# Time Series Prediction Using LSTM - Detailed Documentation

## Table of Contents

## 1. Introduction

The objective of this project is to forecast monthly retail sales using Long Short-Term Memory (LSTM) neural networks. The dataset contains historical sales data, which the LSTM model uses to predict future sales. LSTM is a type of recurrent neural network (RNN) well-suited for sequence prediction problems, particularly when dealing with time series data.

## 2. Dataset Description

The dataset used in this project is named retail_sales.csv. It contains the following columns:

**Date:** The date of the recorded sales, formatted in a monthly interval. This column serves as the time index.

**Sales:** The sales figures for the corresponding month. This is the target variable that we aim to predict.

The data spans a significant period, allowing us to capture seasonal patterns, trends, and other temporal dynamics essential for accurate forecasting.

## 3. Prerequisites and Libraries

The project relies on several Python libraries for data manipulation, visualization, and building the machine learning model. The primary libraries used include:

**pandas**: For data manipulation and analysis, including reading the dataset and handling date-time operations.

**numpy**: For numerical computations and array manipulations.

**matplotlib**: For data visualization, to plot the sales data and model predictions.

**scikit-learn**: For preprocessing tasks such as feature scaling and evaluation metrics.

**tensorflow and keras**: For building and training the LSTM neural network model.

## 4. Data Loading and Preprocessing

### Loading the Dataset

The dataset is loaded into a pandas DataFrame, which allows for easy manipulation and analysis of the data. The initial step involves reading the CSV file containing the sales data.

### Initial Data Inspection

After loading the data, it is crucial to inspect the first few rows and the column names to understand the structure and ensure that the data has been loaded correctly.

**Data Cleaning**

**Strip Column Names**: Removing any leading or trailing whitespace from the column names to avoid potential issues during data manipulation.

**Convert Date Column**: Converting the Date column to a datetime format. This step is essential for time series analysis as it allows for proper indexing and plotting of the data based on time.

**Set Date as Index**: Setting the Date column as the index of the DataFrame. This transformation is necessary for time series forecasting, as it allows the model to recognize the temporal sequence of the data.

**Inspecting the Cleaned Data**

Inspecting the first few rows and checking the data types ensures that the preprocessing steps have been applied correctly. This step helps confirm that the Date column is in datetime format and that it is set as the index.

## 5. Data Visualization

Visualizing the time series data is a crucial step to understand the trends, seasonality, and potential anomalies in the sales data. Plotting the sales data over time provides insights into the overall trend and any periodic patterns that may be present. This visualization helps in understanding the data's temporal dynamics, which is essential for building an accurate forecasting model.

## 6. Feature Scaling

Feature scaling is a preprocessing step that transforms the data to a specific range, typically [0, 1]. This step is particularly important for neural networks, as it helps in faster convergence during training and ensures that all features contribute equally to the model. In this project, the sales data is scaled using MinMaxScaler from the scikit-learn library.

## 7. Preparing Data for LSTM

### Creating the Dataset

The LSTM model requires input data to be in a specific format, where each input sequence contains a fixed number of previous observations. This process is known as creating a sliding window of observations. The create_dataset function generates input-output pairs from the time series data. The input sequences (X) contain a specified number of previous sales values (look-back period), and the output (Y) is the sales value immediately following the input sequence.

### Splitting the Data

The dataset is split into training and testing sets. Typically, 80% of the data is used for training the model, and the remaining 20% is used for testing its performance. This split ensures that the model is evaluated on unseen data, providing an unbiased estimate of its performance.

### Reshaping Data for LSTM

The input data is reshaped to match the expected input shape for LSTM networks, which is [samples, time steps, features]. This reshaping is necessary for the LSTM to process the sequential data correctly.

# 8. Model Building and Training

### Building the LSTM Model

The LSTM model is constructed using the Keras Sequential API. The architecture includes:

**<u>Two LSTM layers</u>**: These layers capture the temporal dependencies in the sales data. The first LSTM layer returns sequences to be fed into the second LSTM layer.

**<u>Two Dense layers</u>**: The first Dense layer has 25 units, and the final output layer has 1 unit, corresponding to the predicted sales value.

### Compiling the Model

The model is compiled using the Adam optimizer, which is well-suited for training deep learning models, and the mean squared error (MSE) loss function, which measures the average squared difference between predicted and actual values.

### Training the Model

The model is trained using the training data, with a batch size of 1 and for 20 epochs. The validation data (test set) is used to monitor the model's performance during training, helping to detect overfitting.

**Plotting Training and Validation Loss**

Plotting the training and validation loss over epochs helps visualize the model's learning process. It indicates how well the model is fitting the training data and how it performs on the validation set.

## 9. Model Evaluation

**Making Predictions**

The model is used to predict sales for both the training and test sets. These predictions are then compared to the actual sales values to evaluate the model's performance.

**Inverse Transforming Predictions**

Since the sales data was scaled before training, the predictions need to be inverse transformed to the original sales scale for accurate evaluation.

**Calculating Performance Metrics**

The model's performance is evaluated using the following metrics:

**Root Mean Squared Error (RMSE)**: Measures the square root of the average squared differences between predicted and actual values.

**Mean Absolute Error (MAE)**: Measures the average absolute differences between predicted and actual values.

**R-squared (R2) score**: Indicates the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

## 10. Predictions and Visualization

### Plotting Predictions

The actual sales, training predictions, and test predictions are plotted for visual comparison. This plot helps in assessing how well the model's predictions align with the actual sales data.

## 11. Future Sales Forecasting

### Forecasting Future Sales

The model is used to forecast sales for a specified number of future periods (e.g., 12 months). This involves using the last available data points to predict the next sales value iteratively.

### Plotting Future Predictions

The future sales predictions are plotted alongside the actual sales data to visualize the model's forecasting performance. This plot provides insights into how the model is expected to perform in future periods based on the learned patterns.

## 12. Conclusion

This project demonstrates the application of LSTM neural networks for time series forecasting using historical retail sales data. The steps involved include data preprocessing, model building, training, evaluation, and forecasting. The model's performance is evaluated using standard metrics, and its predictions are visualized to assess accuracy. Future sales forecasting provides a practical application of the model, showcasing its ability to predict future trends based on historical data.