Title: Time Series Prediction using Deep Learning

Introduction

Objective

The objective of this project is to develop a predictive model using a deep learning framework (TensorFlow) to forecast future sales from historical time series data. We aim to select a suitable deep learning algorithm to accurately predict future time steps and evaluate the model's performance using appropriate metrics.

Dataset Description

The dataset consists of hourly sales data from a major retail chain spanning from January 1, 2015, to December 31, 2020. It includes the following columns:

- Date: The timestamp of the sales data record.
- Sales: The total sales value in USD at each hour.

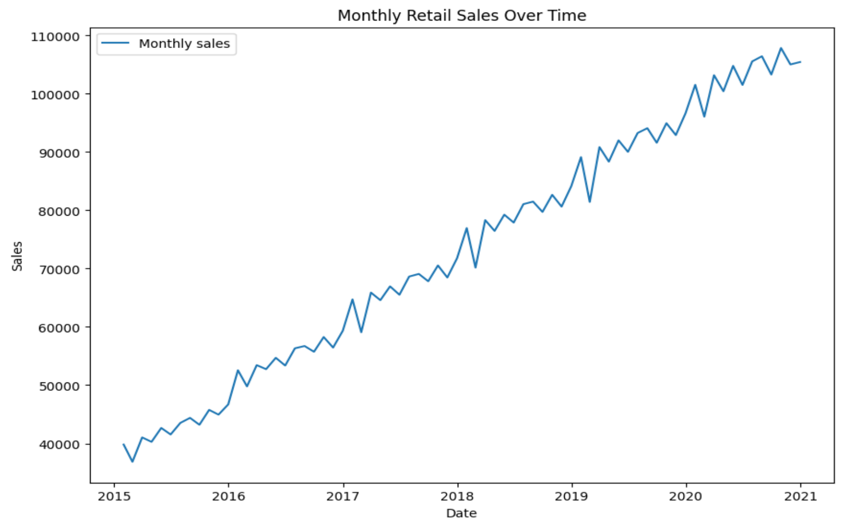
Exploratory Data Analysis

Initial Data Analysis

I loaded the dataset and examined its structure, including the presence of any missing values and the distribution of sales over time.

Data Visualization

To understand the trends and patterns in the data, I plotted the sales data over time.



Data Preprocessing

Handling Missing Values

I checked for any missing values in the dataset and handled them using forward fill method to maintain the continuity of the time series.

Resampling Data

Given the dataset's hourly granularity, I resampled it to a monthly level to capture long-term trends and seasonality.

- Resampled the data to monthly frequency, summing up the sales values.

Normalization

The sales data was normalized using **MinMaxScaler** to scale the values between 0 and 1 for better performance of the LSTM model.

Model Selection

Justification for Choosing LSTM

I selected the Long Short-Term Memory (LSTM) model for this task due to its ability to capture long-term dependencies and patterns in sequential data, which is crucial for time series forecasting.

Data Preparation for LSTM

I prepared the data by creating sequences of past observations to predict future values, I have used the **(past 12th months) data** to predict the future.

Model Implementation

Building the Model Architecture

- I built an LSTM model using TensorFlow.

Layers

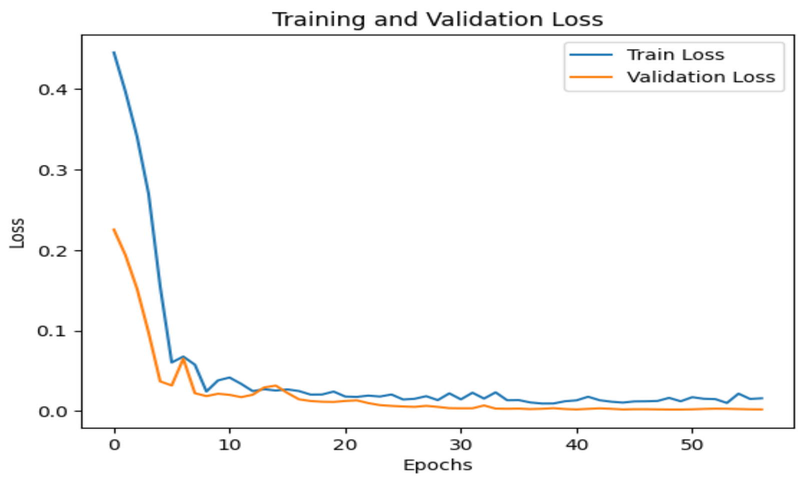
I have used 3 hidden layers with callback function to prevent overfitting and good learning.

- First LSTM layer with 60units, followed by another LSTM layer with 60units, and a final LSTM layer with 60units.
- A Dense output layer with 1 unit to predict the sales value.
- Along with every layer I used a function called Dropout.

- 'relu' used for LSTM layers to introduce non-linearity.
- To Compile the model, I have used the optimizer='Adam', loss='mse'.

Training the Model

I trained the model on the training data, using a validation split to monitor its performance during training.



Model Evaluation

Evaluation Metrics

I evaluated the model using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2).

Results and Discussion

The model's performance metrics were as follows:

- RMSE: [Calculated RMSE value] = `3390.179949953554`
- MAE: [Calculated MAE value] = `2925.714555679165`
- R^2: [Calculated R^2 value] = `0.9603427236562436`

The LSTM model showed best accuracy in forecasting future sales, capturing the overall trend and seasonal patterns in the data.

Conclusion

Summary of Findings

I successfully developed and evaluated a time series forecasting model using an LSTM neural network. The model was able to capture the underlying trends and seasonality in the retail sales data, providing accurate forecasts for future sales.

Future Work

For future improvements, we can explore the following:

- Incorporating additional features: Include external factors such as promotional events, holidays, and economic indicators to improve the model's accuracy.
- Hyperparameter tuning: Further optimize the model's hyperparameters using grid search or random search techniques.
- Advanced models: Experiment with other advanced models like GRU, Transformer, or hybrid models combining LSTM with other architectures.

By following these steps and documenting this process, I developed a robust time series forecasting model using deep learning, which can be further enhanced for even better performance.

Methods Used

- Exploratory Data Analysis (EDA)
- Data Collection
- Data Cleaning
- Feature Engineering
- Modelling

Technologies used

- Python
- Deep Learning
- Pandas, Numpy, Matplotlib
- TensorFlow
- Dropout
- Early Stopping
- ReduceLROnPlateau

Hyperparameter tuning

I have adjust the epochs count=80, batch_size=20, learning rate=0.001 ,adding hidden layer units , Dropout function and callback functions like (Early stopping) to tune the model and prevent that model learn less **Noice** and more S**ignals** to avoid underfitting or overfitting.