Stock Price Prediction with LSTM

Learn Everything AI

June 2, 2023

1 Introduction

Welcome to this hands-on guide where we will solve the stock price prediction problem using the LSTM algorithm. In this project, we will analyze a dataset containing historical stock prices and build a model to predict future stock prices accurately. So, let's dive into the exciting world of time series analysis and machine learning!

2 Data Preprocessing

Before we can train our LSTM model, we need to preprocess the data. This step involves handling missing values, scaling the data, and splitting it into training and testing sets.

2.1 Handling Missing Values

We'll explore different techniques to handle missing values, such as imputation and removal. Let's assume we have loaded the dataset into a DataFrame named `data`. We can handle missing values using the following Python code:

```
import pandas as pd

# Drop rows with missing values
data = data.dropna()
```

2.2 Scaling the Data

To ensure that all features have similar scales, we'll apply feature scaling. We can use the MinMaxScaler from the scikit-learn library to scale the data:

```
from sklearn.preprocessing import MinMaxScaler

# Create a scaler object
scaler = MinMaxScaler()
```

```
# Scale the data
scaled_data = scaler.fit_transform(data)
```

2.3 Splitting the Data

To evaluate the performance of our model, we need to split the dataset into a training set and a testing set. We can split the data using the following Python code:

```
import numpy as np

# Split the data into features and target
X = scaled_data[:, :-1]  # Input features
y = scaled_data[:, -1]   # Target variable

# Split into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

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

3 Model Training and Evaluation

With the preprocessed data in hand, we can now train an LSTM model and evaluate its performance using appropriate metrics.

3.1 Training the Model

We'll use the Keras library to build and train the LSTM model. Here's the Python code:

```
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Create the LSTM model
model = Sequential()
model.add(LSTM(units=50, input_shape=(X_train.shape[1], 1)))
model.add(Dense(units=1))

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

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

3.2 Model Evaluation

To evaluate the trained LSTM model, we can calculate various evaluation metrics and visualize the results. Here's the continuation of the LaTeX code:

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Inverse scale the predictions and actual values
y_pred = scaler.inverse_transform(y_pred)
y_test = scaler.inverse_transform([y_test])

# Calculate evaluation metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

mse = mean_squared_error(y_test[0], y_pred[:, 0])
mae = mean_absolute_error(y_test[0], y_pred[:, 0])
r2 = r2_score(y_test[0], y_pred[:, 0])

# Print evaluation metrics
print('Mean Squared Error (MSE):', mse)
print('Mean Absolute Error (MAE):', mae)
print('R-squared Value:', r2)
```

We have calculated the Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared value as evaluation metrics for the model.

To visualize the results, we can create a line plot of the actual stock prices and the predicted stock prices using matplotlib:

```
import matplotlib.pyplot as plt

# Plotting the actual and predicted prices
plt.plot(y_test[0], color='blue', label='Actual Price')
plt.plot(y_pred[:, 0], color='red', label='Predicted Price')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.title('Actual vs Predicted Stock Prices')
plt.legend()
plt.show()
```

This will generate a line plot showing the actual and predicted stock prices over time.

Feel free to customize the code, add more evaluation metrics or visualizations, and adjust the formatting to meet your specific requirements.

4 Summary

In this project, we solved the stock price prediction problem using the LSTM algorithm. We preprocessed the data by handling missing values and scaling the features. Then, we split the data into training and testing sets and trained an LSTM model to predict future stock prices. We evaluated the model's performance using evaluation metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared value. We also visualized the actual and predicted stock prices over time. This project demonstrates the application of LSTM in predicting stock prices and provides insights into the field of time series analysis and machine learning.