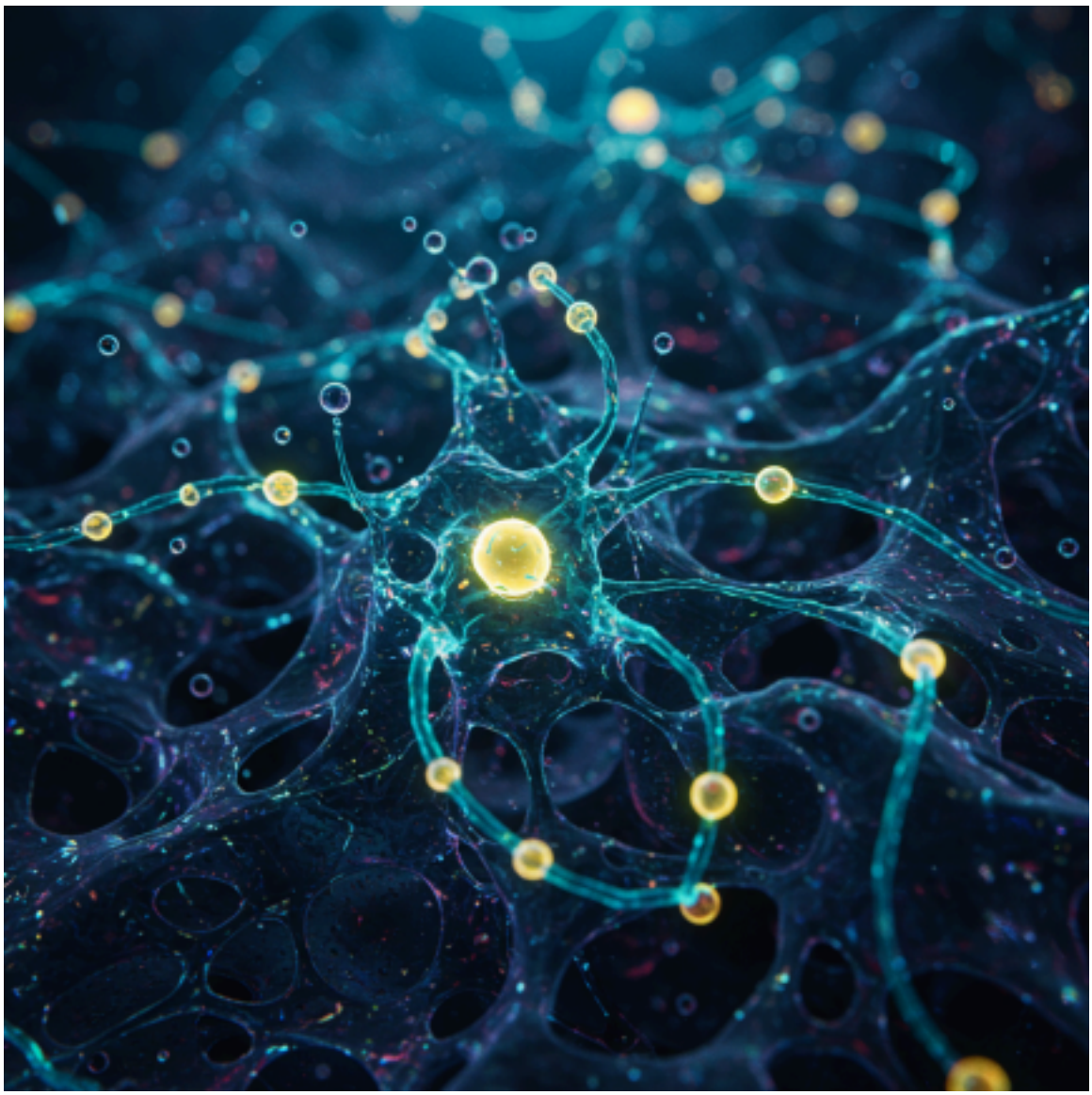


Assignment #4

PROJECT 4: Time Series Forecasting Using FCNN, CNN and RNN

Paul Felker | CSC-180-01 | 11/10/25
Benjamin Church | CSC-180-01 | 11/10/25

Problem Statement

This assignment takes the RDS-B close of a day based on the data for the 7 days prior. This model inputs models Open, High, Low, Volume and Close. This data is chronological, meaning we are able to take the first 7 days worth of data and use it to predict the close on the 8th day. This is accomplished in 3 ways. The first is to acquire the values and flatten them into 1x35 size datasets to be used in a FCNN. The second is to form it into a 2D image of 7x5 (7 Height, 5 width, 1 channel) which is given to a CNN.

The third way is to use an LSTM model capable of learning from 7 vectors each with 5 dimensions. Additionally, a comparison has been implemented for the same model using 5 different window frame sizes (3,5,7,9,11) with the intent of finding the most effective range of values.

Methodology

Data Cleaning

The first steps were somewhat simple, reading in the dataset, dropping unneeded columns (Date, Adj, Close), reordering in the format of [Open, High, Low, Volume, Close], dropping rows with NaN values, making a copy of Close to not be normalized, resetting the indexes, splitting (without shuffle) into df_train and df_test, and normalizing with encode_numeric_zscore. The big difficulty was in creating the sliding window protocol. This was accomplished with a for loop, which per index retrieves all values that would go into x, flattens them and gets the matching y from the unnormalized copy of close. This data is then shaped accordingly and then distributed to each of the models.

Model Approach

The FCNN is composed of a series of dense layers that lessen in width with each layer. Additionally, a series of dropout layers are placed in between the FCNN in order to prevent overfitting. It should be noted that during training, for both the FCNN and the CNN, shuffling was used during fitting in order to improve performance on the dataset while maintaining consistency of each individual window.

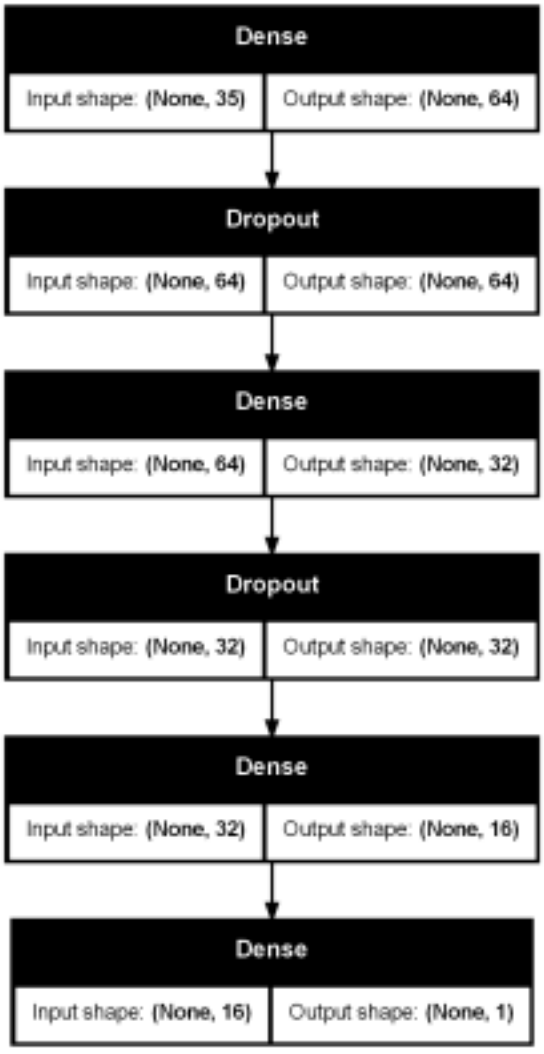
The CNN layer uses a few Convolution layers as well as a max pooling and dropout layer. Flattening is then used in combination with an additional dense layer before being outputted.

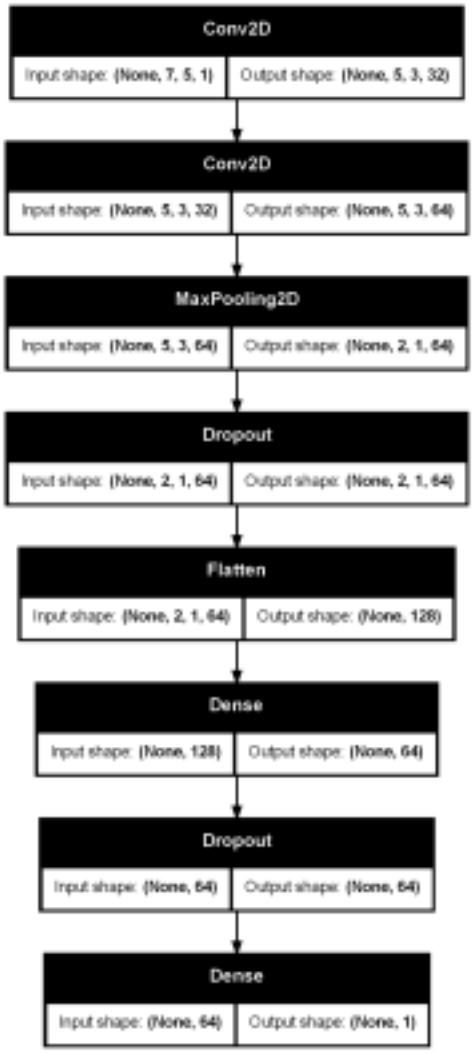
The RNN implementation utilizes two LSTM layers as well as bi-directional functionality in order to retain the context of the sequences ahead of it and behind it.

Additionally, a single dropout is used to reduce overfitting as well as a single additional dense layer before an output layer. Uses tanh activation for the LSTM layer, and then relu for the dense layer, as well as an Adam optimization.

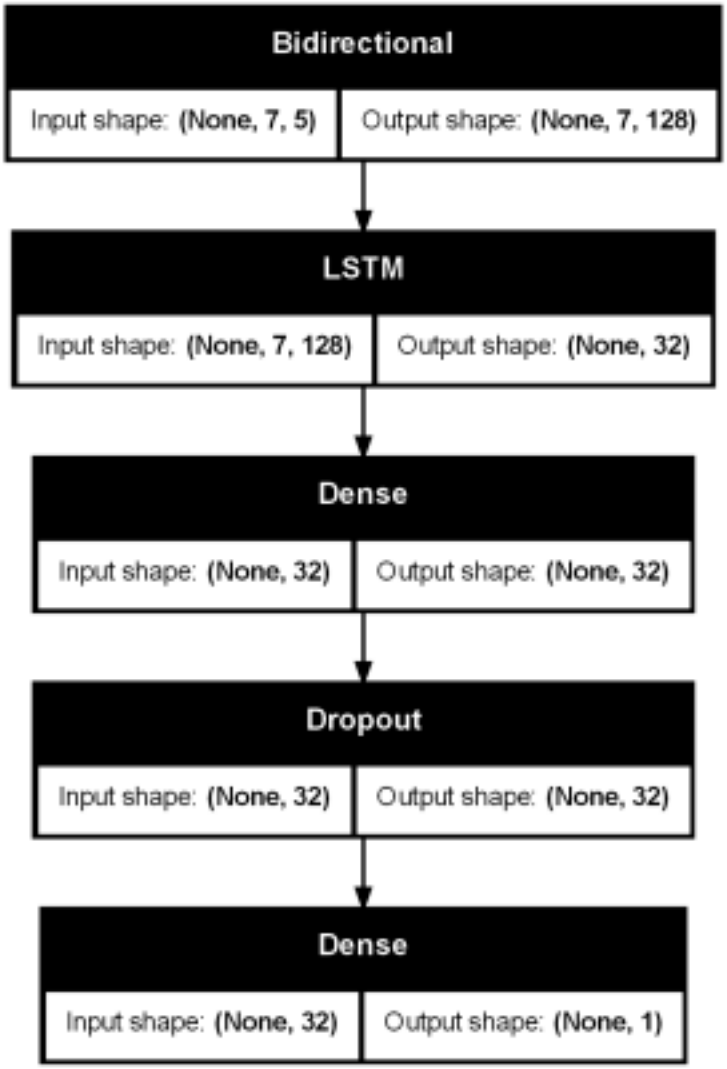
Model Representation

FCNN CNN





LSTM



Experimental Results and Analysis

Highlighted cell in each column denotes best performing model of their respective type

Comparison of RMSE of Different Models (Relu)

Dollars	Run 1 (RMSE)	Run 2 (RMSE)	Run 3 (RMSE)	Run Avg (RMSE)
FCNN	7.78	7.79	10.26	9.89
CNN	5.74	4.33	5.39	5.15
LSTM	2.82	3.43	5.01	4.95

Comparison of RMSE of Different Models (Sigmoid)

Dollars	Run 1 (RMSE)	Run 2 (RMSE)	Run 3 (RMSE)	Run Avg (RMSE)
FCNN	8.23	8.88	10.26	5.91
CNN	5.23	5.13	4.98	5.11
LSTM	7.25	7.51	6.51	7.09

Comparison of RMSE of Different Models (Tanh)

Dollars	Run 1 (RMSE)	Run 2 (RMSE)	Run 3 (RMSE)	Run Avg (RMSE)
FCNN	8.33	8.15	9.01	5.27
CNN	5.10	7.57	6.09	6.25
LSTM	7.26	7.22	7.52	7.33

Comparison of RMSE of differing Optimizers (SGD & Relu)

Dollars	Run 1 (RMSE)	Run 2 (RMSE)	Run 3 (RMSE)	Run Avg (RMSE)
FCNN	9.70	8.51	4.11	8.67
CNN	3.93	3.98	2.61	3.51
LSTM	6.59	6.75	7.21	6.85

Comparison of Results Based on Size of Frame (In Days) (Best of 10 Runs)

	Run 1 (RMSE)	Run 2 (RMSE)	Run 3 (RMSE)	Run Avg (RMSE)

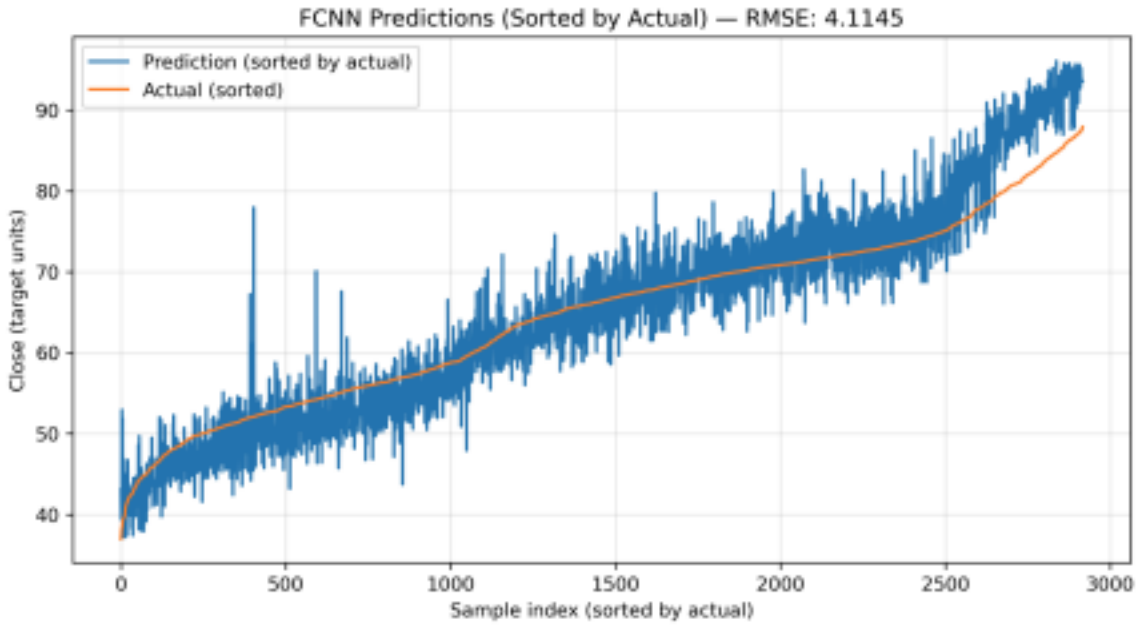
3 Days	**7.51**	**7.47**	**7.75**	**6.41**
5 Days	**7.48**	**7.98**	**7.44**	**7.36**
7 Days	**7.69**	**7.57**	**7.36**	**6.58**
9 Days	**4.93**	**7.71**	**8.05**	**5.86**
11 Days	**8.02**	**7.73**	**8.20**	**7.16**

Comparison of Result Based on Depth and Width*

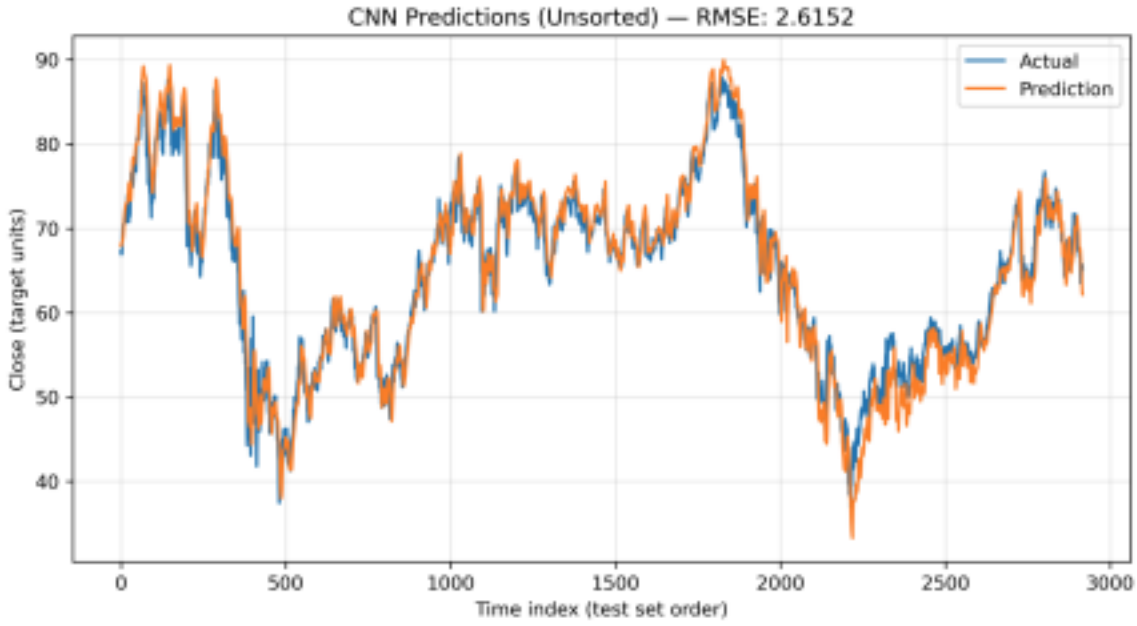
Dollars	**Iteration-1**	**Iteration-2**
FCNN	**11.29**	**8.26**
CNN	**2.74**	**12.18**
LSTM	**7.22**	**6.42**

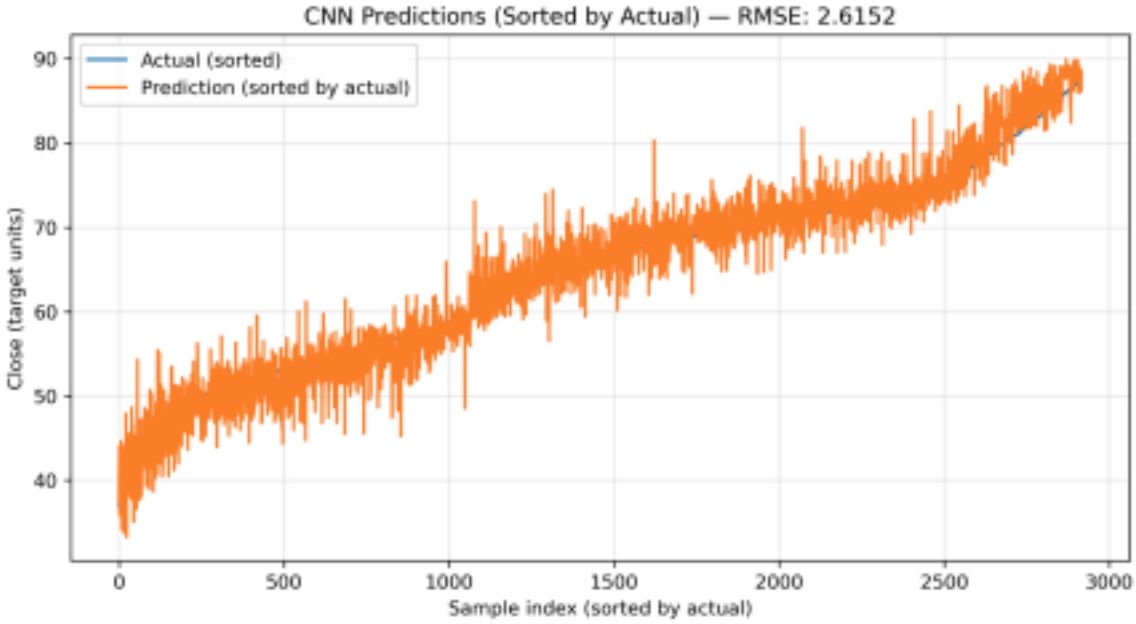
*Iteration-2 utilized 4 hidden layers of Convolution, Iteration-1 utilized 3 hidden layers (not accounting for dropouts and pooling). Iteration-2 utilized a max width of [128, 64, 32, 16], Iteration-1 max width of [64, 32, 16].

Lift Chart of Best Performing FCNN

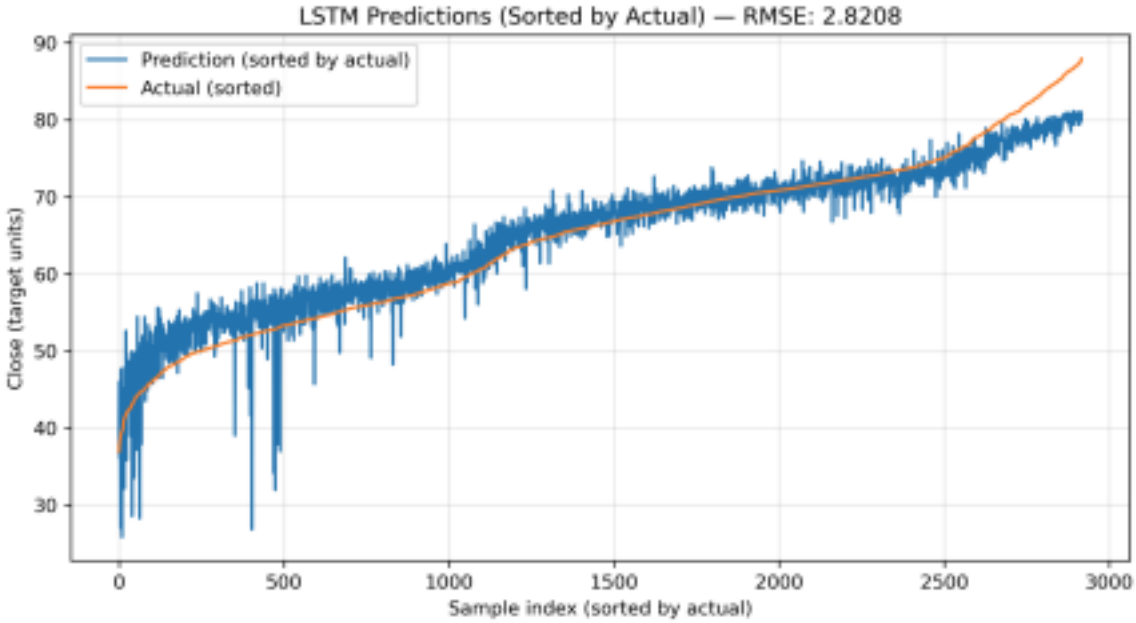


Lift Chart of Best Performing CNN

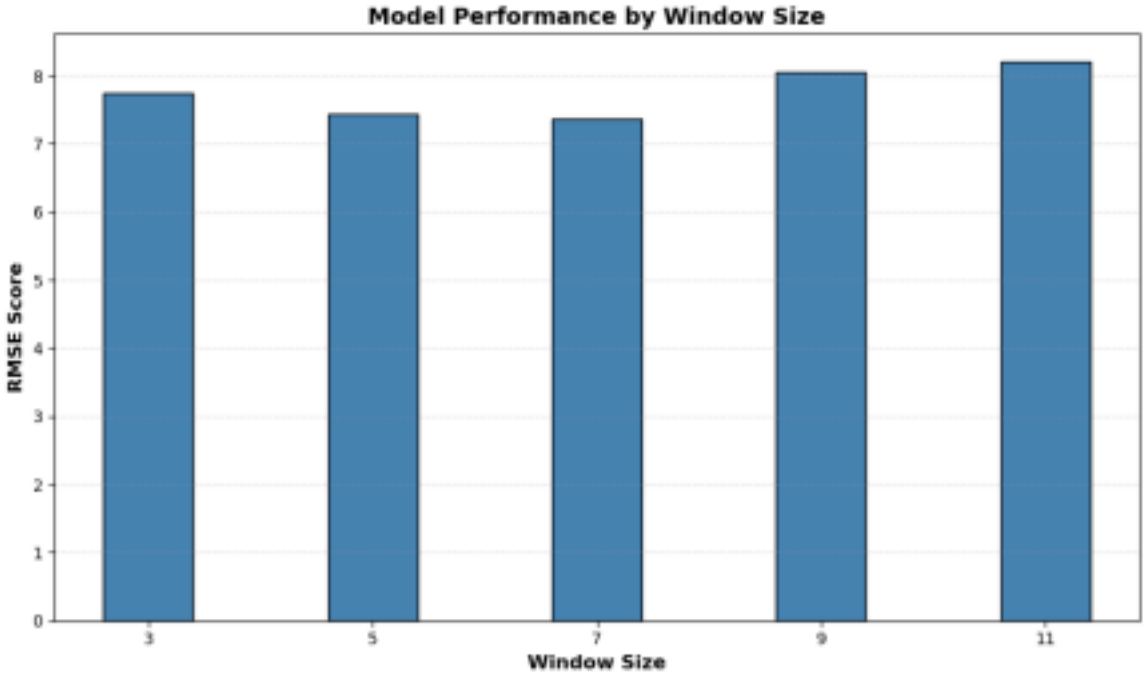




Lift Chart of Best Performing LSTM



Performance Bar Graph of RMSE Vs. N Window size of FCNN*



*Single RMSE sample, from left to right, 2.04, 7.47, 7.3, 7.35, 5.63

Comparative Analysis between FCNN, CNN & LSTM

Of the three models, the CNN was found to perform marginally best with an RMSE of 2.61 using a combination of SGD and Relu. LSTM performed second best once employing a bi-directional layer followed by a Dense layer with an RMSE of 2.82. Lastly FCNN struggled the most having occasional large outliers in the dataset, resulting in a RMSE of 4.11. It should also be noted that performance was largely contingent on the dataset used, as other datasets rendered differing performance. Additionally, upon adjusting to various other window sizes, the FCNN was found to on average favor a smaller window size of 3 days, possibly allowing it to more easily predict price based off of very recent data, ultimately making it less valuable for this type of prediction.

Project Reflection

Challenges Encountered

One large challenge faced was determining how to best prepare the data. The frame approach of getting a large chunk of data and comparing it to data further in time was a new problem and we had a tough time determining how to approach it. Additionally, realizing we would need to separate some values to prevent the outputs from being normalized and still needing to map them properly was difficult for me to wrap my head

around. The end approach was to create a copy of the Close column, and create a special sliding window protocol that would loop through the data, append all data values to a list within range i-1 and i-7, and then grab the close value from the copied column at i.

Another major issue was realizing that the randomized splitting of the data was not in accordance with the prompt. Originally, we had run the sliding window protocol to get the proper data in order, but then randomized it in the train/test split. While this did perform better, it took a while to realize that this is improper in relation to the prompt of the assignment, so it was changed.

Learning experience

Lots of time was invested in working out exactly how to prepare the data properly. The sliding window protocol was extremely fun and interesting to figure out how to implement, as well as working with all of the provided datasets and finding out which ones lent themselves best to our implementation of the model. The LSTM struggled with our first dataset APPL, but after switching and trying out other datasets, we were able to determine that RDS-B was shaped in a way that was best for LSTM learning.