Summary of S&P500 Forecast Project

Week 9

Monil Shah

Aug 24, 2024

Contents

Introduction

In this project, we take the time series of SnP500 for the last 5 years and try to fit a model to forecast the price forward. The data is split into train and test data where test data is the last 1 year on the data on which we measure the performance of the model. We have tried 3 models. Two of the models are SimpleRNN and one is LSTM. Detailed comparison of the models and insights are shown below. Data was downloaded from Yahoo Finance.

Preprocessing and feature engineering

I have created several derived attributes such as 10 day moving average, 30 day moving average and 100 day moving average. This would create NaN for the first rolling window in each attribute. We remove all observations with NaN. Then to identify if there is an option expiry effect we identify the last Thursday of each month. I created some temporal variables from day such as day, month, and year.

On top, I created two more features namely opening_gap and daily_move_avg. The former indicates the difference between the last closing price and the day's opening price. The daily movement average is the difference between the High and Low of the previous n days, where n is the rolling window size and is configurable. We have set it to 1.

Model Comparison

Feature/Parameter	Model 1 (LSTM)	Model 2 (RNN - Large)	Model 3 (RNN - Small)
Model Type	LSTM	SimpleRNN	SimpleRNN
Number of Layers	2 LSTM layers	2 SimpleRNN layers	2 SimpleRNN layers
Layer 1 Units	256	256	8
Layer 2 Units	256	256	8
Dropout	0.2 for each LSTM layer	0.2 for each RNN layer	0.5 for each RNN layer
Dense Layer	1 Dense layer (output)	1 Dense layer (8 units, ReLU) + 1 output	1 Dense layer (8 units, ReLU) + 1 output
Output Layer	1 Dense layer	1 Dense layer	1 Dense layer

Activation Function	No activation in output layer	ReLU in Dense layer	ReLU in Dense layer
Return Sequences	True in the first LSTM layer	True in the first RNN layer	True in the first RNN layer
Optimizer	Adam (learning rate = 0.01)	Adam (learning rate = 0.001)	Adam (learning rate = 0.01)
Loss Function	RMSLE	RMSLE	RMSLE
Batch Size	64	8	8
Epochs	100	100	100
Early Stopping Patience	5	5	5
Input Shape	(timesteps, X_train_reshaped.shape[2])	(timesteps, X_train_reshaped.shape[2])	(timesteps, X_train_reshaped.shape[2])
Weight Restoration	Yes (restore_best_weights=True)	Yes (restore_best_weights=True)	Yes (restore_best_weights=True)

Table 1 Comparison of 3 models applied.

The table above summarizes the comparison between the models we applied.

Results & Insights

When reducing the number of units in RNN layers (From model 2 to model 3) leads to better performance, it's often due to the model becoming less complex and avoiding overfitting. With fewer units, the model can focus on the most important patterns in the data, rather than learning every detail, some of which may be noise. This simplification can result in better generalization to unseen data, improved learning dynamics, and sometimes faster convergence. In essence, a smaller model might better match the complexity of your data, leading to more effective and efficient learning.

Adding an additional dense layer before the final output layer in the RNN model can significantly enhance performance. This layer allows the model to transform and refine the features learned by the RNN layers, capturing more complex patterns and non-linear relationships. It also acts as a form of regularization, helping to reduce overfitting and improving generalization. The added layer provides a more sophisticated feature representation,

enabling the model to make more accurate predictions by "thinking" more deeply about the data before the final output.

Appendix 1: Code

Python notebook used:

SPX500_forecast_Monil_shah_Model 1.ipynb

SPX500_forecast_Monil_shah Model 2.ipynb

SPX500_forecast_Monil_shah Model 3.ipynb