FUTURE SALES PREDICTION - INNOVATION

INTRODUCTION:

The "Future Sales Prediction" project aims to create a model that utilizes past sales data to forecast future sales for a retail company. This predictive model serves the purpose of helping the company optimize its inventory management and make informed decisions based on data-driven insights. By accurately predicting future sales trends, the retail company can better plan for stock levels, marketing campaigns, and overall business strategies to improve efficiency and profitability.

ABOUT THE DATASET:

The dataset for the "Future Sales Prediction" project can be found on Kaggle using the following link: Future Sales Prediction Dataset.

DATASET LINK:

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

The dataset contains the following columns:

TV:

This column represents the amount of money spent on advertising through television. Television advertising is known for its wide reach and the potential to create awareness among a large and diverse audience. The values in this column indicate the advertising budget allocated to TV for different marketing campaigns or time periods.

Radio:

This attribute reflects the advertising expenditure on radio campaigns. Radio advertising is often used to target specific demographics or regions, and its effectiveness can vary based on factors such as the choice of radio stations and time slots for airing advertisements.

Newspaper:

This column measures the advertising spending on newspaper advertisements. Newspaper ads are typically employed for local or regional targeting and can come in various formats, sizes, and placements within the newspaper.

Sales:

This is the dependent variable in the dataset, representing the actual sales figures achieved during or after each advertising campaign. The primary objective of analyzing this data is to understand how the budgets allocated to TV, Radio, and Newspaper advertising impact sales, allowing businesses to make informed marketing decisions.

DETAILS OF THE LIBRARIES USED AND ALSO ABOUT THE METRICS USED FOR ACCURACY CHECK :

To perform sales prediction using a linear regression model, you will need several Python libraries. Here's a breakdown of the libraries you'll need and how to download them:

1. NumPy and Pandas: These libraries are essential for data manipulation and analysis.
   - To install NumPy: `pip install numpy`
   - To install Pandas: `pip install pandas`

2. Matplotlib: You can use Matplotlib for data visualization and creating plots.
   - To install Matplotlib: `pip install matplotlib`

3. Scikit-Learn: This library provides various machine learning algorithms and tools for model training and evaluation.
   - To install Scikit-Learn: `pip install scikit-learn`

Now, let's go through the steps to train and test your sales prediction model:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("Sales.csv")
```

```
print(data.head())

print(data.info())

print("The sum of duplicated items:", data.duplicated().sum())

print("To check the null values:", data.isna().sum())

plt.boxplot(data[['TV', 'Radio', 'Newspaper']])
plt.show()

X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']

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)

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

This code will load your dataset, preprocess it by checking for duplicates and null values, split it into training and testing sets, train a Linear Regression model, and evaluate its performance using Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2) metrics.

Mean Absolute Error (MAE):

MAE measures the average absolute difference between the predicted and actual values. It provides a straightforward understanding of how far, on average, your predictions are from the actual sales figures. Lower MAE values are desirable as they indicate that the model's predictions are closer to the true values.

Mean Squared Error (MSE):

MSE calculates the average of the squared differences between the predicted and actual values. Squaring the errors gives more weight to larger errors, making it useful for identifying outliers. Like MAE, lower MSE values indicate better accuracy. It's commonly used but can be sensitive to outliers.

R-squared (R2):

R2, also known as the coefficient of determination, represents the proportion of the variance in the target variable (sales) that is predictable from the features (TV, Radio, and Newspaper advertising expenditures). It ranges from 0 to 1, with higher values indicating a better fit. An R2 of 1 means the model explains all the variance, while an R2 of 0 means the model doesn't explain any variance.

TIME SERIES FORECASTING ALGORITHM:

Let's dive into the details of the two forecasting techniques: Facebook Prophet and Long Short-Term Memory (LSTM) networks, along with the provided code snippets.

Facebook Prophet

Algorithm Explanation:

Prophet is an open-source forecasting tool developed by Facebook. It is designed to handle time series data with strong seasonal patterns, missing data, outliers, and holidays.

Prophet decomposes time series data into three main components: trend, seasonality, and holidays. It uses a generalized additive model (GAM) to capture these components and make predictions.

Code Explanation:

```
from fbprophet import Prophet

# Prepare the data for Prophet
prophet_data = pd.DataFrame({'ds': data.index, 'y': data['Sales']})

# Create and fit the Prophet model
model = Prophet()
model.fit(prophet_data)

# Create a future dataframe for forecasting (e.g., 30 days into the future)
future = model.make_future_dataframe(periods=30)

# Make predictions
forecast = model.predict(future)

# Plot the forecast
fig = model.plot(forecast)
```

First, we prepare the data in a format that Prophet can work with, where 'ds' is the timestamp column and 'y' is the target variable.

We create a Prophet model, fit it to the historical sales data, and then create a future dataframe for forecasting, specifying how far into the future we want to forecast.

The model.predict(future) call generates forecasts for the specified time horizon.

Finally, we visualize the forecast using model.plot(forecast).

Long Short-Term Memory (LSTM) Network

Algorithm Explanation:

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) designed for sequence prediction tasks, making it well-suited for time series forecasting.

LSTMs can capture complex patterns and dependencies in time series data due to their ability to maintain and update information over long sequences.

Code Explanation:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Define the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(X_train.shape[1], 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
# Reshape the data for LSTM input
X_train_lstm = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test_lstm = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Train the model
model.fit(X_train_lstm, y_train, epochs=50, batch_size=32)

# Evaluate the model on the test data
loss = model.evaluate(X_test_lstm, y_test)
```

We start by defining an LSTM model using TensorFlow and Keras. The model architecture consists of an LSTM layer with 50 units and a ReLU activation function, followed by a Dense layer for regression.

The data needs to be reshaped into a 3D format to be compatible with LSTM input. This is done to accommodate the sequential nature of time series data.

We compile the model using the Adam optimizer and mean squared error (MSE) as the loss function.

The model is trained using the training data with a specified number of epochs and batch size.

Finally, we evaluate the model's performance on the test data and calculate the loss.

These two techniques provide different approaches to forecasting time series data. Prophet is an automated tool that excels at capturing seasonality and holidays,

while LSTM networks are deep learning models that can capture complex temporal patterns. The choice between them depends on the specific characteristics of your data and the desired level of model complexity.

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


<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


The sum of duplicated items: 0


To check the null values: TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

```
boxplot after treating outliers {'whiskers': [<matplotlib.lines.Line2D object at 0x7f9bb93ab280>, <matplotlib.lines.Line2D object at 0x7f9bb93ab520>, <matplotlib.lines.Line2D object at 0x7f9bb9408550>, <matplotlib.lines.Line2D object at 0x7f9bb94087f0>, <matplotlib.lines.Line2D object at 0x7f9bb9409690>, <matplotlib.lines.Line2D object at 0x7f9bb9409930>, <matplotlib.lines.Line2D object at 0x7f9bb940a8f0>, <matplotlib.lines.Line2D object at 0x7f9bb940ab90>], 'caps': [<matplotlib.lines.Line2D object at 0x7f9bb93ab7c0>, <matplotlib.lines.Line2D object at 0x7f9bb93aba60>, <matplotlib.lines.Line2D object at 0x7f9bb9408a90>, <matplotlib.lines.Line2D object at 0x7f9bb9408c10>, <matplotlib.lines.Line2D object at 0x7f9bb9409bd0>, <matplotlib.lines.Line2D object at 0x7f9bb9409e70>, <matplotlib.lines.Line2D object at 0x7f9bb940ae30>, <matplotlib.lines.Line2D object at 0x7f9bb940b0d0>], 'boxes': [<matplotlib.lines.Line2D object at 0x7f9bb93aaf88>, <matplotlib.lines.Line2D object at 0x7f9bb94082b0>, <matplotlib.lines.Line2D object at 0x7f9bb94093f0>, <matplotlib.lines.Line2D object at 0x7f9bb940a650>], 'medians': [<matplotlib.lines.Line2D object at 0x7f9bb93abd00>, <matplotlib.lines.Line2D object at 0x7f9bb9408eb0>, <matplotlib.lines.Line2D object at 0x7f9bb940a110>, <matplotlib.lines.Line2D object at 0x7f9bb940b370>], 'fliers': [<matplotlib.lines.Line2D object at 0x7f9bb93abfa0>, <matplotlib.lines.Line2D object at 0x7f9bb9409150>, <matplotlib.lines.Line2D object at 0x7f9bb940a3b0>, <matplotlib.lines.Line2D object at 0x7f9bb940b610>], 'means': []}


[[-4.04248386e-01 -1.02823707e+00 -3.37675384e-01]
 [ 3.20607716e-01 -9.19827737e-01 -1.16143931e+00]
 [-1.27051084e+00  2.59123702e-01  2.54250789e-01]
 [-1.04235941e+00 -6.96233499e-01 -5.74445854e-01]
 [ 8.79183401e-01 -1.38734296e+00 -7.07629243e-01]
 [-1.32873699e+00 -1.29926038e+00 -7.96418169e-01]
 [-9.43731452e-01 -4.65863678e-01  5.35415722e-01]
 [-3.23140256e-02  6.94073782e-02 -5.34984109e-01]
 [-5.39713297e-01 -1.16374872e+00  2.19721762e-01]
 [-8.75998996e-01  3.13328366e-01 -6.87898371e-01]
 [-8.53421511e-01  1.62101588e+00  2.24654481e-01]
 [ 2.18414888e-01 -1.06889056e+00 -8.45745350e-01]
 [-1.67928215e+00  1.76330312e+00  2.22240532e+00]
 [-1.68997675e+00  1.08574483e+00  1.01882210e+00]
 [-8.74810708e-01 -1.49575229e+00 -7.47090988e-01]
 [-2.45017701e-01 -1.16374872e+00  6.68075010e-02]
 [-9.10459368e-01 -3.98107848e-01 -8.40812632e-01]
 [ 1.65980907e+00  1.31611465e+00  1.04841841e+00]]
```

```
[-4.41085335e-01 -3.71005516e-01  4.26895923e-01]
[-1.49985056e+00  8.28272672e-01  1.77352797e+00]
[ 1.67169196e+00 -1.27215805e+00 -1.05785223e+00]
[-1.55213526e+00 -4.65863678e-01 -3.77137129e-01]
[ 1.70615233e+00  3.26879532e-01 -1.38834434e+00]
[-1.56045328e+00 -7.30111414e-01 -3.22877230e-01]
[-1.86791555e-01 -1.21795339e+00 -1.01839048e+00]
[-1.47846136e+00  1.09252041e+00 -1.01839048e+00]
[-5.89621422e-01 -8.99500988e-01 -1.29955542e+00]
[-8.29655737e-01 -1.54995695e+00 -1.02332320e+00]
[ 4.56072627e-01 -3.23576435e-01 -2.14357432e-01]
[ 3.20607716e-01  7.13087762e-01  4.31828641e-01]
[-2.88984383e-01  9.50233166e-01  2.42957948e+00]
[ 7.07989829e-01 -1.54186860e-01 -9.49332431e-01]
[-8.45987280e-02  4.35288859e-01 -8.55610786e-01]
[ 4.14482522e-01  1.42452397e+00 -1.39327706e+00]
[ 8.60090782e-01 -1.25860689e+00  9.79360352e-01]
[ 1.08943050e+00  1.02476458e+00 -3.32742666e-01]
[-1.57827761e+00  1.56003563e+00  1.94124038e+00]
[-3.50775395e-01  3.81084195e-01 -7.76687297e-01]
[-7.16768312e-01 -8.79174239e-01  2.83847098e-01]
[ 4.88156421e-01  3.94635361e-01 -5.79378572e-01]
[-8.85505306e-01 -8.18193993e-01 -1.18117018e+00]
[ 5.13110484e-01 -3.03249686e-01  1.76366253e+00]
[-7.64299859e-01  8.75701753e-01  9.54696761e-01]
[ 1.38887925e+00  1.41774839e+00 -1.23049736e+00]
[-1.31328923e+00  1.23480765e+00 -1.19103562e+00]
[ 1.32233508e+00  1.34321698e+00  1.22106354e+00]
[-1.62580916e+00 -4.72639260e-01  9.69494916e-01]
[ 5.19051927e-01  8.48599421e-01  2.25200163e+00]
[ 1.73942441e+00  9.09579668e-01  3.49997931e+00]
[ 1.37343150e+00 -1.94840358e-01 -1.46233511e+00]
[ 7.68592552e-01  7.61829612e-02 -1.27982454e+00]
[-3.79509701e-03  8.62150587e-01 -1.18117018e+00]
```

```
[-2.24816794e-01  1.34999256e+00 -5.15777338e-02]
[-1.48440281e+00  1.91367872e-01  6.58733675e-01]
[ 1.60188742e-01  9.43457583e-01 -1.11211213e+00]
[ 1.17379900e+00  2.86226034e-01 -3.47540821e-01]
[ 6.23621332e-01 -3.90019495e-02  8.16056553e-02]
[-1.62699745e+00 -1.52285462e+00 -2.14357432e-01]
[-1.73394343e+00 -7.63989329e-01 -1.19596834e+00]
[-6.66860187e-01  1.39742164e+00  1.01388938e+00]
[ 1.34134770e+00 -1.31281155e+00 -5.15253237e-01]
[ 9.30199814e-01 -4.04883431e-01 -1.84761123e-01]
[ 1.23321343e+00 -1.06757779e-01  2.68216547e-03]
[ 1.10725483e+00  1.77007871e+00  7.08060856e-01]
[ 1.06685301e+00 -1.27215805e+00  3.43039715e-01]
[ 9.26634948e-01  1.00443783e+00  1.01336528e-01]
[ 7.76910573e-01  4.90806292e-02  1.36411236e+00]
[ 1.06804130e+00  1.26190998e+00 -5.64580418e-01]
[-5.06441213e-01 -6.01375337e-01 -9.39466994e-01]
[ 4.48942894e-01 -1.20308945e-01 -1.00852505e+00]
[ 6.75906034e-01 -9.80807984e-01 -1.74895687e-01]
[-1.27169913e+00  2.04919038e-01 -4.65926055e-01]
[ 1.75940994e-02  1.24835882e+00  1.40850683e+00]
[-1.44013163e-01  1.59391355e+00  1.43317042e+00]
[ 3.87151882e-01  1.58036238e+00  1.41837226e+00]
[ 8.29195276e-01  7.19863345e-01  7.47522601e-01]
[ 7.81968227e-02 -1.37379180e+00 -1.06771767e+00]
[ 1.50533154e+00 -1.39411855e+00 -3.08079076e-01]
[ 6.53328549e-01  1.50583097e+00 -5.10320518e-01]
[-9.87698133e-01 -7.57213746e-01  3.38106997e-01]
[-8.74810708e-01  2.59123702e-01 -3.77137129e-01]
[-6.45470990e-01 -1.45509879e+00 -1.11211213e+00]
[-3.54340261e-01 -2.21942690e-01 -9.04937968e-01]
[ 9.00492597e-01 -9.94359150e-01  1.30985247e+00]
[-1.44281270e+00 -1.44832321e+00  1.50663709e-01]
[-9.69873803e-01  1.46517747e+00  2.78914380e-01]
```

```
[ 1.33897112e+00  4.01410944e-01 -6.92831089e-01]
[-7.08450291e-01 -1.52963021e+00 -3.32742666e-01]
[ 5.91537537e-01 -1.33991388e+00  2.29587199e-01]
[-6.63295321e-01 -1.21795339e+00 -1.07758310e+00]
[ 7.22249293e-01  1.81073220e+00  3.82501460e-01]
[ 9.17128639e-01 -4.79414843e-01  9.84293070e-01]
[-7.33404353e-01  1.77816706e-01  2.14348183e+00]
[ 1.03833409e+00 -1.20440222e+00 -3.17944512e-01]
[-1.64214070e-01 -2.49045022e-01 -6.58302062e-01]
[ 2.67134725e-01 -3.23576435e-01  3.72111923e-02]
[-1.57827761e+00 -1.27215805e+00  8.16056553e-02]
[-1.07444321e+00 -7.36886997e-01  6.48868238e-01]
[-8.95011616e-01  1.79718104e+00  7.77118909e-01]
[-2.60680832e-03 -1.46187438e+00 -2.78482767e-01]
[ 1.99402269e-01  1.29578790e+00 -1.29955542e+00]
[-1.25387480e+00  1.11284716e+00  7.47522601e-01]
[ 7.88793460e-01  1.27546115e+00  4.76223104e-01]
[-1.25149822e+00  1.98143455e-01 -4.60993337e-01]
[-1.77434525e+00  1.13317391e+00 -1.04798679e+00]
[-3.44833951e-01 -9.80807984e-01  9.25100453e-01]
[ 4.50131183e-01 -3.84556682e-01 -5.94176726e-01]
[-1.76096957e-01  1.27546115e+00  7.86984346e-01]
[-1.28565410e-01 -5.81048588e-01 -2.14357432e-01]
[-4.67227687e-01  1.20092974e+00  1.64034458e+00]
[ 7.53144799e-01  1.16836459e-01 -8.30947196e-01]
[-1.55926499e+00 -7.96554474e-02  1.00895666e+00]
[ 9.75354785e-01 -9.67256818e-01 -1.04798679e+00]
[ 8.12559234e-01  3.26879532e-01  1.15693820e+00]
[ 1.61584239e+00 -6.08150920e-01 -1.29462270e+00]
[-4.77922285e-01 -5.81048588e-01  8.65383734e-02]
[-1.48559110e+00 -8.04642827e-01 -1.21159889e-02]
[ 6.42633951e-01  6.79209847e-01  7.91917064e-01]
[ 8.04241213e-01  7.19863345e-01  1.43317042e+00]
[ 1.85142805e-01 -8.72398657e-01 -6.08974881e-01]
```

```
[[ 0.15781217  0.59112727  1.13227461]
 [ 0.53925283  1.68199613  1.13227461]
 [ 1.69783431  0.36753303  0.65380096]
 [-1.64363349  0.95023317  0.75245532]
 [ 0.83513672  1.77007871 -1.31928629]
 [-0.89025846  0.82149709  1.12240918]
 [ 0.79354661  1.42452397 -0.13543394]
 [-1.18851892 -0.76398933 -0.56951314]
 [ 0.86009078 -1.31958713 -0.8309472 ]
 [ 0.29803023 -0.02545078  0.07667294]
 [-1.40835233  0.11683646 -1.36861347]
 [-1.11484502 -1.16374872 -0.01211599]
 [ 1.00387371 -1.31958713  2.70581169]
 [-1.71849568  0.47594236 -1.01345777]
 [-0.12500054 -1.40766971 -0.16503025]
 [ 0.23980408 -1.02146148  0.25918351]
 [-1.69591819  0.35398186  0.56501203]
 [ 0.56539519  0.02875388 -0.7766873 ]
 [-0.88788188 -0.17451361  0.12600012]
 [ 1.03833409  0.31332837 -0.93453428]
 [ 0.94445928  0.63855635  2.18294357]
 [-0.97700354  0.92990642  4.14616538]
 [-1.33111357  1.18060299 -0.89013981]
 [ 1.19875306  0.92313083  2.08922193]
 [-0.96274407 -0.91982774 -1.43273881]
 [-1.14692882 -1.4144453  -0.42153159]
 [ 0.75433309  1.36354373  0.19012545]
 [-0.12737712 -0.56749742 -0.97399602]
 [-0.74647553 -0.75043816 -0.19955928]
 [-1.68284702  0.29300162 -1.37354619]
 [ 0.59153754 -1.3737918  -0.43139703]
 [-0.96036749 -0.16096244 -0.57444585]
```

CONCLUSION:

In conclusion, the "Future Sales Prediction" project is a valuable initiative aimed at helping a retail company optimize its inventory management and marketing strategies through data-driven insights. The project leverages a dataset that includes information on advertising expenditures in various media channels (TV, Radio, and Newspaper) and their impact on actual sales figures. The goal is to build a predictive model that can accurately forecast future sales trends.

The document outlines the essential libraries required for data manipulation, analysis, and model development, such as NumPy, Pandas, Matplotlib, and Scikit-Learn. It also provides a step-by-step guide on how to preprocess the data, split it into training and testing sets, train a Linear Regression model, and assess its performance using key metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2).

Additionally, the document introduces two distinct time series forecasting techniques: Facebook Prophet and Long Short-Term Memory (LSTM) networks. Each technique is accompanied by code snippets and explanations to help users understand their implementation and choose the most suitable approach based on their data and project requirements.

This comprehensive guide equips individuals or teams interested in sales prediction with the necessary knowledge and tools to create accurate forecasts and make informed business decisions. By harnessing past sales data and leveraging modern machine learning and time series forecasting techniques, companies can enhance their efficiency, profitability, and competitiveness in the retail industry.