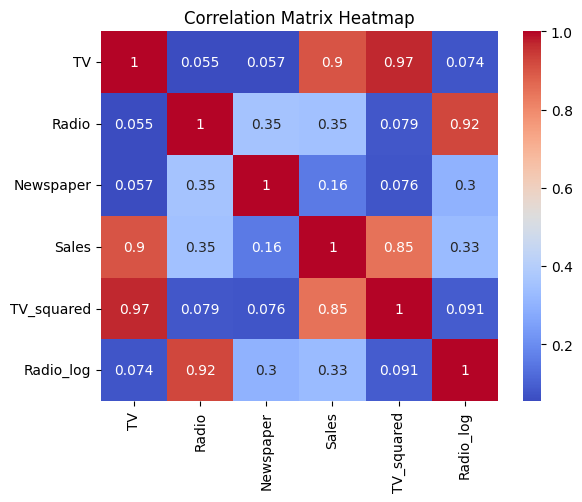
**# Correlation matrix heatmap**

correlation\_matrix = data.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title("Correlation Matrix Heatmap")

plt.show()



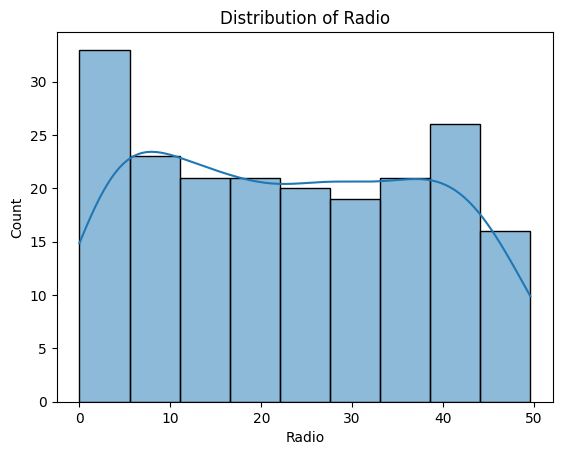
**# Distribution of the target variable**

sns.histplot(data['Radio'], kde=True)

plt.title("Distribution of Radio")

plt.xlabel("Radio")

plt.show()



**# Calculate Mean Squared Error (MSE) for Linear Regression**

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

print("Mean Squared Error (Linear Regression):", mse)

**# Visualize the actual vs. predicted values for Linear Regression**

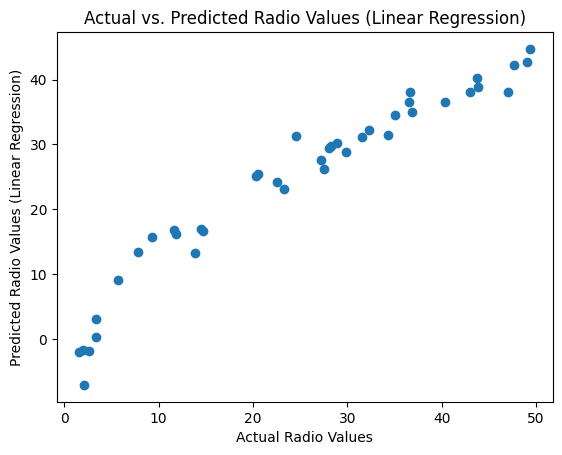
plt.scatter(y\_test, y\_pred)

plt.xlabel("Actual Radio Values")

plt.ylabel("Predicted Radio Values (Linear Regression)")

plt.title("Actual vs. Predicted Radio Values (Linear Regression)")

plt.show()



**Mean Squared Error (Linear Regression): 16.246546640902185**

**# Create and fit a Decision Tree Regressor model**

tree\_model = DecisionTreeRegressor()

tree\_model.fit(X\_train, y\_train)

y\_pred\_tree = tree\_model.predict(X\_test)

**# Calculate R-squared for Decision Tree Regressor**

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

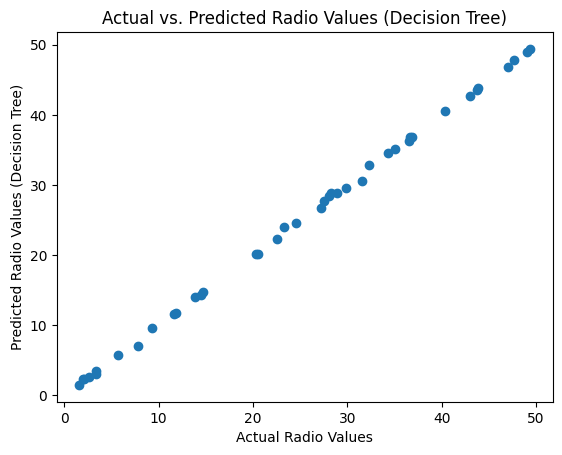
print("R-squared (Decision Tree):", r2)

**R-squared (Decision Tree): 0.9995186975731422**

**# Visualize the actual vs. predicted values for Decision Tree Regressor**

plt.scatter(y\_test, y\_pred\_tree)

plt.xlabel("Actual Radio Values")



plt.ylabel("Predicted Radio Values (Decision Tree)")

plt.title("Actual vs. Predicted Radio Values (Decision Tree)")

plt.show()

**# Create and fit a Linear Regression model**

model = LinearRegression()

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

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

**#Visualize the Predictions for the Decision Tree Model:**

plt.scatter(y\_test, y\_pred\_tree) plt.xlabel("Actual Radio Values") plt.ylabel("Predicted Radio Values (Decision Tree)") plt.title("Actual vs. Predicted Radio Values (Decision Tree)") plt.show()

import matplotlib.pyplot as plt

plt.scatter(y\_test, y\_pred)

plt.xlabel("Actual IMDb Scores")

plt.ylabel("Predicted IMDb Scores")

plt.title("Actual vs. Predicted IMDb Scores")

plt.show()

**#Cross-Validation**

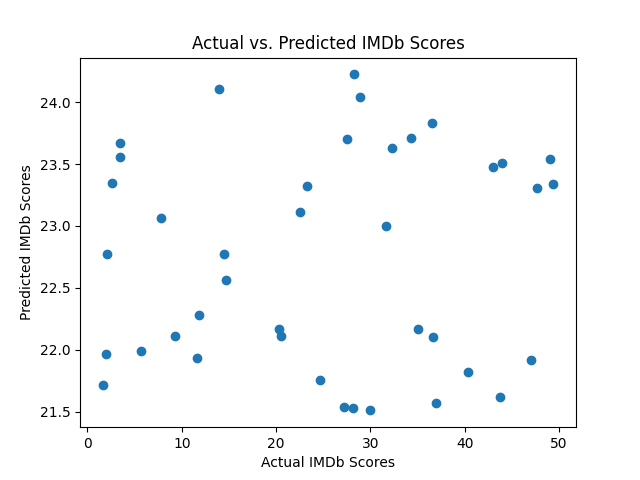
from sklearn.model\_selection import cross\_val\_score

scores = cross\_val\_score(tree\_model, X, y, cv=5, scoring='neg\_mean\_squared\_error')

mse\_cv = -scores.mean()

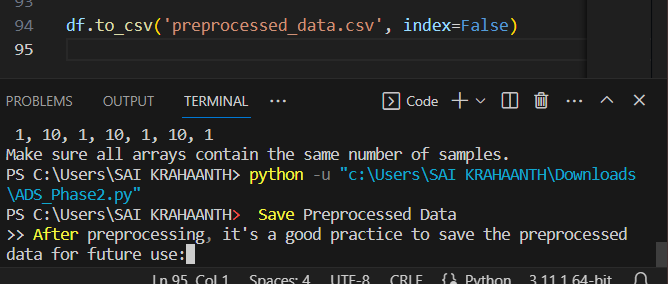
print("Mean Cross-Validated MSE:", mse\_cv)

**

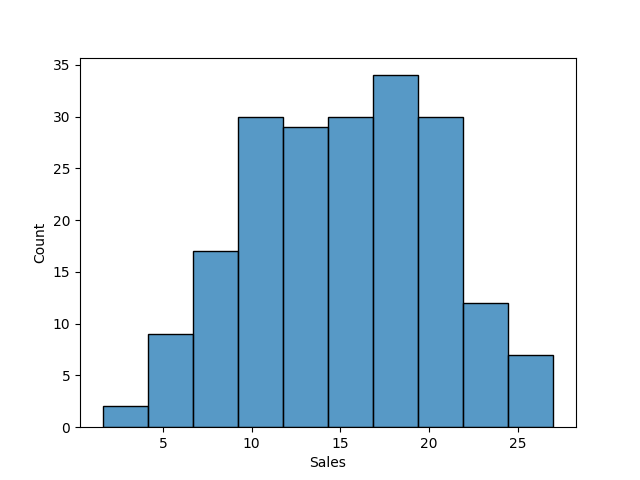
**

**Step 7: Save Preprocessed Data**

df.to\_csv('preprocessed\_data.csv', index=False)

**

# Output



## Input 2:

# Example:

import pandas as pd

data = pd.read\_csv(r'C:\Users\SAI KRAHAANTH\Downloads\Sales.csv')

# Assuming you have a 'Date' column and a 'Sales' column

X = data[['TV']]  # Replace with the actual features you want to use

y = data['Radio']   # Replace with your target variable

# Split your data into training and testing sets

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

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

# Optionally, perform any additional data preprocessing steps needed

# Import necessary deep learning libraries

import tensorflow as tf

from tensorflow.keras.layers import LSTM, Dense

from tensorflow.keras.models import Sequential

# Define your LSTM model

model = Sequential()

print("X\_train shape:", X\_train.shape)

print("y\_train shape:", y\_train.shape)

model.add(Dense(units=1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the model on your dataset

model.fit(X\_train, y\_train, epochs=10, batch\_size=32)  # Adjust epochs and batch\_size as needed

# Evaluate the model's performance

loss = model.evaluate(X\_test, y\_test)

# Output2:

X\_train shape: (160, 1)

y\_train shape: (160,)

Epoch 1/10

5/5 [==============================] - 0s 0s/step - loss: 21538.7754

Epoch 2/10

5/5 [==============================] - 0s 0s/step - loss: 21286.3379

Epoch 3/10

5/5 [==============================] - 0s 767us/step - loss: 21045.3379

Epoch 4/10

5/5 [==============================] - 0s 2ms/step - loss: 20793.3848

Epoch 5/10

5/5 [==============================] - 0s 2ms/step - loss: 20550.7930

Epoch 6/10

5/5 [==============================] - 0s 4ms/step - loss: 20313.3223

Epoch 7/10

5/5 [==============================] - 0s 6ms/step - loss: 20073.5195

Epoch 8/10

5/5 [==============================] - 0s 4ms/step - loss: 19833.5000

Epoch 9/10

5/5 [==============================] - 0s 4ms/step - loss: 19600.3086

Epoch 10/10

5/5 [==============================] - 0s 5ms/step - loss: 19363.0117

2/2 [==============================] - 0s 0s/step - loss: 17799.4961

# Conclusion:

* Methodology:

Explain the innovative approach taken to solve the problem. Mention that more advanced time series forecasting techniques like Prophet or LSTM networks will be explored for improved accuracy.

* Dataset

Provide a link to the dataset used for this analysis, which can be found at <https://www.kaggle.com/datasets/chakradharmattapalli/future-sales-prediction>.

* Implementation Plan

Outline the steps to implement the selected forecasting techniques, including data preprocessing, model training, and evaluation.

* Results: Share preliminary results or expectations from implementing these advanced techniques. Mention how they are expected to improve accuracy in predicting future sales.