Rossmann Pharmaceuticals Sales Forecasting using LSTM: A Comprehensive End-to-End Approach

Executive Summary

Forecasting future sales is essential for effective decision-making in retail, and Rossmann Pharmaceuticals faces the challenge of predicting store sales across several cities six weeks in advance. The finance team, which previously relied on managers' judgment, requires a more data-driven approach that incorporates promotional campaigns, holidays, seasonality, and competition information. This project aims to deliver an end-to-end solution for sales forecasting using traditional machine learning models as well as deep learning techniques, specifically Long Short-Term Memory (LSTM) networks.

The primary goal is to help the Rossmann finance team plan ahead by delivering accurate daily sales predictions that enable better inventory management and resource allocation. The solution involves data preprocessing, model building, evaluation, and finally, model serialization for use in a real-world environment. In this blog, I will present the problem-solving approach, technical steps, and findings with detailed insights.

Business Overview

Problem Statement

Rossmann Pharmaceuticals, like many retail companies, needs to accurately forecast sales for over 1,000 stores spread across various cities. Accurate forecasting is critical for optimizing inventory, resource management, and promotional strategies. My role as a Machine Learning Engineer is to develop a predictive model that forecasts daily sales for each store six weeks ahead, incorporating a variety of factors like promotions, competition, holidays, and more.

Key Questions

This project aims to answer a variety of questions critical to store performance:

- How do holidays and promotions influence sales?
- What are the seasonal patterns in customer purchasing behavior?
- Can we identify stores where promotions are underperforming or overperforming?
- How does competition affect sales?
- How well can we predict sales for the next six weeks using machine learning and deep learning models?

Task 1: Exploratory Data Analysis (EDA)

Overview

Before building any machine learning model, it's crucial to understand the data. Exploratory Data Analysis (EDA) was conducted to explore customer purchasing behavior, the influence of holidays and promotions, and how features like store type and competition distance impact sales. The goal of the EDA is to gain insights that can guide feature engineering and inform modeling decisions.

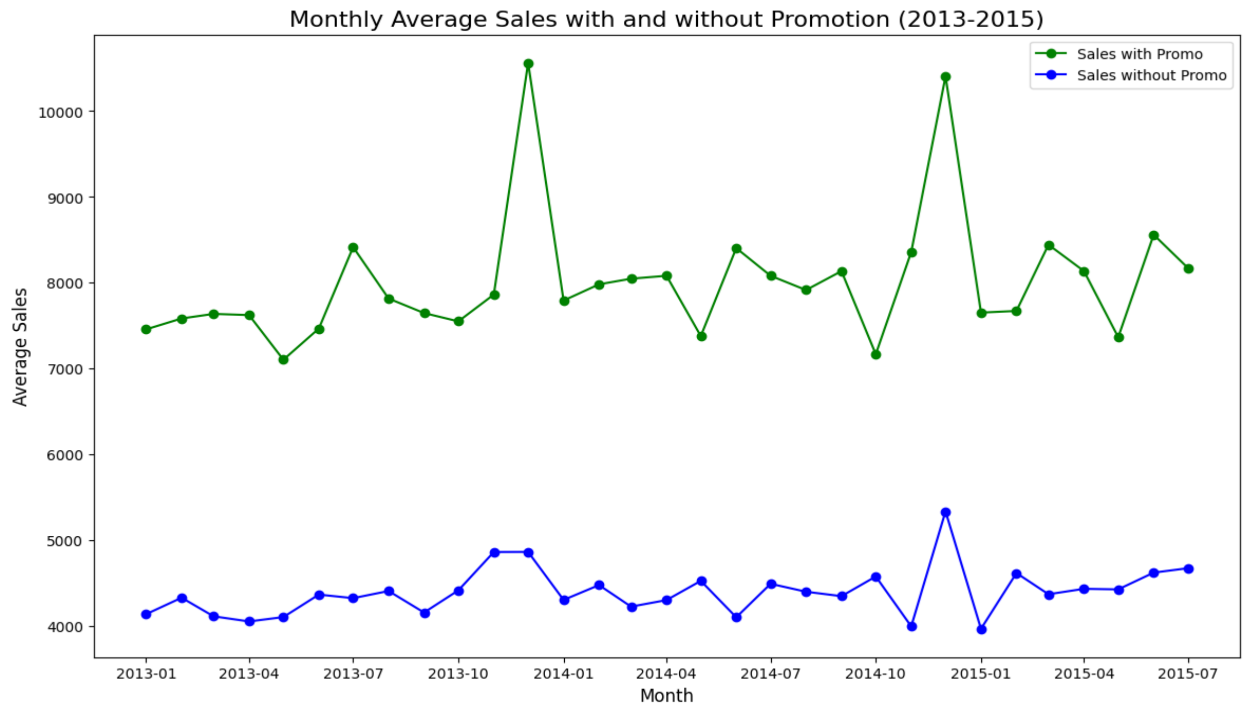
Key Findings from EDA

1. **Holiday Sales Impact**: Sales were significantly higher during holidays, particularly around public holiday and Easter. Stores that remained open during these holidays saw a substantial boost in revenue. However, sales dropped sharply after holidays.
2. **Promotional Influence**: Promotions attracted more customers and led to higher sales, particularly during weekdays. Some stores benefited more from promotions than others, indicating that promotions may be more effective for certain store types or regions.
3. **Seasonal Trends**: We observed clear seasonal trends in customer behavior. For example, sales peaked during the summer and dropped during the winter months, except during the holiday season.
4. **Store Type and Assortment**: Stores with a larger assortment of products tended to have higher sales, while smaller stores with limited offerings saw less foot traffic. Assortment type also influenced customer purchasing behavior during promotional periods.
5. **Competition Distance**: Stores located closer to competitors generally saw lower sales, particularly when the competitor opened recently. However, for stores in city centers, competition distance had less impact due to the high density of customers.

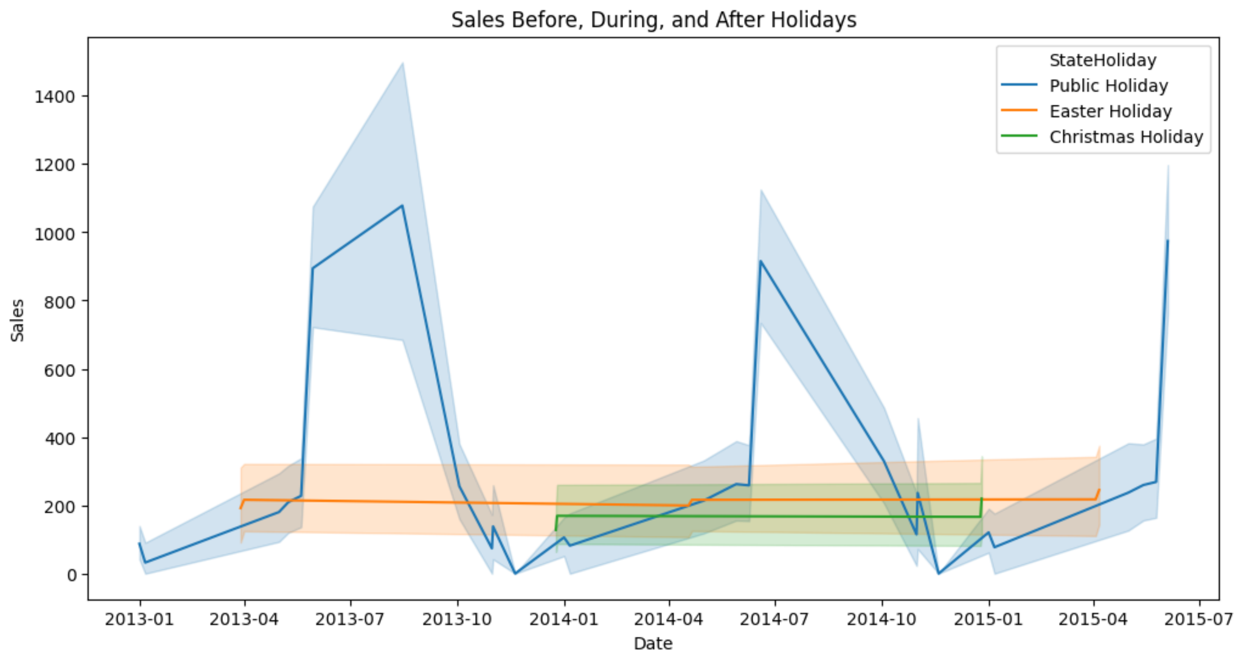
Visualizations and Insights

Key visualizations were generated to illustrate these trends, such as:

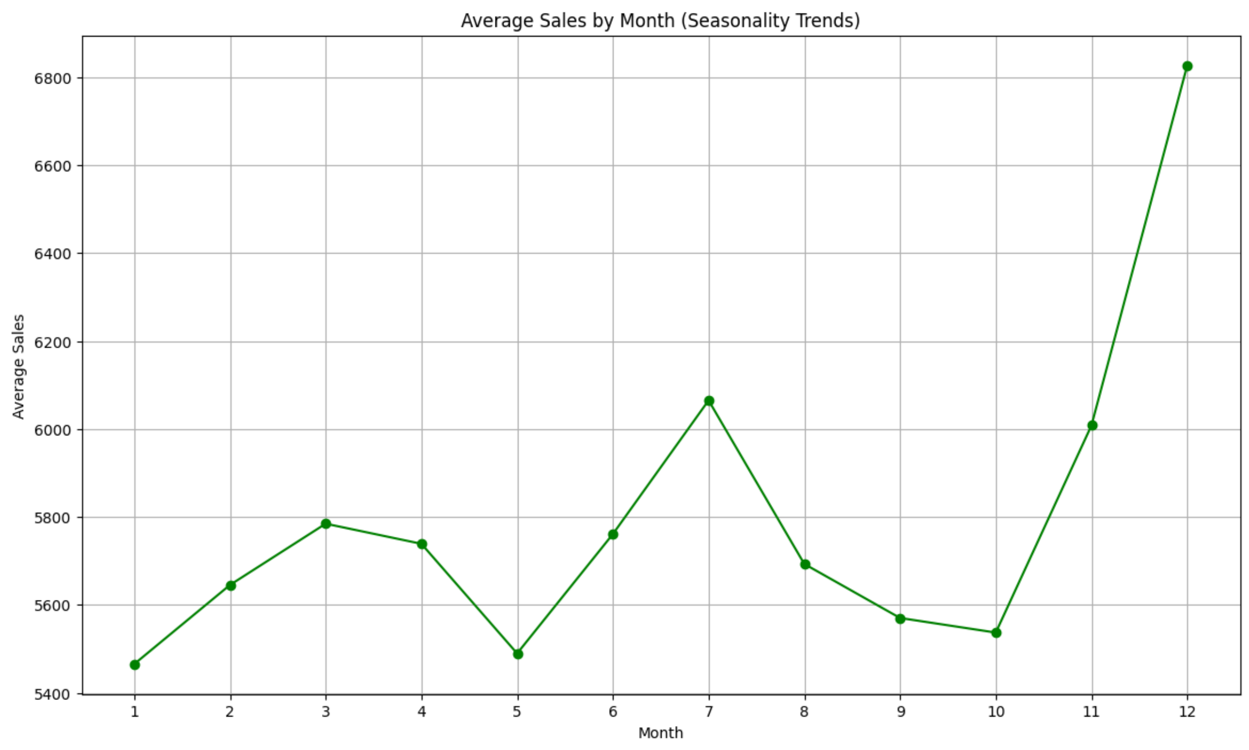
- **Sales vs. Promotions**: A line graph showing the spike in sales during promotional periods.



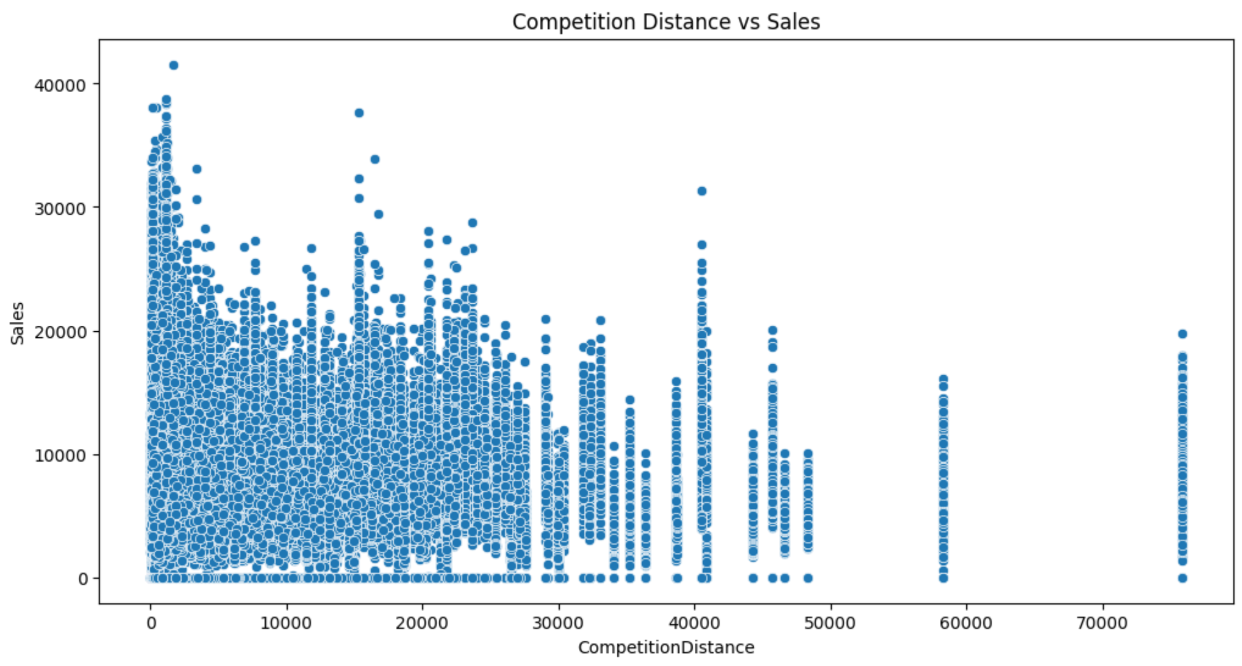
- **Holiday Sales Distribution**: A line chart illustrating sales before, during, and after major holidays.



- **Seasonality Trends**: A time series plot showing sales trends across different months of the year.



- **Impact of Competition**: A scatter plot correlating store distance from competitors with daily sales.



These insights informed feature selection and feature engineering steps during the modeling phase.

Task 2: Prediction of Store Sales

2.1 Data Preprocessing

Effective machine learning models rely on properly preprocessed data. For this project, I performed the following preprocessing steps:

1. **Handling Missing Data**: Missing values in features such as CompetitionDistance and Promo2SinceWeek were filled or imputed based on business logic and exploratory insights.
2. **Feature Engineering**: Key features were engineered from existing columns. For example:
   - From the Date column, I extracted the day of the week, month, and year.
   - I created features to capture proximity to holidays (e.g., days before or after a holiday).
   - A binary feature indicating the beginning, middle, or end of the month was generated.
3. **Scaling**: Continuous variables such as Sales, Customers, and CompetitionDistance were scaled using MinMaxScaler to ensure the data was in a suitable range for machine learning models.
4. **Encoding Categorical Variables**: Categorical features like StoreType and Assortment were one-hot encoded.
5. **Train-Test Split**: The dataset was split into training (70%) and test sets (30%), ensuring that the time-based order was preserved (i.e., no random shuffling).

2.2 Building Models with Sklearn Pipelines

I used scikit-learn pipelines to streamline the model building process, making it modular and easily reproducible. For baseline models, I focused on tree-based algorithms like Random Forest, which are robust and provide feature importance scores.

Baseline Model: Random Forest Regressor

Random Forest was chosen due to its ability to handle large datasets and non-linear relationships. Below is a sample of the pipeline used for Random Forest modeling:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler


pipeline = Pipeline([
   ('scaler', StandardScaler()),
   ('model', RandomForestRegressor(n_estimators=100, random_state=42))
])
pipeline.fit(X_train, y_train)
```

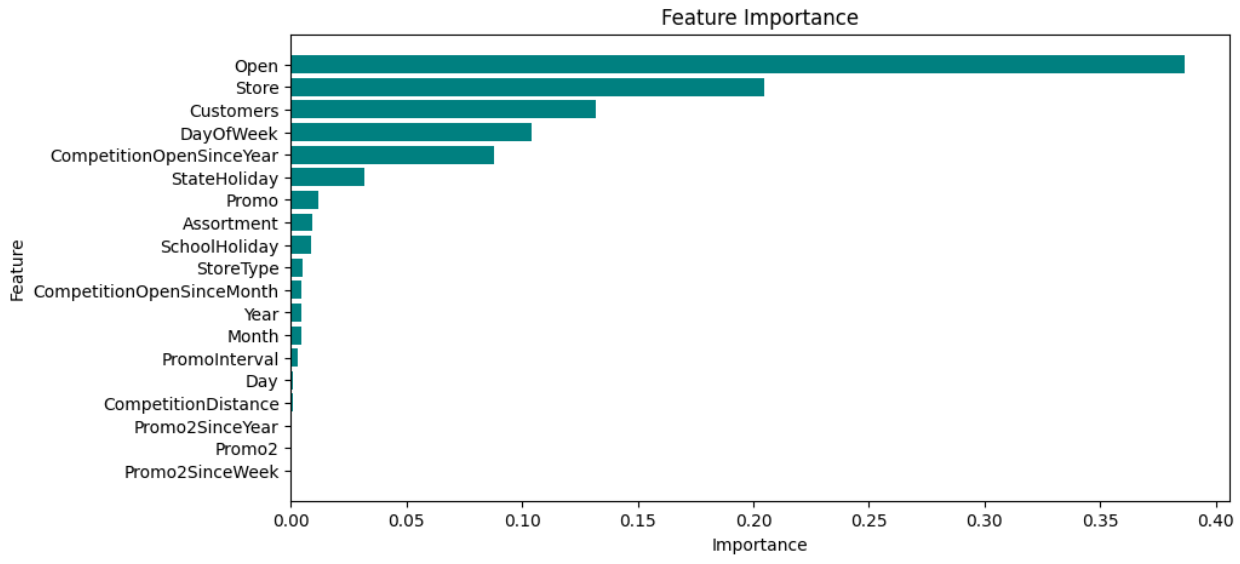
2.3 Loss Function Choice

For this regression task, where the goal is to minimize the difference between predicted and actual sales values, I selected **Mean Squared Error (MSE)** as the loss function. MSE is well-suited for such forecasting problems because it penalizes larger deviations more heavily than smaller ones. This feature is especially important in sales forecasting, where significant deviations from actual sales could result in either inventory shortages or surplus, leading to inefficient resource management.

- **Cross-Validation RMSE**: 1689.62
  This indicates that, on average, the predicted sales values deviate from the actual sales by about 1689.62 units during cross-validation. This metric provides an indication of how well the model might generalize to unseen data.
- **Test RMSE**: 1684.22
  The test set RMSE being close to the cross-validation RMSE suggests that the model generalizes well to new data, showing consistency in performance.
- **R^2 Score**: 0.808
  The R^2 score of 0.808 means that the model explains 80.8% of the variance in sales based on the input features. This is a strong indicator that the model captures essential patterns, and its performance is satisfactory, though there's room for further improvement.

2.4 Post-Prediction Analysis: Digging into Feature Importance

After training the Random Forest model, the next step was analyzing **feature importance**. This allowed me to pinpoint which factors contributed the most to predicting sales, providing actionable insights for improving the model and the business strategy.



Most Important Features:

1. **Open**: Whether a store is open or closed was one of the most influential factors, understandably, as no sales happen when the store is closed.
2. **Store**: Different stores exhibited varying sales patterns, likely due to location, customer base, or store-specific promotions.
3. **Customers**: Unsurprisingly, the number of customers significantly impacted sales. However, relying heavily on this feature may limit the model's ability to generalize without customer data.
4. **DayOfWeek**: Sales varied based on the day of the week, with distinct differences between weekdays and weekends.

Moderately Important Features:

- **CompetitionOpenSinceYear** & **StateHoliday**: External factors such as the presence of competition and holiday seasons had a moderate effect on sales, highlighting the need to consider market conditions when forecasting.
- **Promo**: Promotional activities impacted sales, but to a lesser degree than expected. This could be due to how promotions were encoded in the data or their timing relative to other factors.
-

Least Important Features:

Features such as **Promo2SinceWeek**, **Promo2SinceYear**, **CompetitionDistance**, and **Day** showed minimal importance. These may not have had a direct influence on sales, or their contribution might be redundant due to overlap with more critical features.

Actionable Insights for Model Refinement

1. **Focus on Critical Features**: Prioritizing key predictors like "Open," "Store," "Customers," and "DayOfWeek" for further refinement could improve the model's accuracy. This might involve more sophisticated feature engineering, such as creating interaction terms between these variables.
2. **Handle Less Important Features**: Simplifying the model by removing less important features can reduce complexity, speeding up training times and lowering the risk of overfitting. These features either provide redundant information or have minimal impact on sales.

After training the Random Forest model, feature importance was analyzed to identify the most impactful features. Promotional features, store type, and the number of customers were among the top predictors of sales. The Random Forest model provided reasonable predictions, but I sought to improve them using a more advanced deep learning technique: LSTM

Task 2.6 Building an LSTM Deep Learning Model

Why LSTM?

Long Short-Term Memory (LSTM) networks are well-suited for time-series forecasting because they can capture long-term dependencies in the data, making them an ideal choice for predicting sales over time. Unlike traditional feedforward networks, LSTMs maintain a "memory" of previous time steps, enabling the model to better understand trends and seasonality in the data.

Steps for Building the LSTM Model:

1. **Stationarity Check**: Using the Augmented Dickey-Fuller (ADF) test, I confirmed that the sales time series data exhibited non-stationarity, which was expected due to seasonal sales patterns.
2. **Data Differencing**: To make the data stationary, I applied differencing techniques, focusing on year-over-year and month-over-month differencing to capture seasonal trends.
3. **Transforming Data into Supervised Learning**: Using a sliding window technique, I converted the time-series data into a format suitable for supervised learning, where past sales data served as input to predict future sales.
4. **Scaling**: The features were scaled to a range of (-1, 1) using MinMaxScaler for better performance with the LSTM model.

LSTM Model Architecture

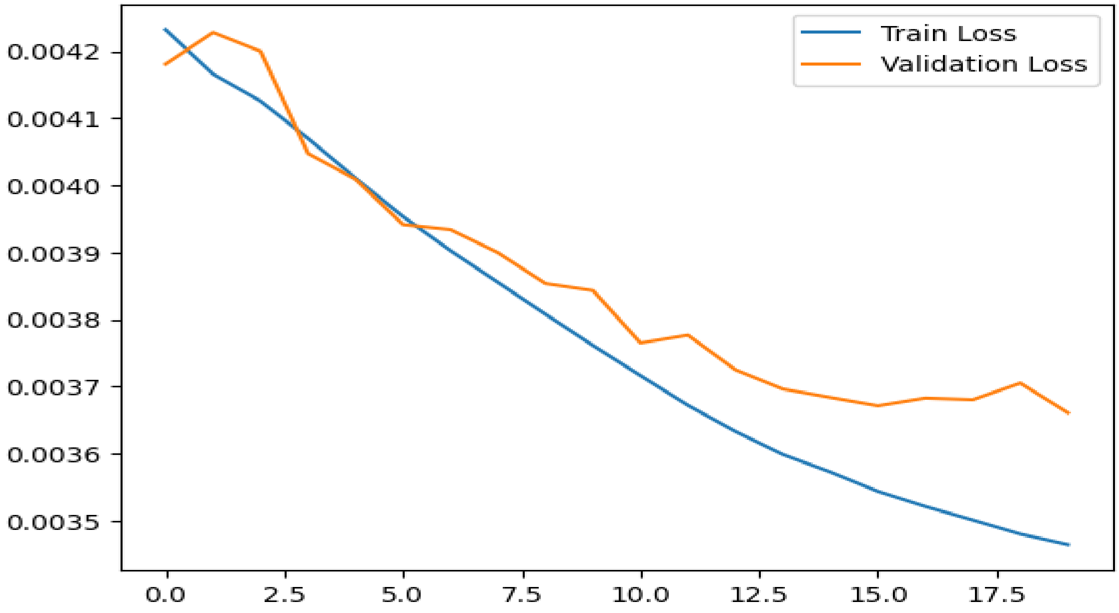
The LSTM model was designed with two layers to capture the time dependencies without making the model too deep, ensuring it could run efficiently on Google Colab.

```
from keras.models import Sequential

from keras.layers import LSTM, Dense


def build_lstm_model(n_timesteps, n_features):

    model = Sequential()

    model.add(LSTM(units=50, return_sequences=True, input_shape=(n_timesteps, n_features)))

    model.add(LSTM(units=50))

    model.add(Dense(1))  # Output layer for regression


    model.compile(optimizer='adam', loss='mean_squared_error')

    return model
```

- **Input Shape**: The input to the LSTM is a sequence of past sales data points over a time window.
- **Hidden Layers**: Two LSTM layers with 50 units each, allowing the model to capture long-term dependencies.
- **Output Layer**: A Dense layer that outputs the predicted sales for the next day.

Model Training and Results

The LSTM model is performing well, with both training and validation losses decreasing steadily. The model is learning efficiently without signs of overfitting. It might be possible to continue training or consider early stopping if the validation loss stabilizes further without significant improvement.



Insights from Training and Validation Loss Curves

- **Initial Loss Decrease (Epochs 0-4):** Both training and validation losses drop steadily, showing the model is learning effectively. The slightly higher validation loss is expected and indicates the model is fitting well to the training data.
- **Stabilization (After Epoch 4):** From epoch 4 onwards, the losses continue to decrease at a similar rate for both sets, showing that the model generalizes well without overfitting.
- **No Overfitting:** The close alignment of the loss curves suggests no overfitting, as validation loss does not increase while training loss decreases.
- **Convergence (Around Epoch 18):** The curves converge, indicating the model is nearing optimal performance, with diminishing improvements in loss reduction.

Conclusion and Recommendations

This project successfully implemented a sales forecasting model for Rossmann Pharmaceuticals using both traditional machine learning techniques and advanced deep learning models. The LSTM model provided superior performance for predicting daily sales across various stores, especially when capturing seasonal patterns and long-term dependencies.

Key Recommendations:

1. **Fine-tune the LSTM Model**: Hyperparameter tuning (e.g., learning rate, number of epochs) could further improve the model's performance.
2. **Feature Engineering**: Additional features such as weather data and online sales could enhance predictive accuracy.
3. **Model Deployment**: Serialize the trained model and integrate it into Rossmann's daily operations for real-time forecasting.

By leveraging both tree-based models and LSTM, Rossmann Pharmaceuticals can now more accurately forecast store sales and optimize inventory and promotions for maximum efficiency.