“Neural Networks”

Shervin Halat

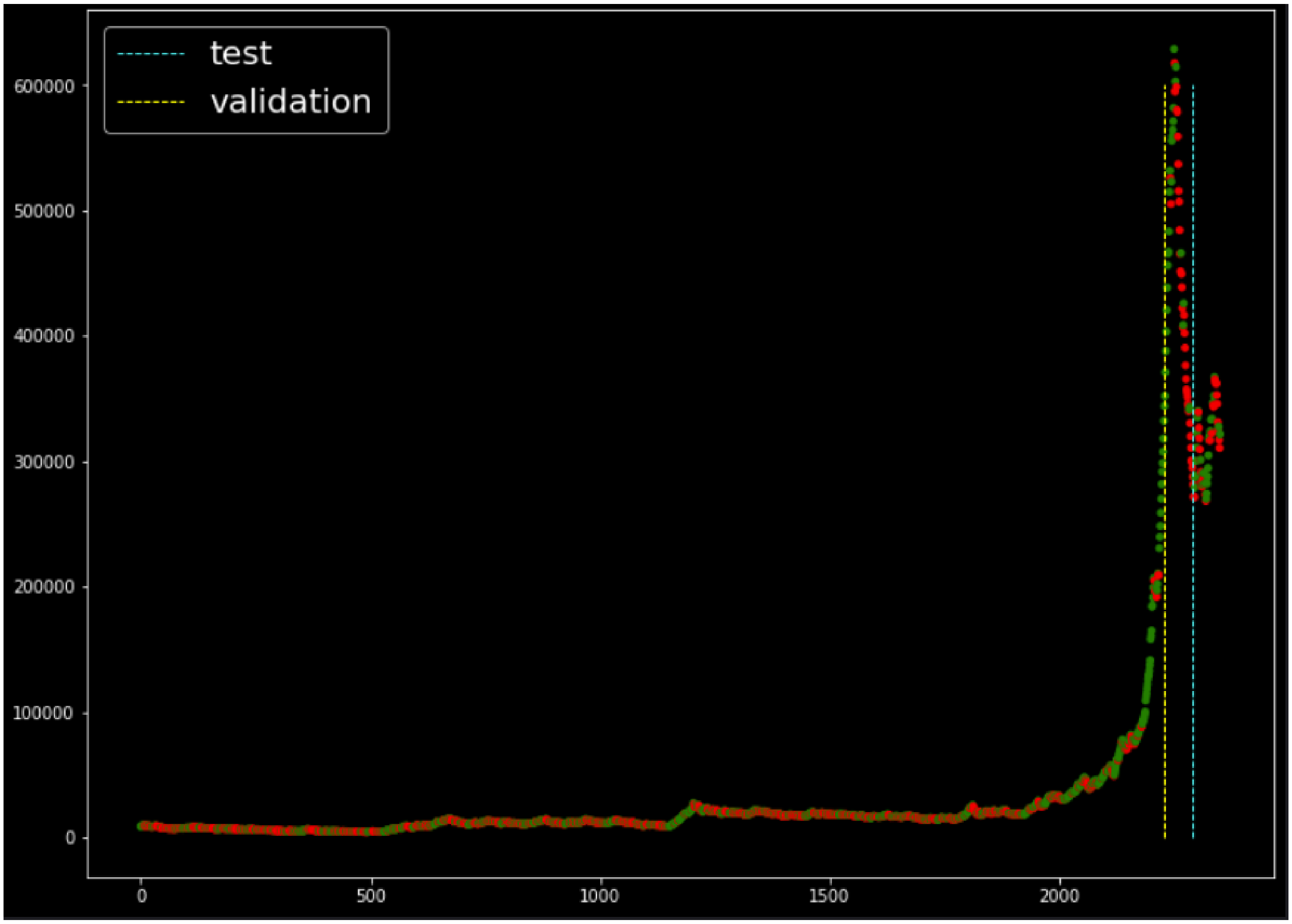
98131018

Homework 4

1.

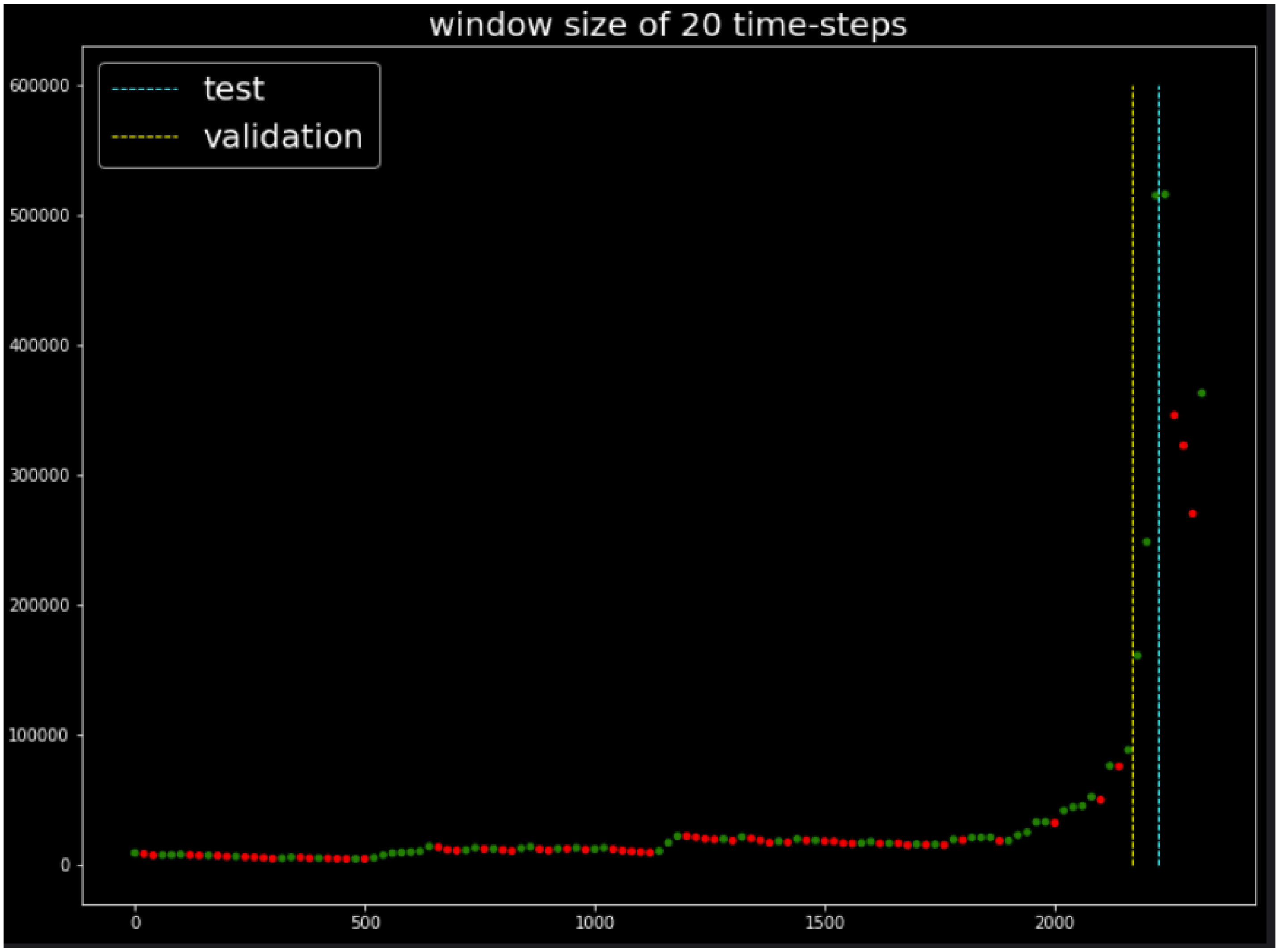
Final price of the stock index versus time is plotted below:

(It should be noted that both portions of test and validation dataset are specified by vertical dashed-lines with ‘cyan’ and ‘yellow’ colors respectively.)



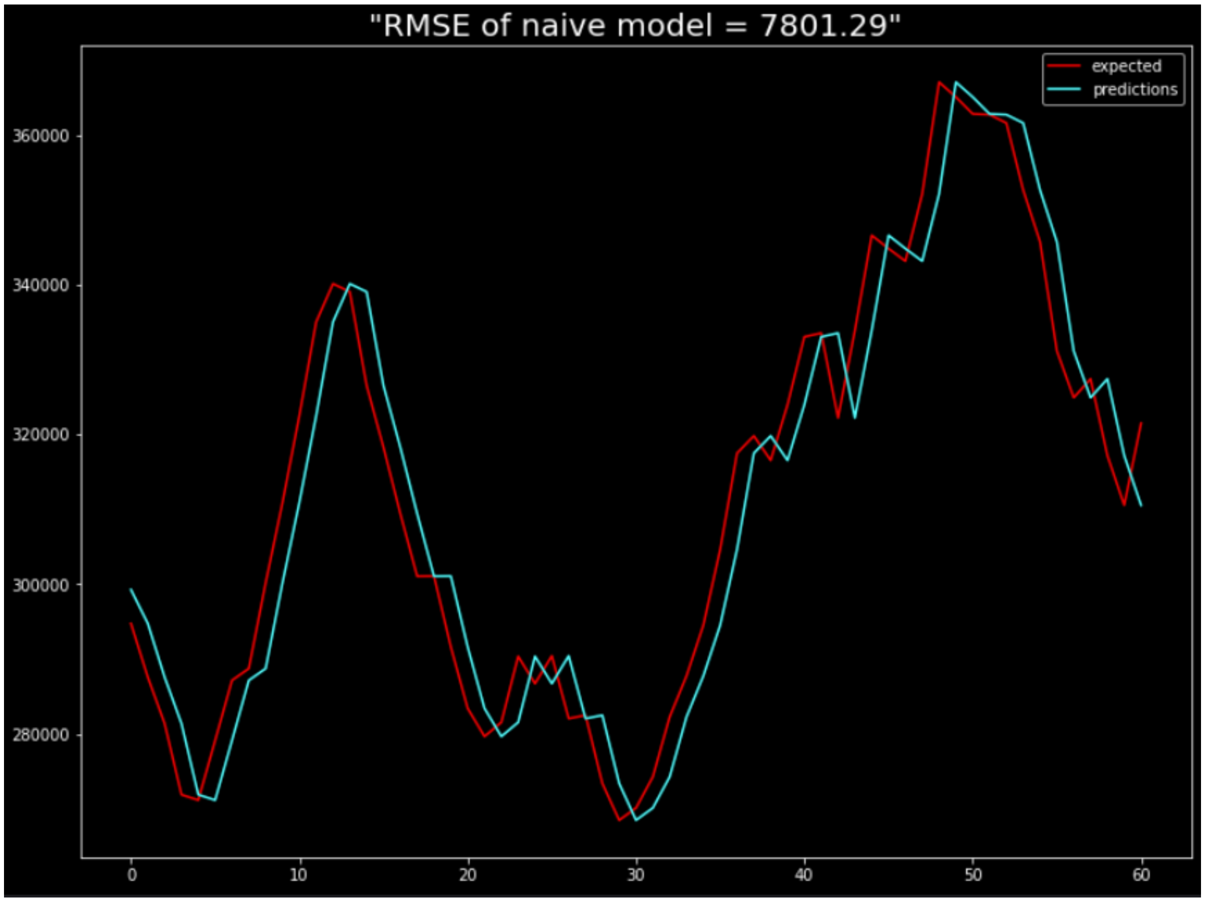
2.

Considering a window size of 20 timesteps, the figure above changes to the figure below:



3.

At first, we define a naïve model (which is also called ‘Persistence Model’) to evaluate if our defined LSTM model works well enough and can extract acceptable features for forecasting. The naïve model considers the next time step’s prediction equal to the last observation (value(t+1) = value(t)). Therefore, we expect our LSTM model to outperform this naïve model (‘Persistence Model’); in other words, we expect a lower RMSE loss of LSTM model compared to the naïve model. In the following, the naïve models’ predictions on test dataset is as follows:

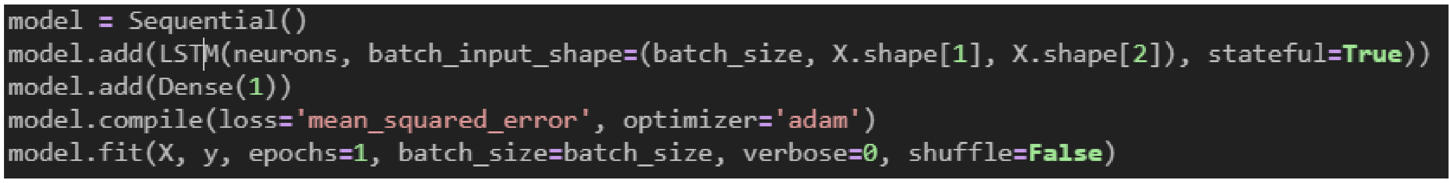


As we can see the RMSE loss is about 7809.

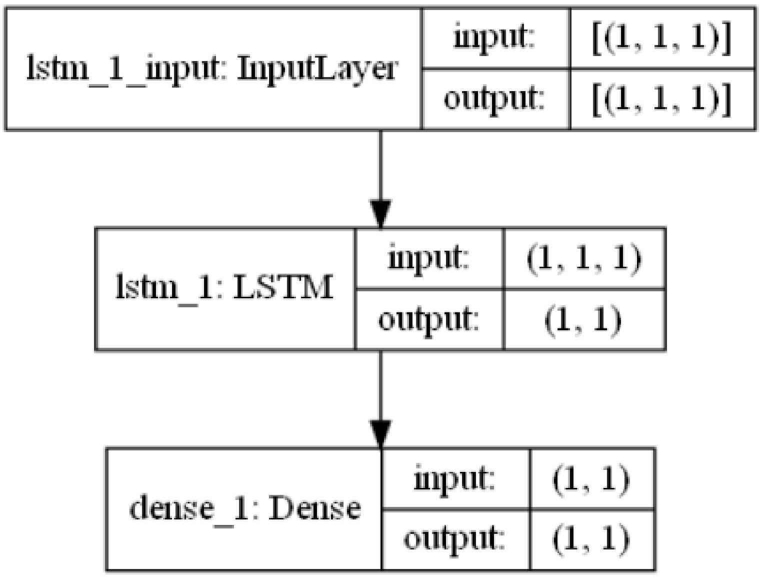
Next, we define a basic LSTM model by the help of which we are going to find the best configuration of our LSTM model. By the best model, we mean a model which results in the least RMSE loss on the test dataset.

The basic models' configuration is as follows:

> The basic model has one LSTM layer consisting single LSTM unit. Also, MSE loss function is used along with 'adam' optimizer.

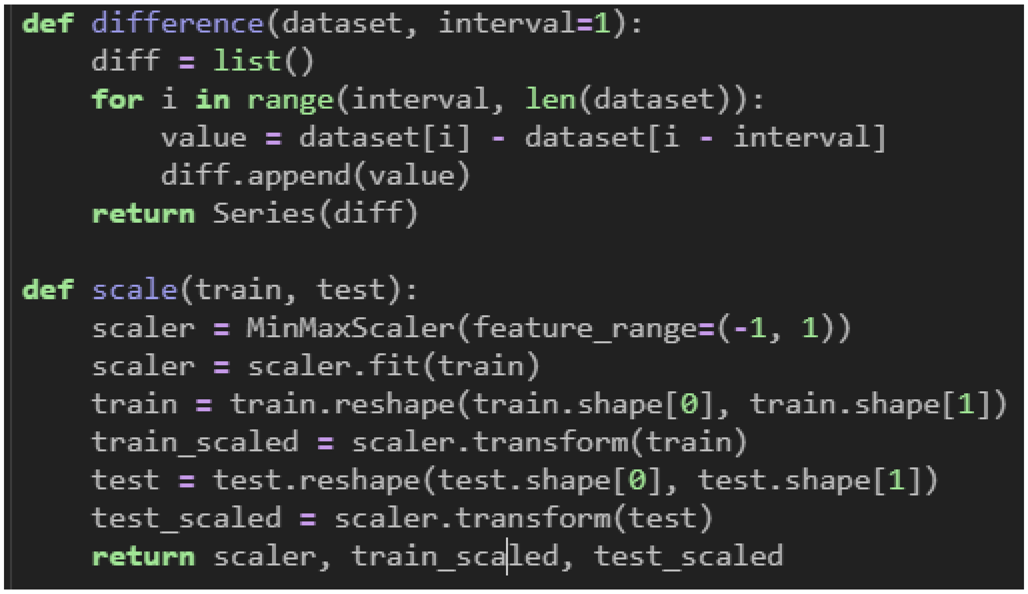


```
model = Sequential()
model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=True))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X, y, epochs=1, batch_size=batch_size, verbose=0, shuffle=False)
```



Also, it should be noted that all timesteps' values in the dataset are subtracted from their next timestep values in order to generate a more efficient dataset called stationary; this way, it seems that a bias in favor of the probable trends with respect to time will be removed; subsequently, a more robust and stable model will be generated. Next, all values will be scaled to the range of [-1,1] since default LSTM models' outputs are in the mentioned range because of 'tanh' activation function for the outputs.

These data preparations are defined in the following functions:



```
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return Series(diff)

def scale(train, test):
    scaler = MinMaxScaler(feature_range=(-1, 1))
    scaler = scaler.fit(train)
    train = train.reshape(train.shape[0], train.shape[1])
    train_scaled = scaler.transform(train)
    test = test.reshape(test.shape[0], test.shape[1])
    test_scaled = scaler.transform(test)
    return scaler, train_scaled, test_scaled
```

Also, it should be noted that these transformations in an exactly reverse procedure will be applied on the obtained outputs for them to be comparable to original datasets.

Now we go to the experiments part:

Pay attention that this problem is divided into two parts consisting:

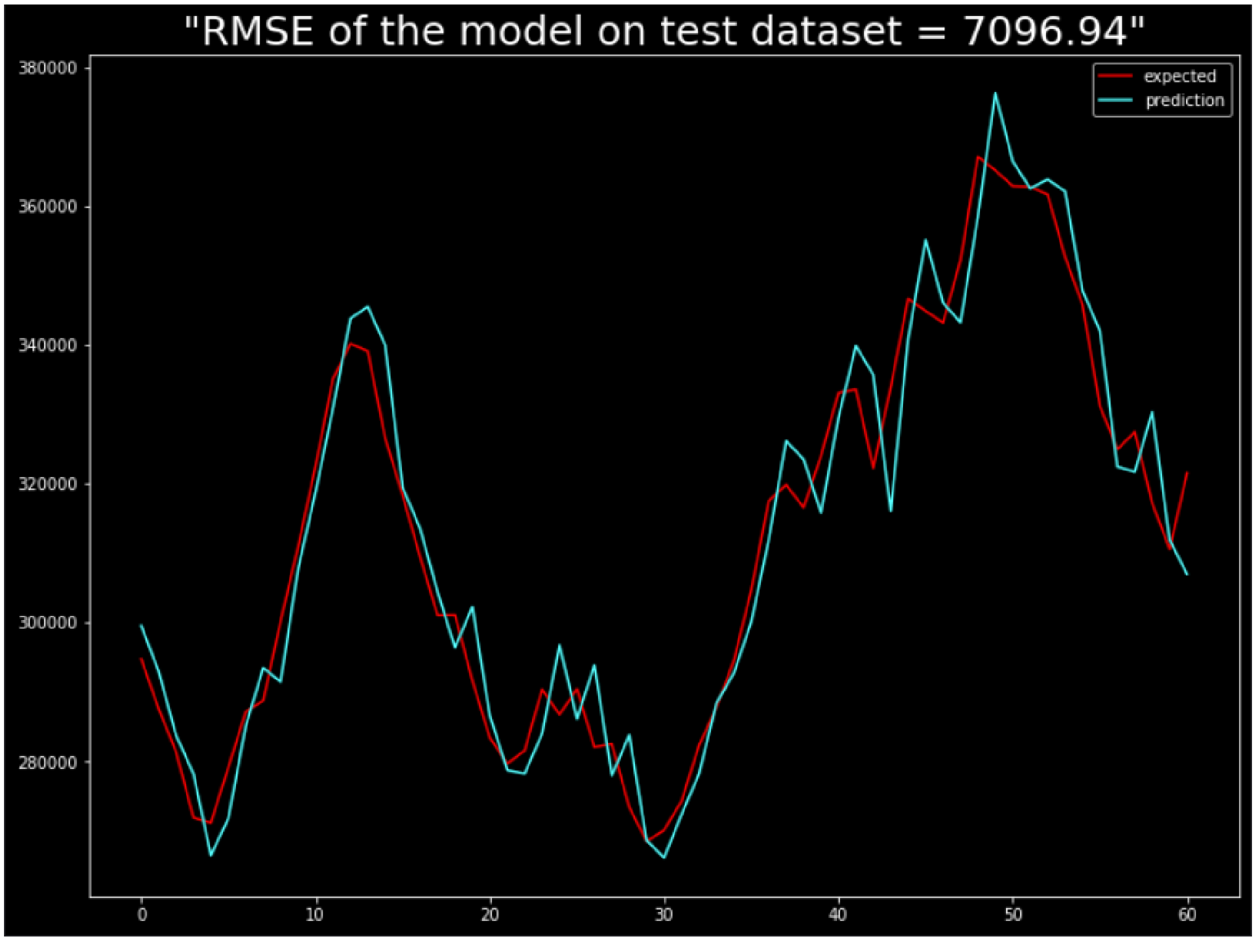
a. regular dataset

b. windowed dataset by a window of 20 timesteps.

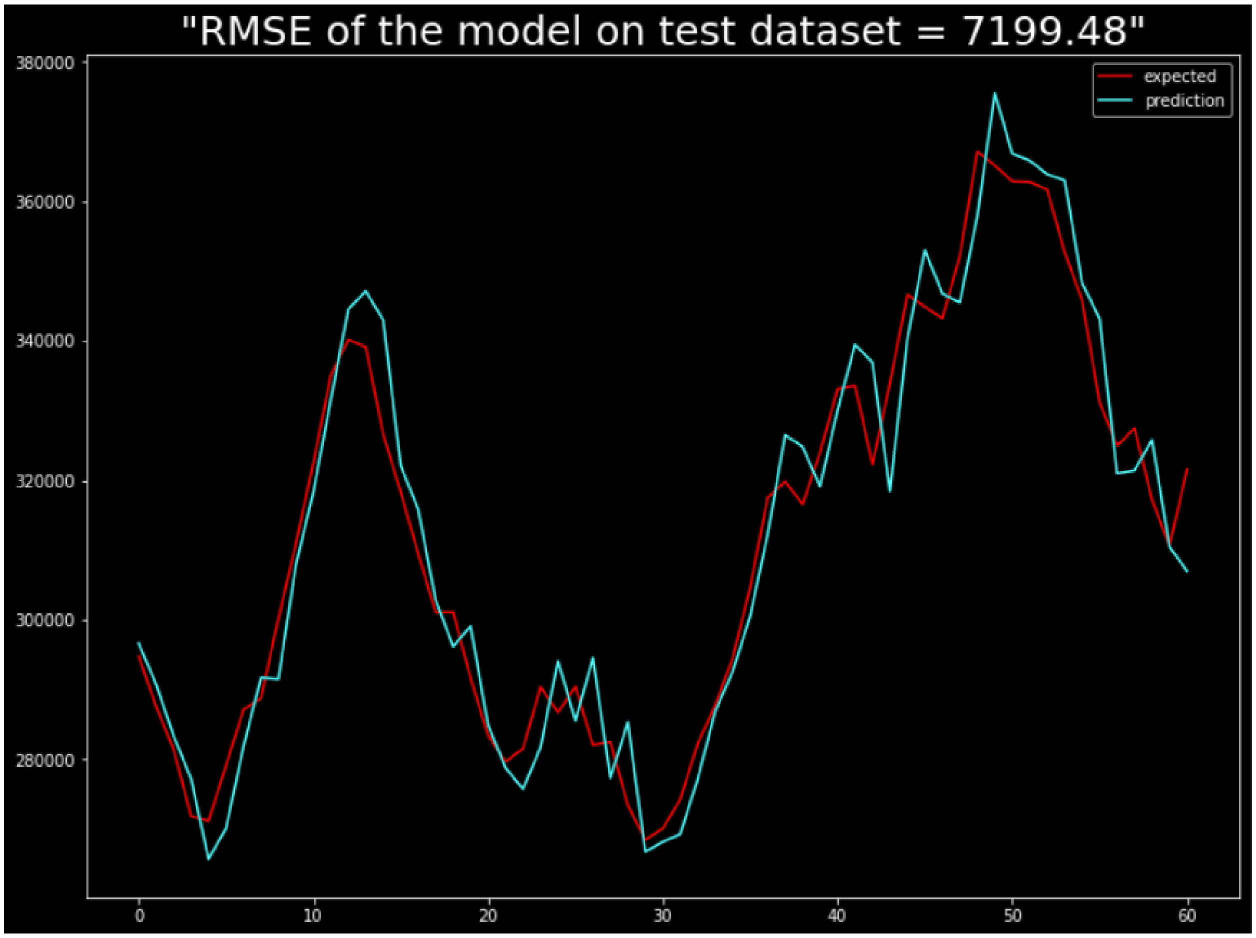
In order to find the optimal parameters of the LSTM model, multiple experiments on number of LSTM layers along with number of nodes (blocks or units) of each LSTM layer were conducted **(each experiment is repeated 2 times and an average of all obtained RMSE errors was computed)**. All models are trained with 25 number of epochs. Experiments are as follows:

a. (regular dataset)

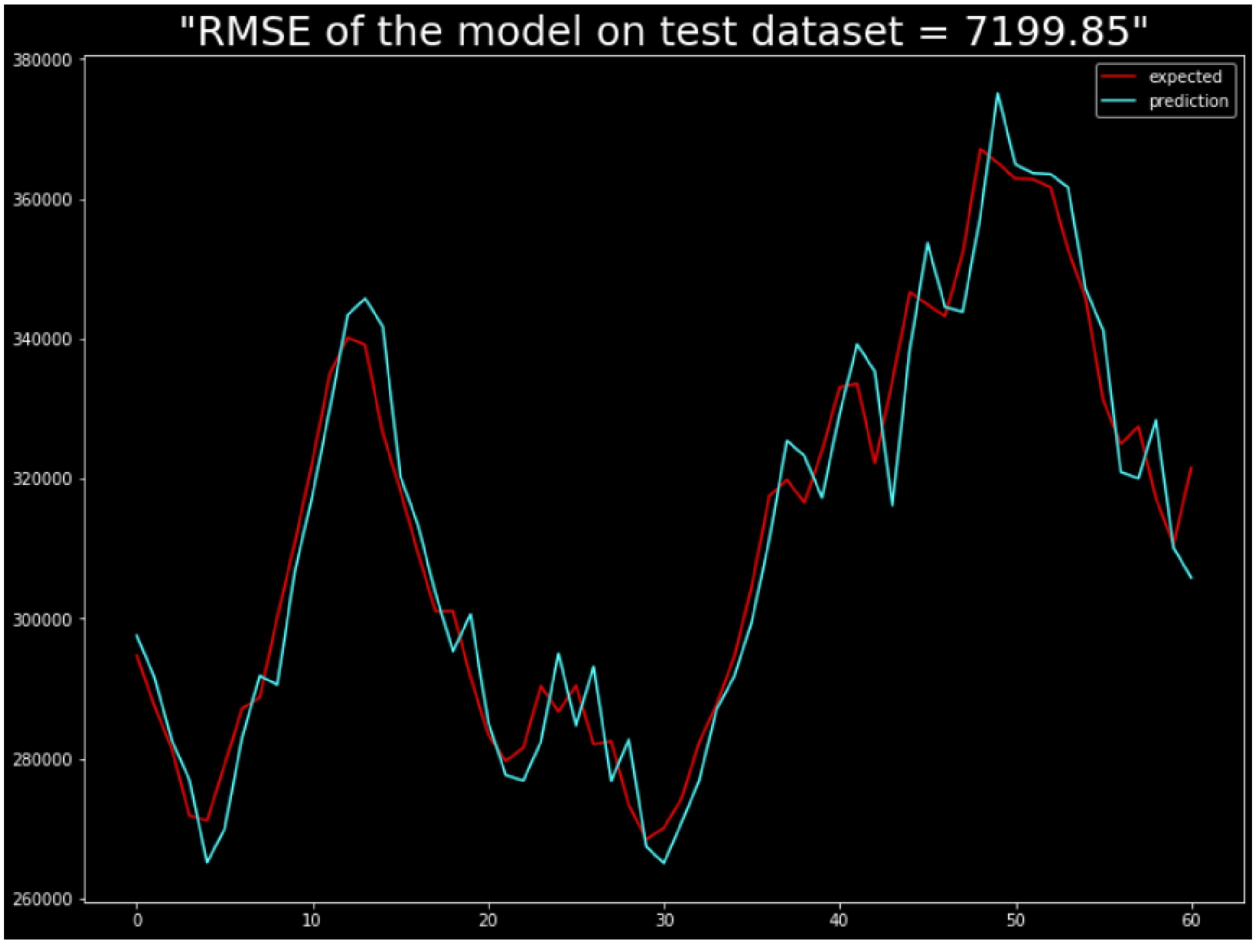
‘1 LSTM layer with 1 unit (basic model)’



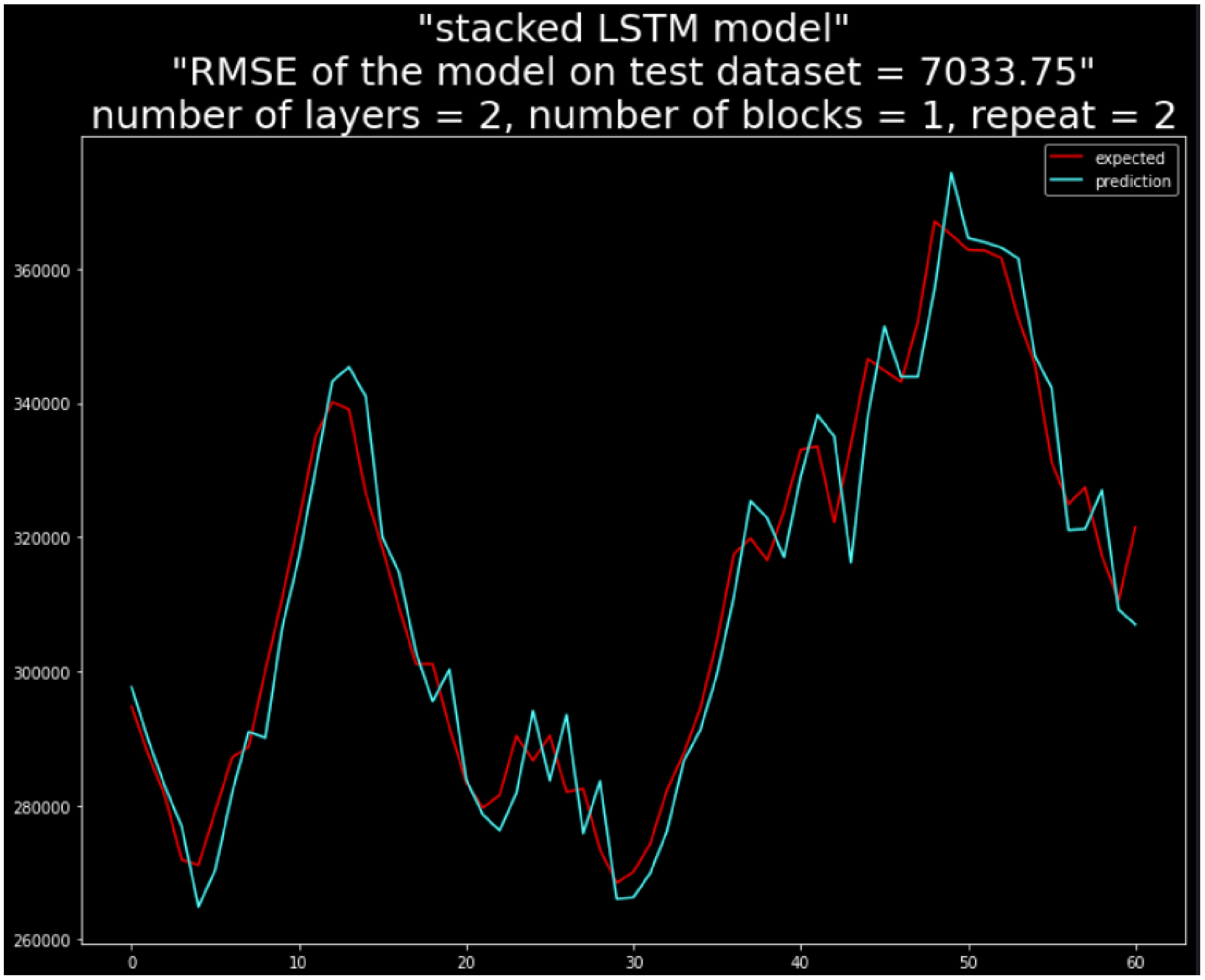
‘1 LSTM layer with 3 units’



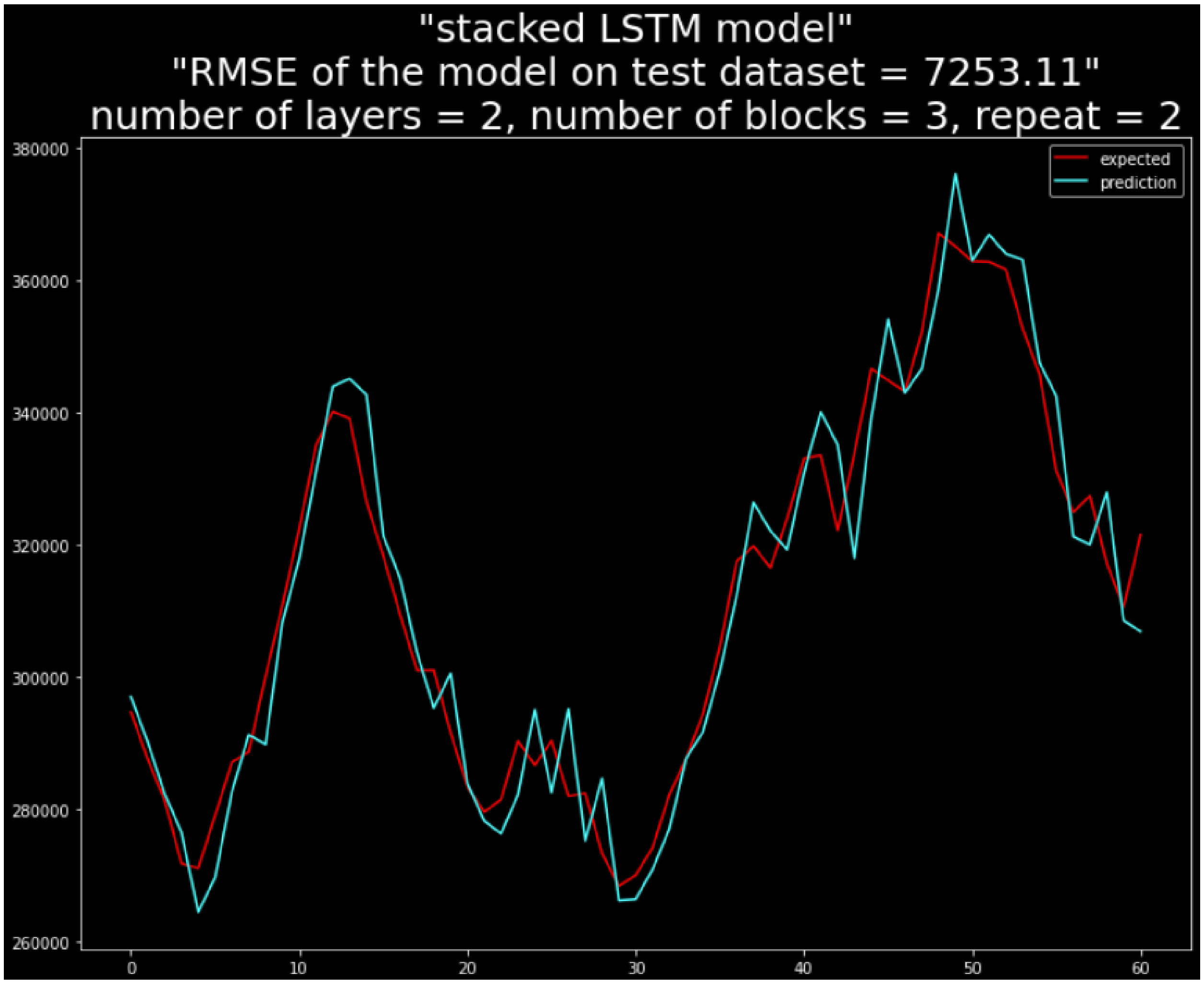
‘1 LSTM layer with 5 units’



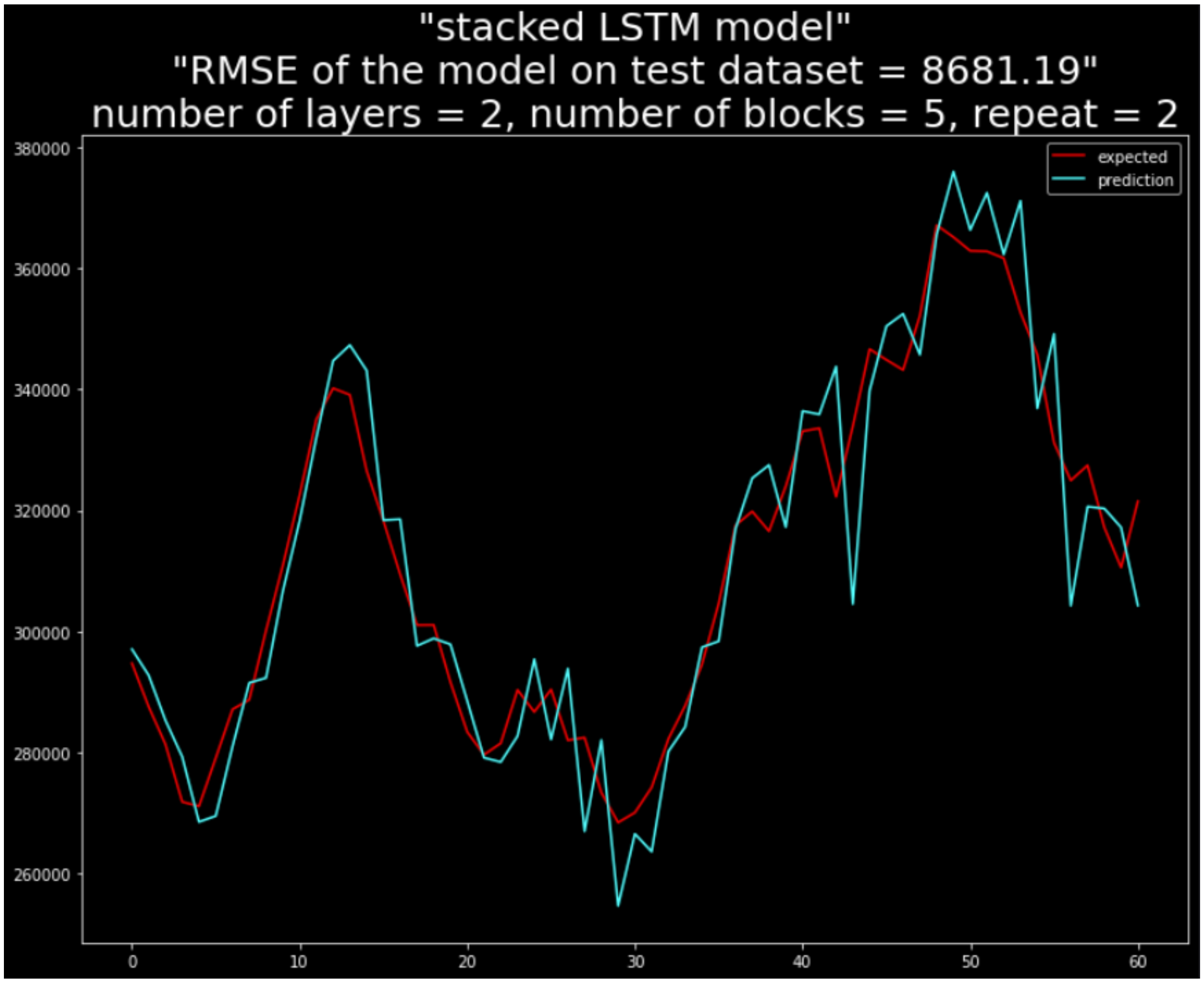
‘2 LSTM layers with 1 unit (Stacked LSTM Model)’



‘2 LSTM layers with 3 units each (Stacked LSTM Model)’

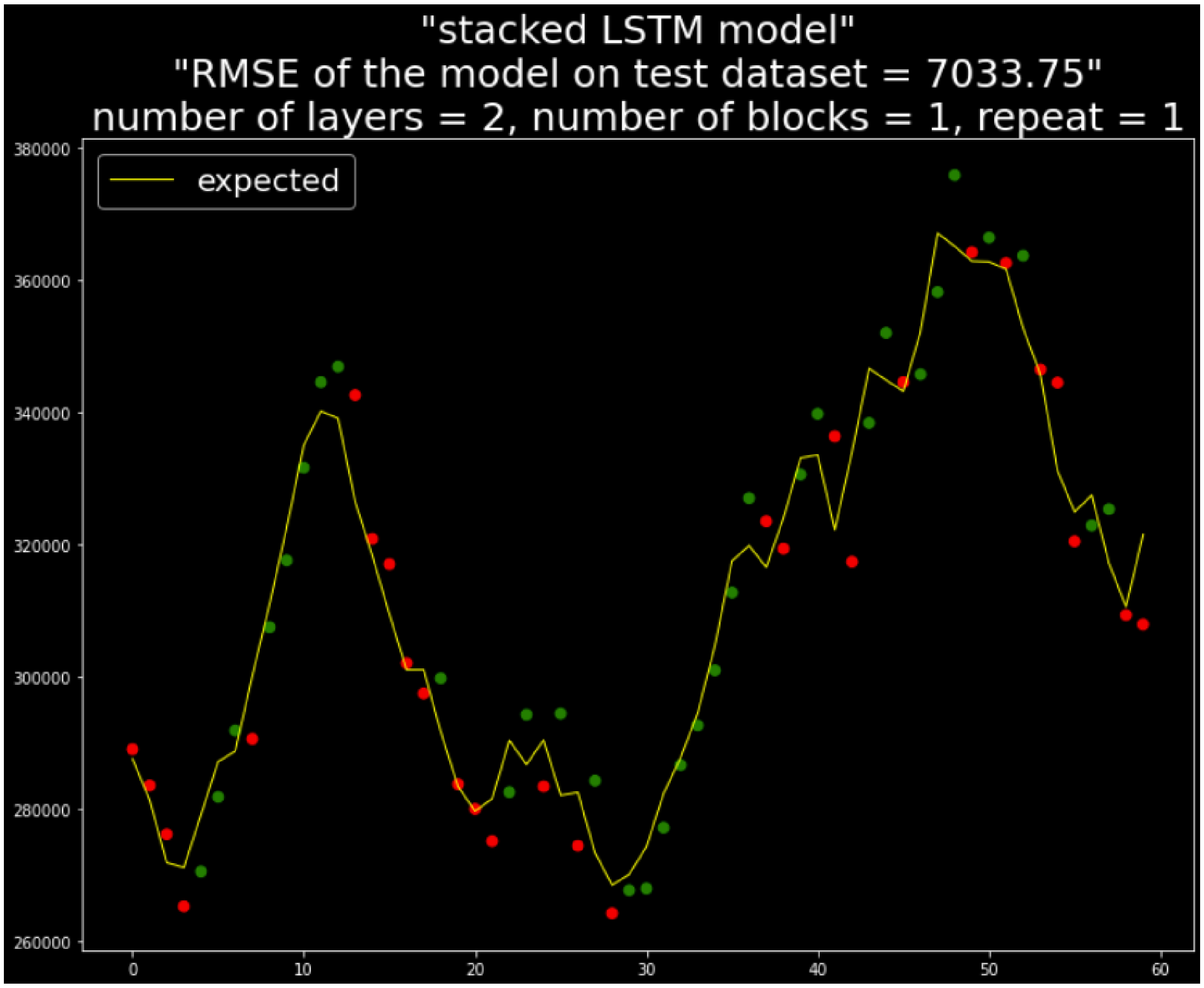


‘2 LSTM layers with 5 units each (Stacked LSTM Model)’



As it can be figured out from the ‘rmse’ losses obtained above, all models except the last one (2 LSTM layers with 5 blocks each) outperforms the naïve model (i.e. all models above, have less ‘rmse’ loss compared to the naïve model which is 7800); therefore, in part ‘a’, nearly all models may extract acceptable features from raw data. Besides, stacked lstm model with 2 lstm layers consisting 5 blocks per layer has reached the least error; therefore, it will be considered as the best model in part ‘a’ (regular dataset).

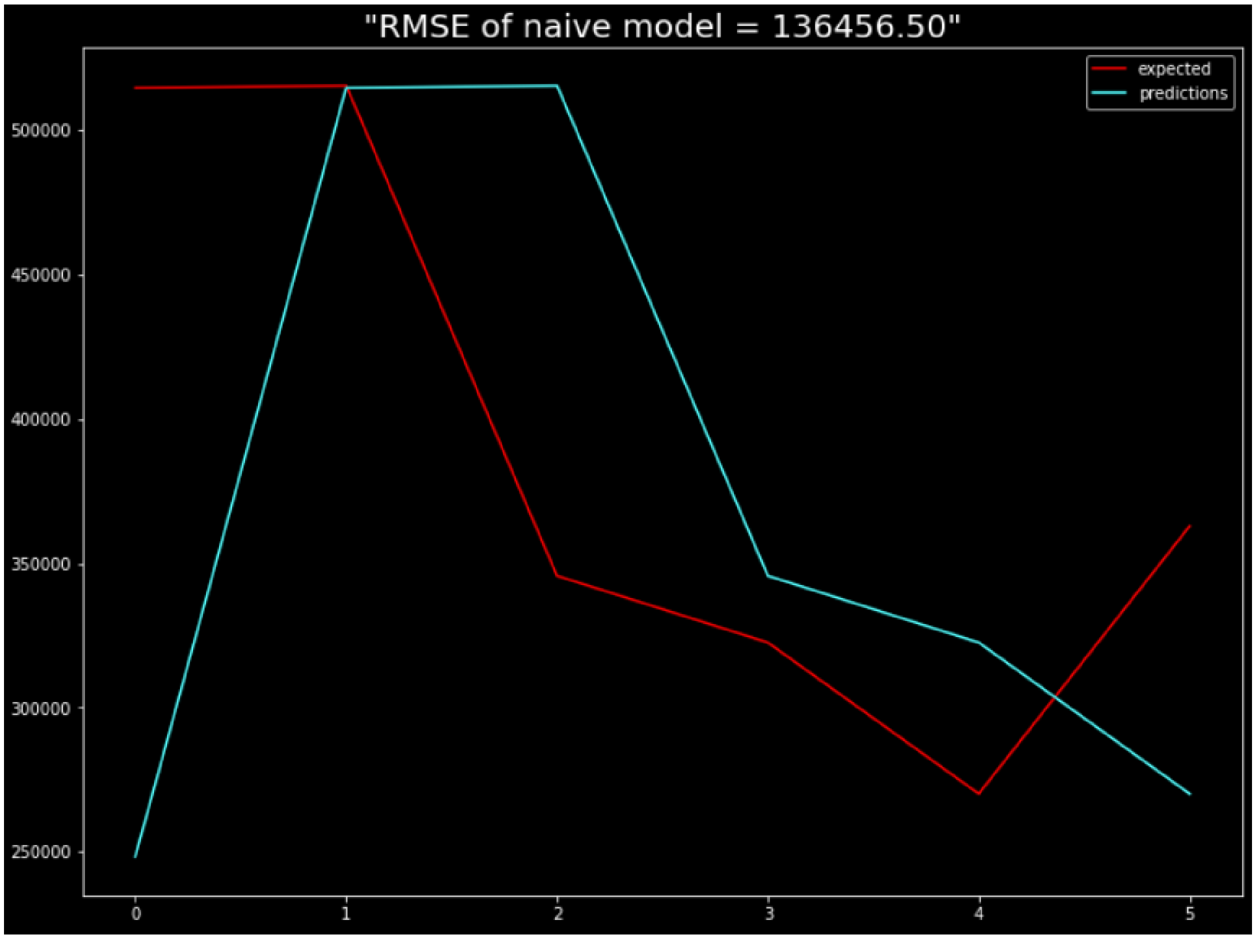
The best model predictions' visualization, as it is required in the question, is shown in figure below:



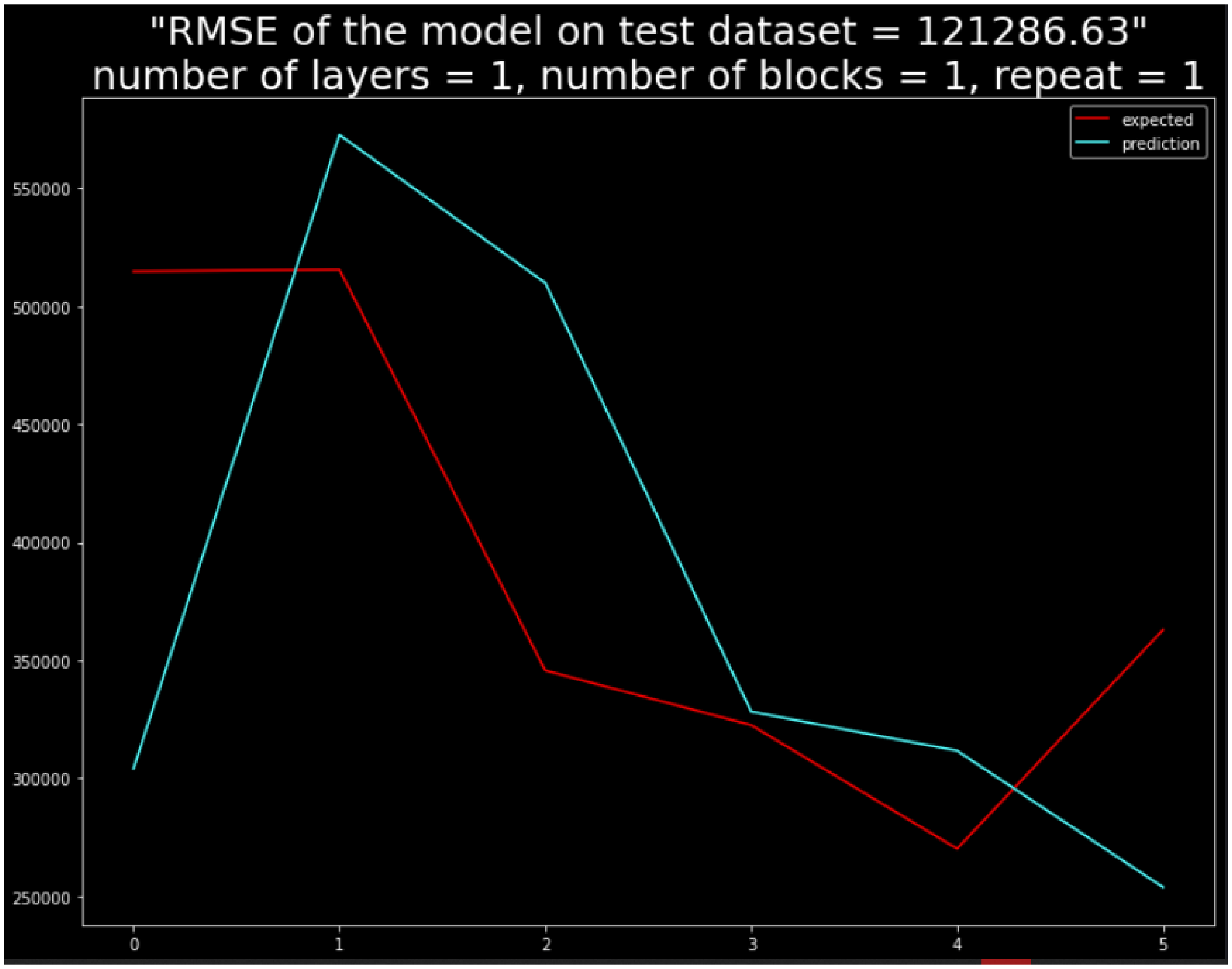
b. (windowed dataset)

In this part, all configurations are exactly the same as the previous part except number of epochs which is 300 with no repeat.

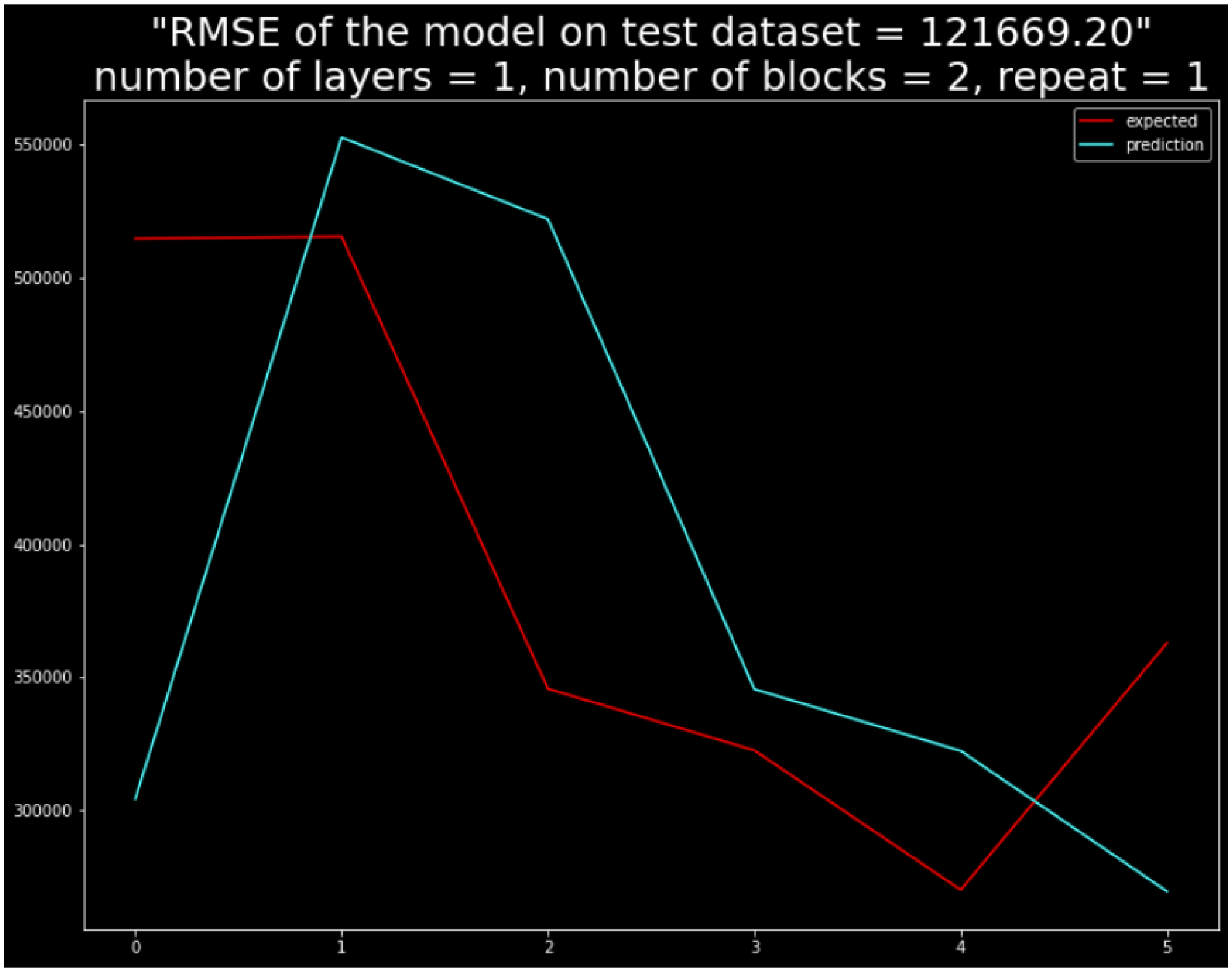
‘naïve model for windowed dataset’



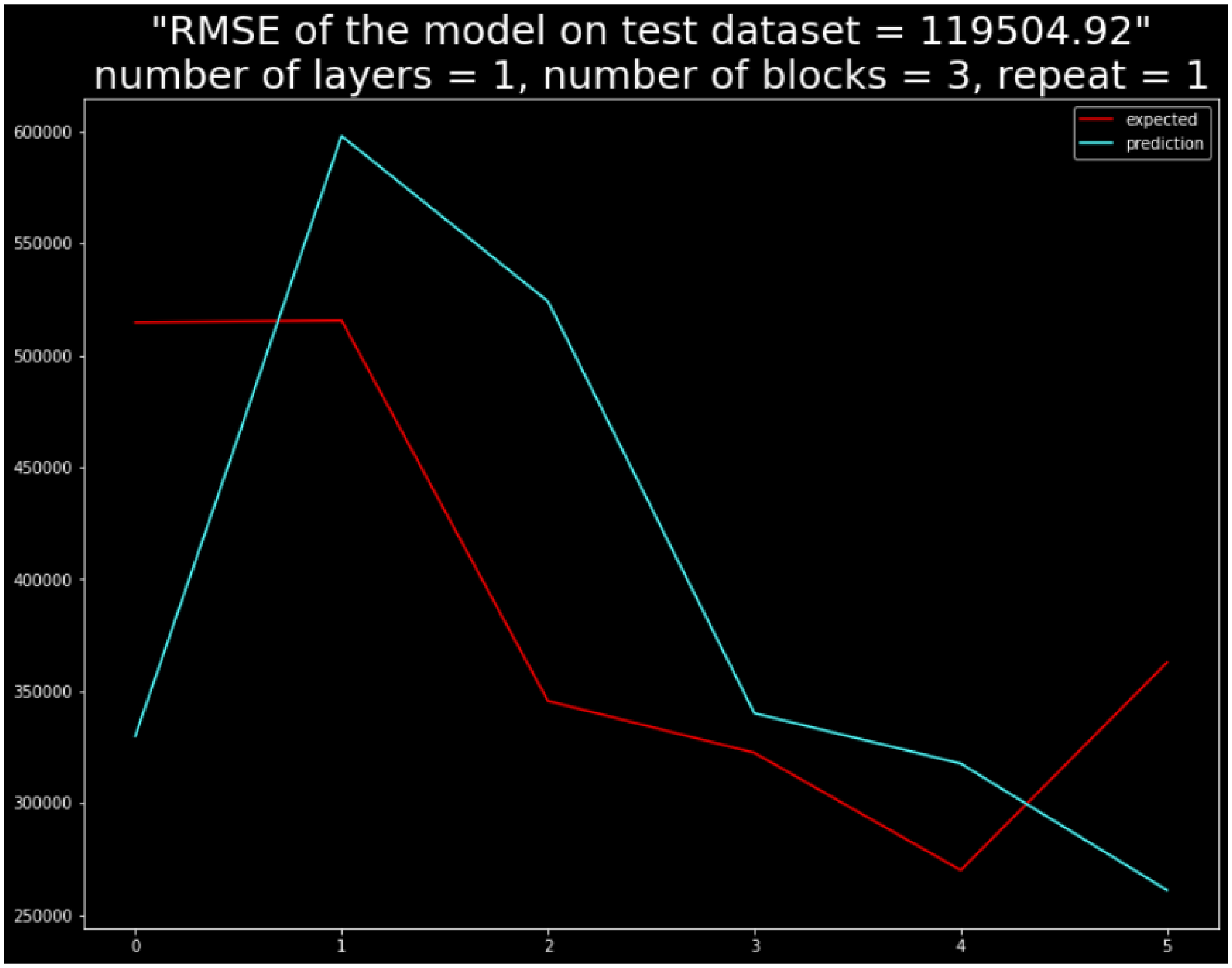
‘1 LSTM layer with 1 unit (basic model)’



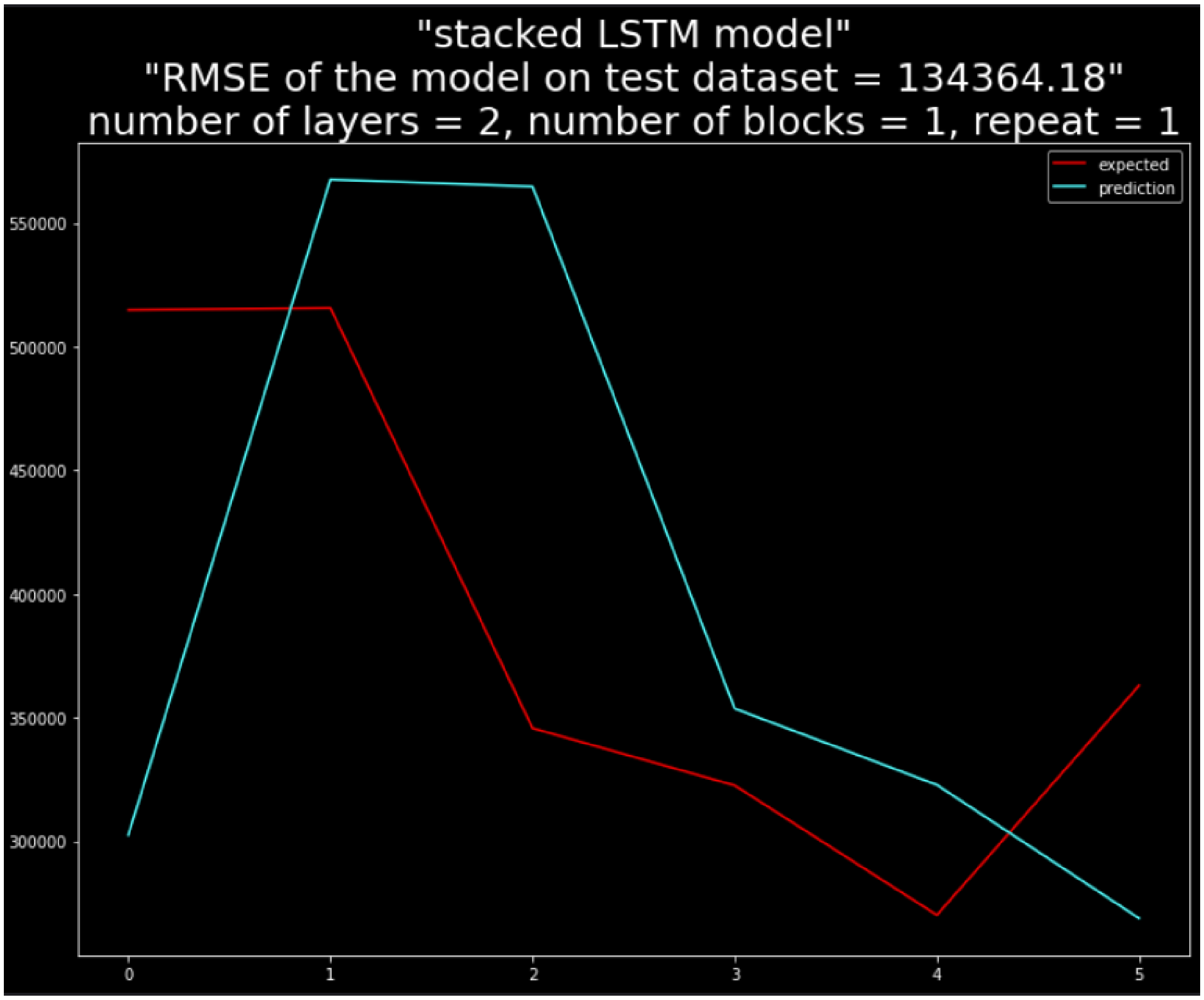
‘1 LSTM layer with 2 units’



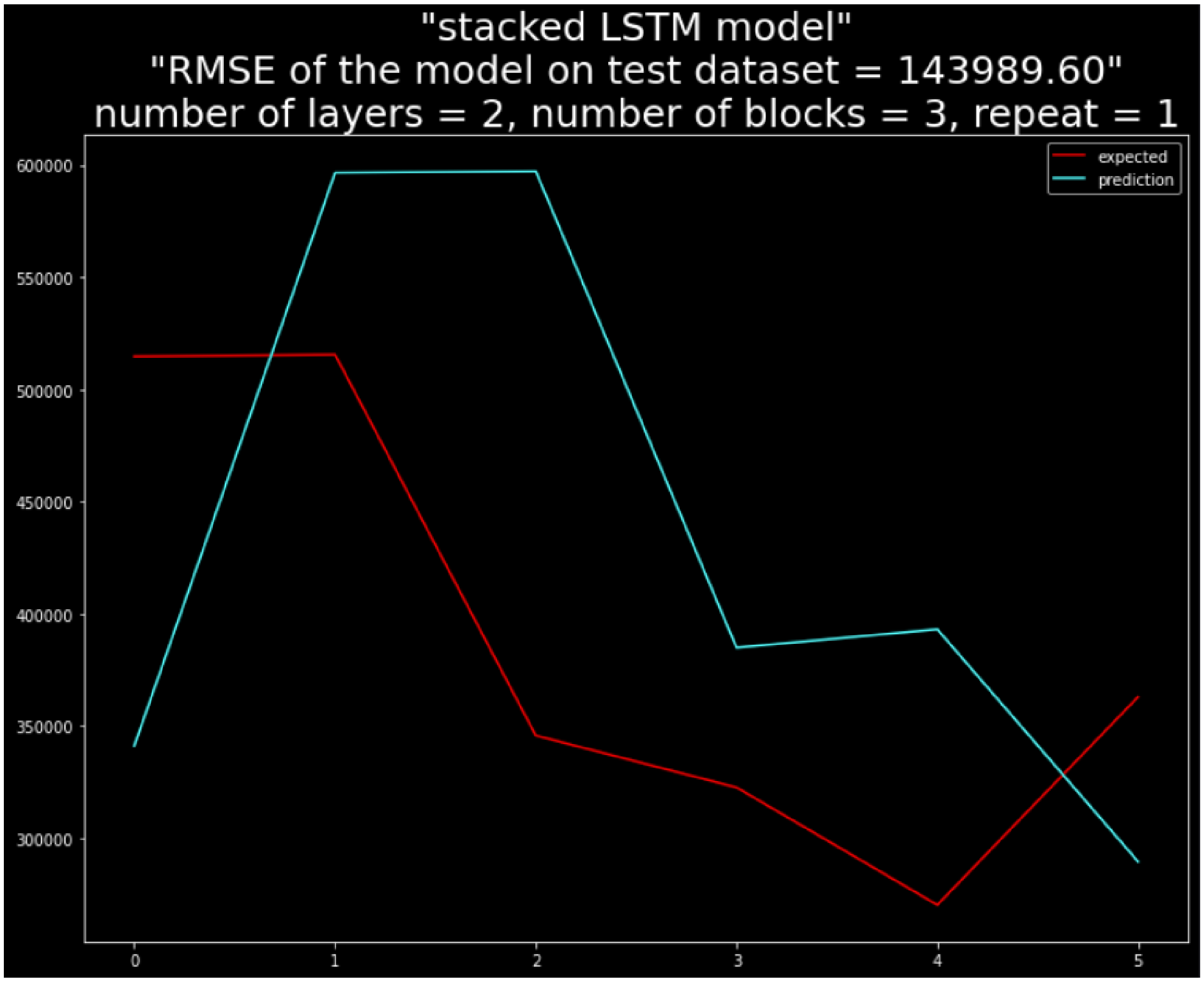
‘1 LSTM layer with 3 units’



‘2 LSTM layers with 1 unit (Stacked LSTM Model)’

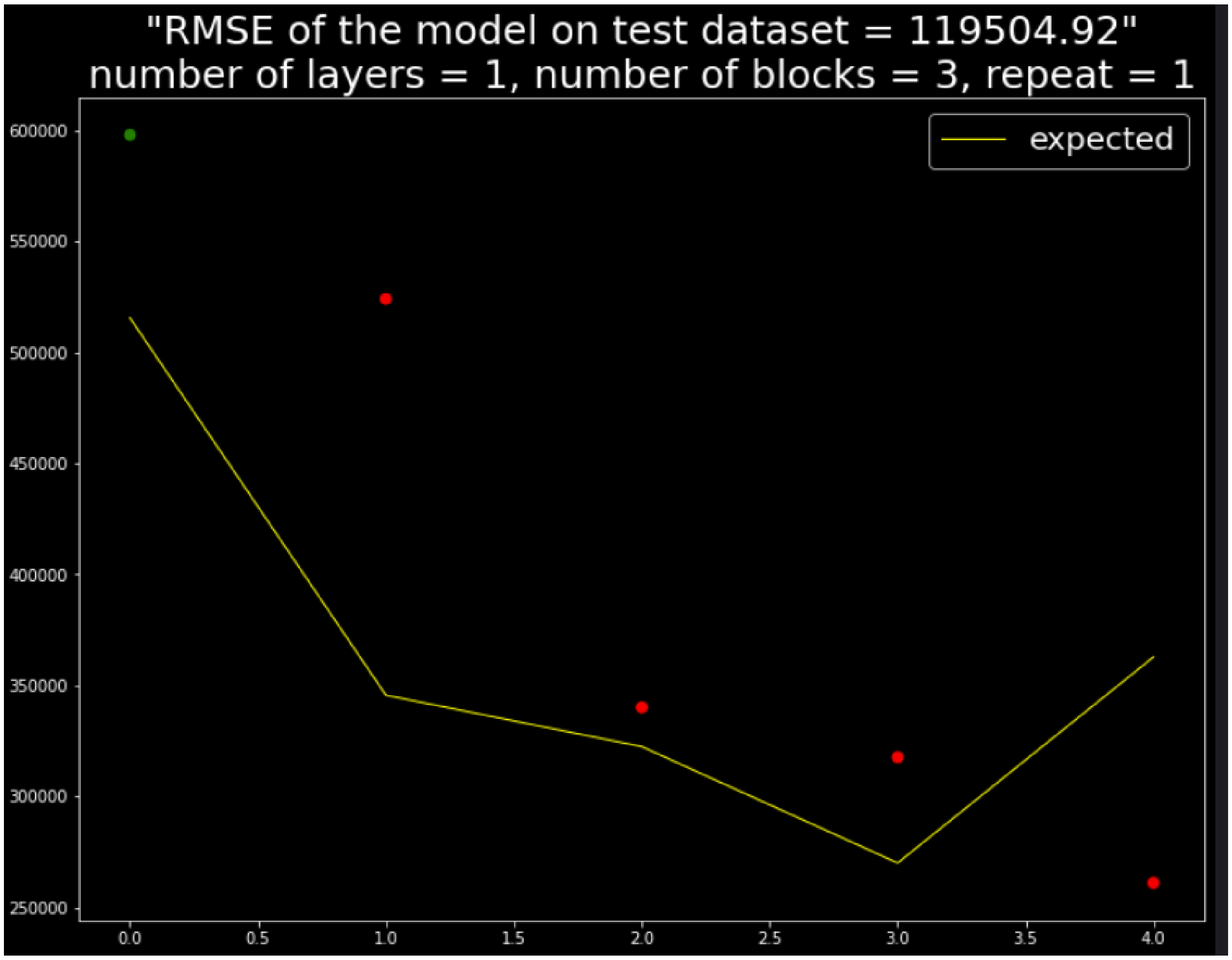


‘2 LSTM layers with 3 units each (Stacked LSTM Model)’



As it can be figured out from the ‘rmse’ losses obtained above, all models except the last one (2 LSTM layers with 3 blocks each) outperforms the naïve model (i.e. all models above, have less ‘rmse’ loss compared to the naïve model which is 136000); therefore, in part ‘a’, nearly all models may extract acceptable features from raw data. Besides, stacked lstm model with 1 lstm layers consisting 3 blocks has reached the least error; therefore, it will be considered as the best model in part ‘b’ (windowed dataset).

The best model predictions' visualization, as it is required in the question, is shown in figure below:



Conclusion:

As it can be figured out from results above, all models defined in part 'a' (regular dataset) outperforms all models of part 'b' which was expectable since dataset of part 'a' is nearly 20 times greater in size compared to that of part 'b' which is windowed by window size of 20 timesteps; on the contrary, it should be noted that models in part 'b' have reduced the 'rmse' loss more than that of part 'a' in term of ratio to their corresponding naïve model's 'rmse' loss.

4.

Remained unsolved!

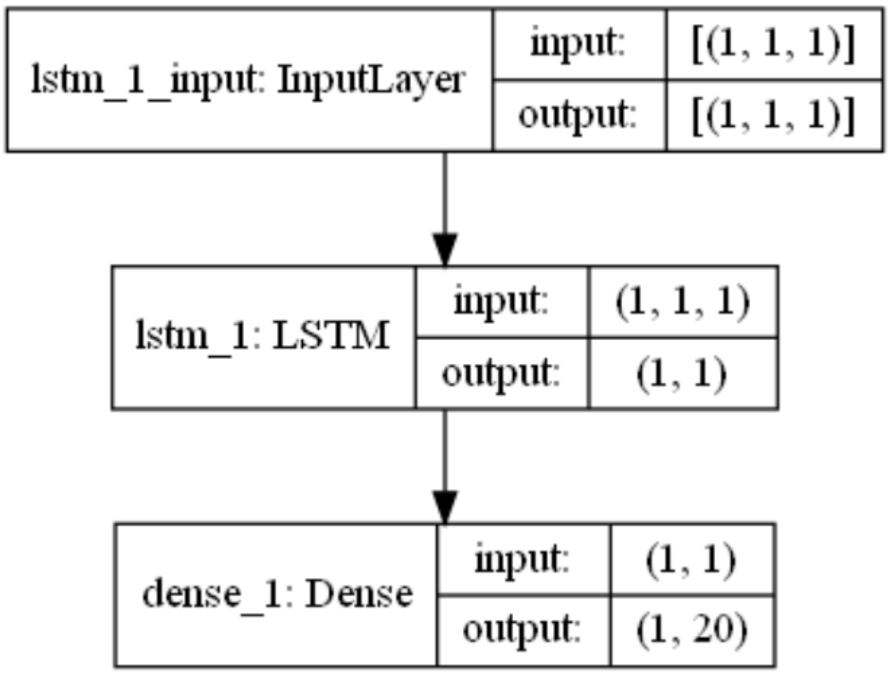
5.

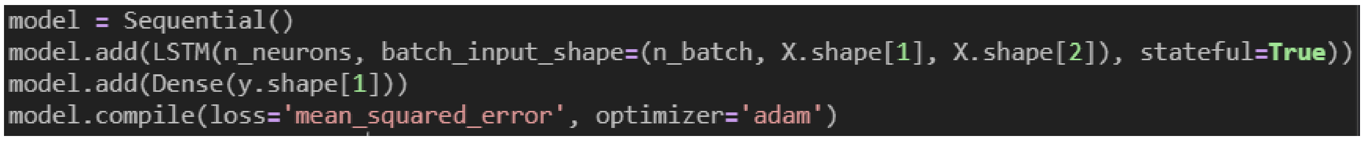
For this question, only regular dataset will be considered since the whole test set of windowed dataset contains only six datapoints which makes no sense in forecasting 20 timesteps ahead.

To solve this type of forecasting (i.e. multi-step forecasting) multiple approaches are introduced, namely:

1. “Direct Multi-step Forecast Strategy”
2. “Recursive Multi-step Forecast”
3. “Direct-Recursive Hybrid Strategies”
4. “Multiple Output Strategy”

Each of the mentioned approaches above has its own pros and cons but the latter (4th approach) seems to be the best one considering the logic of neural networks. In the following, the structure of the intended model and how it is different from the network defined in the question 3 is discussed. The multiple output strategy involves developing a model that is capable of predicting the entire forecast sequence (in our case a sequence of 20 time-step ahead) at once after giving the input; therefore, it is sufficient to replace the last layer of the question 3 (which had a single neuron for forecasting only a one time-step ahead) with a layer consisting 20 nodes. This way, each of the 20 nodes will predict one of the intended 20 time-steps. To train the mentioned model in supervised manner, each input time-step has to have a label consisting of its corresponding next 20 time-steps (as against in question one with label of next time-step). The model’s structure is as follows:





```
model = Sequential()
model.add(LSTM(n_neurons, batch_input_shape=(n_batch, X.shape[1], X.shape[2]), stateful=True))
model.add(Dense(y.shape[1]))
model.compile(loss='mean_squared_error', optimizer='adam')
```

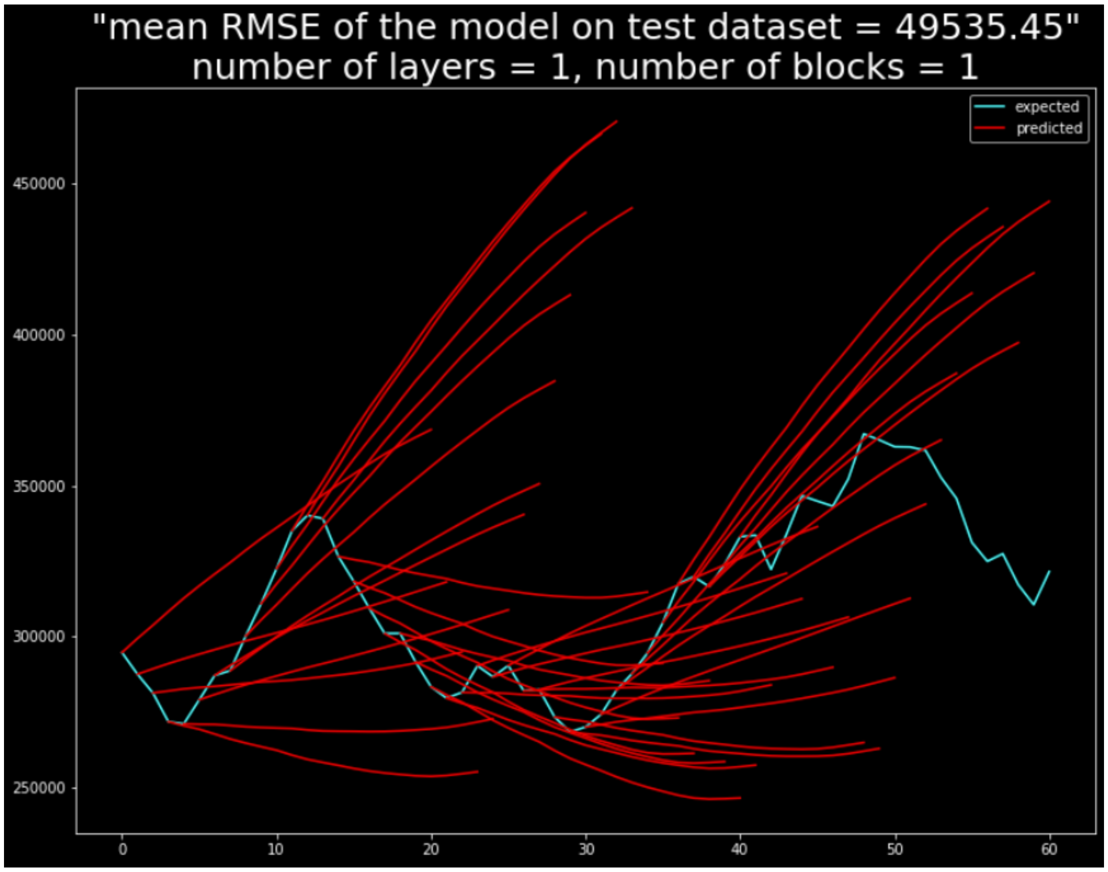
Only number of nodes of the 'Dense' layer is changed compared to the model of question 3.

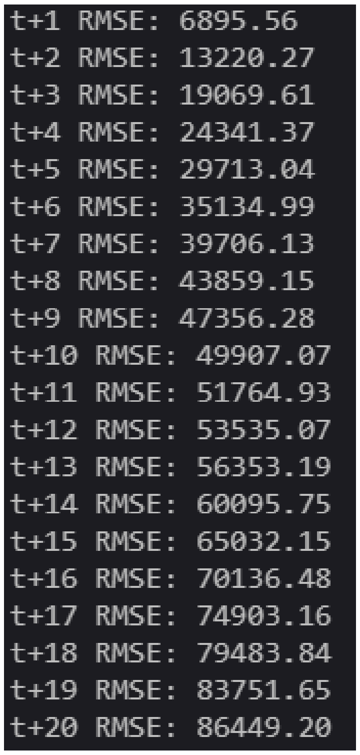
Again, similar to question 3, multiple experiments were conducted in order to find an optimal configuration (i.e. number of LSTM layers and number of blocks of each layer).

Each model is trained by 100 number of epochs:

(pay attention that each model's performance is evaluated by it's obtained 'RMSE' error which is shown on title of each figure; the obtained RMSE error is the mean of all RMSE errors of all timesteps.)

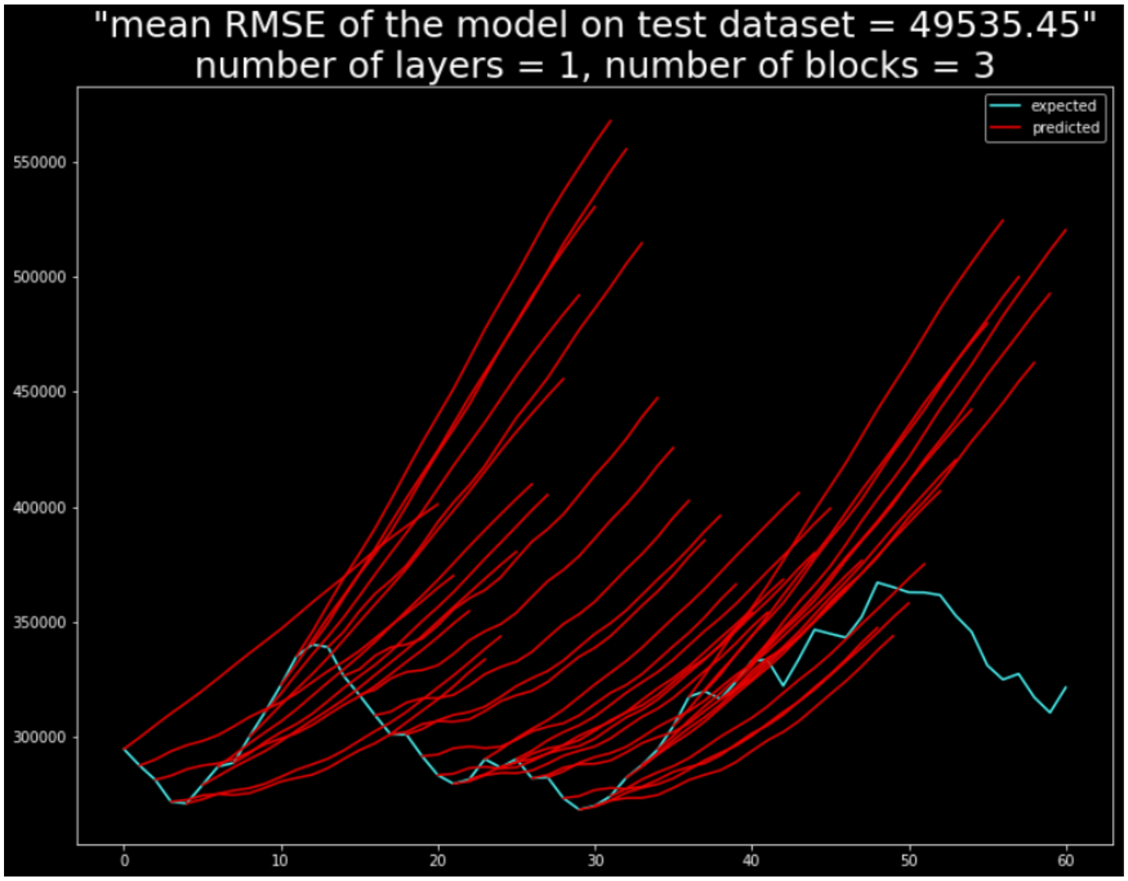
‘1 LSTM layer with 1 unit (basic model)’

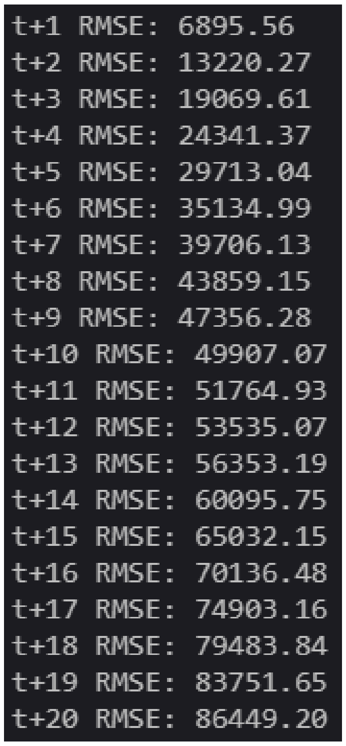




```
t+1 RMSE: 6895.56
t+2 RMSE: 13220.27
t+3 RMSE: 19069.61
t+4 RMSE: 24341.37
t+5 RMSE: 29713.04
t+6 RMSE: 35134.99
t+7 RMSE: 39706.13
t+8 RMSE: 43859.15
t+9 RMSE: 47356.28
t+10 RMSE: 49907.07
t+11 RMSE: 51764.93
t+12 RMSE: 53535.07
t+13 RMSE: 56353.19
t+14 RMSE: 60095.75
t+15 RMSE: 65032.15
t+16 RMSE: 70136.48
t+17 RMSE: 74903.16
t+18 RMSE: 79483.84
t+19 RMSE: 83751.65
t+20 RMSE: 86449.20
```

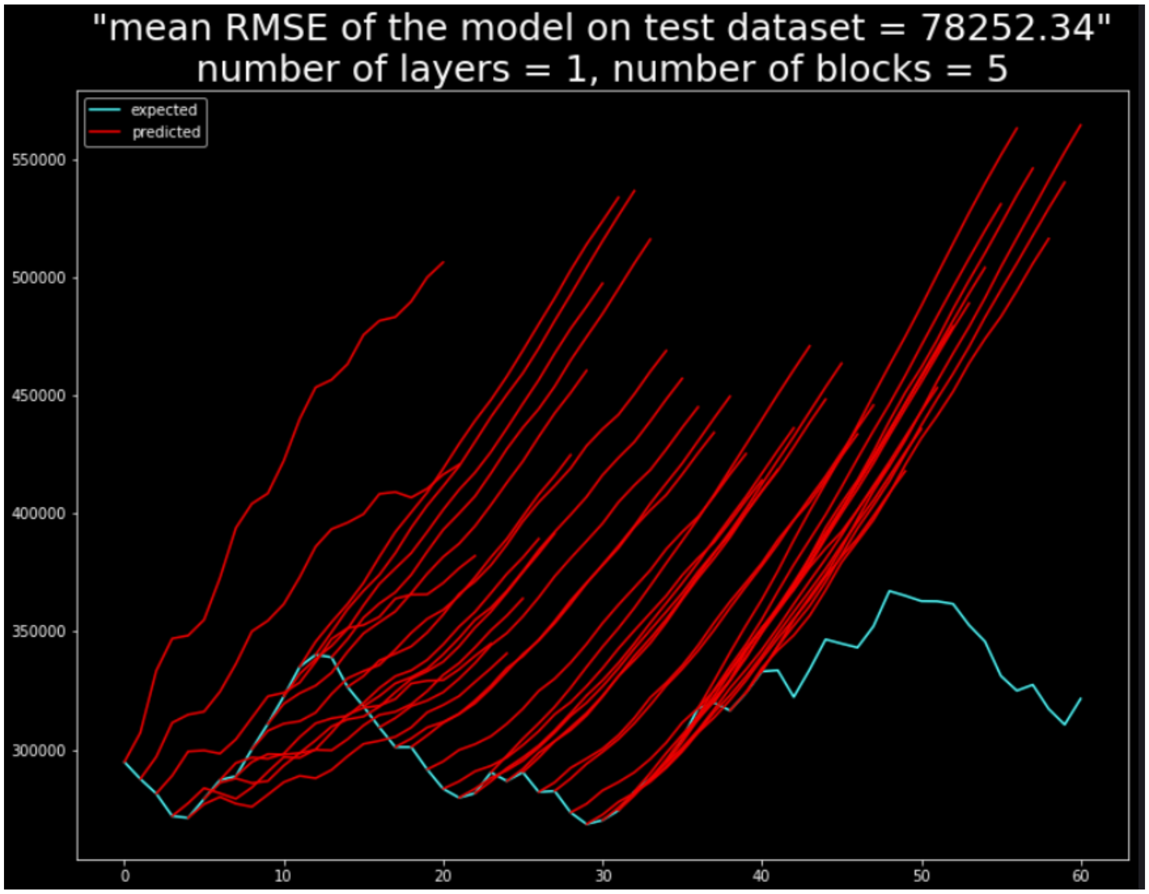
‘1 LSTM layer with 3 units’

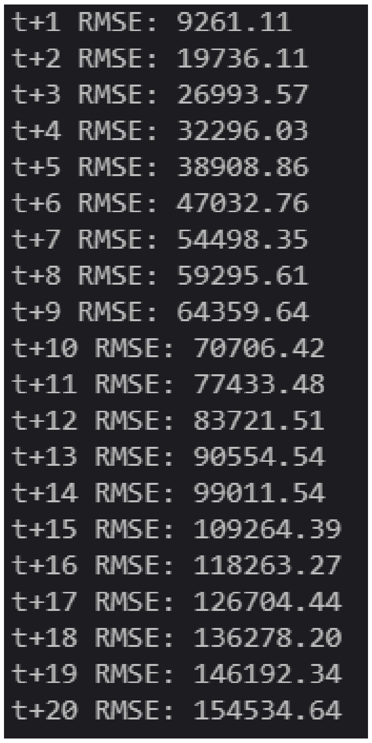




```
t+1 RMSE: 6895.56
t+2 RMSE: 13220.27
t+3 RMSE: 19069.61
t+4 RMSE: 24341.37
t+5 RMSE: 29713.04
t+6 RMSE: 35134.99
t+7 RMSE: 39706.13
t+8 RMSE: 43859.15
t+9 RMSE: 47356.28
t+10 RMSE: 49907.07
t+11 RMSE: 51764.93
t+12 RMSE: 53535.07
t+13 RMSE: 56353.19
t+14 RMSE: 60095.75
t+15 RMSE: 65032.15
t+16 RMSE: 70136.48
t+17 RMSE: 74903.16
t+18 RMSE: 79483.84
t+19 RMSE: 83751.65
t+20 RMSE: 86449.20
```

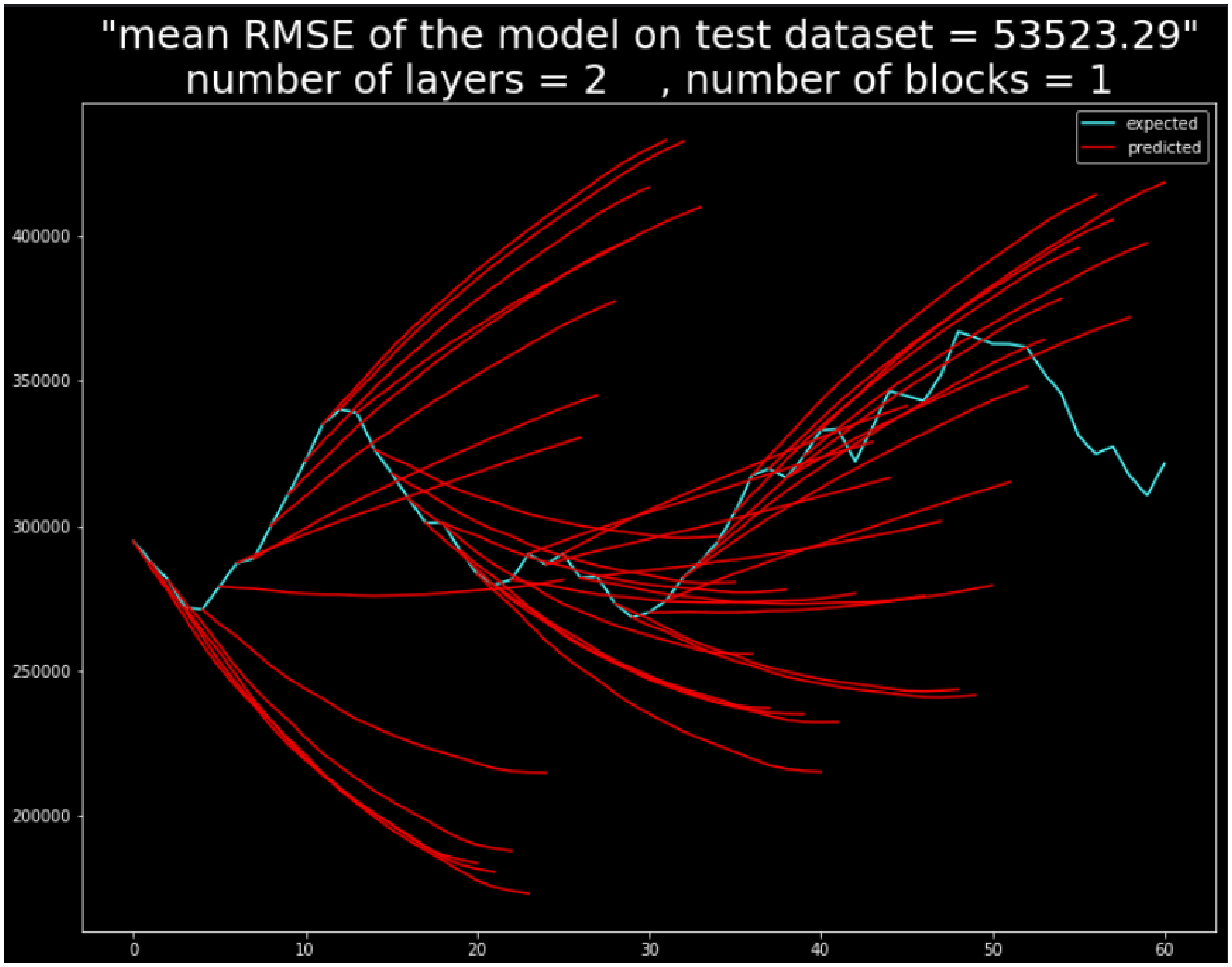
‘1 LSTM layer with 5 units’

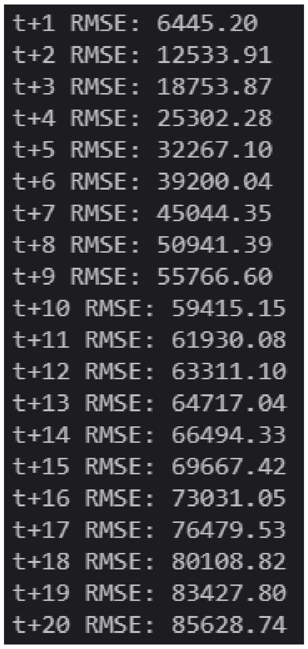




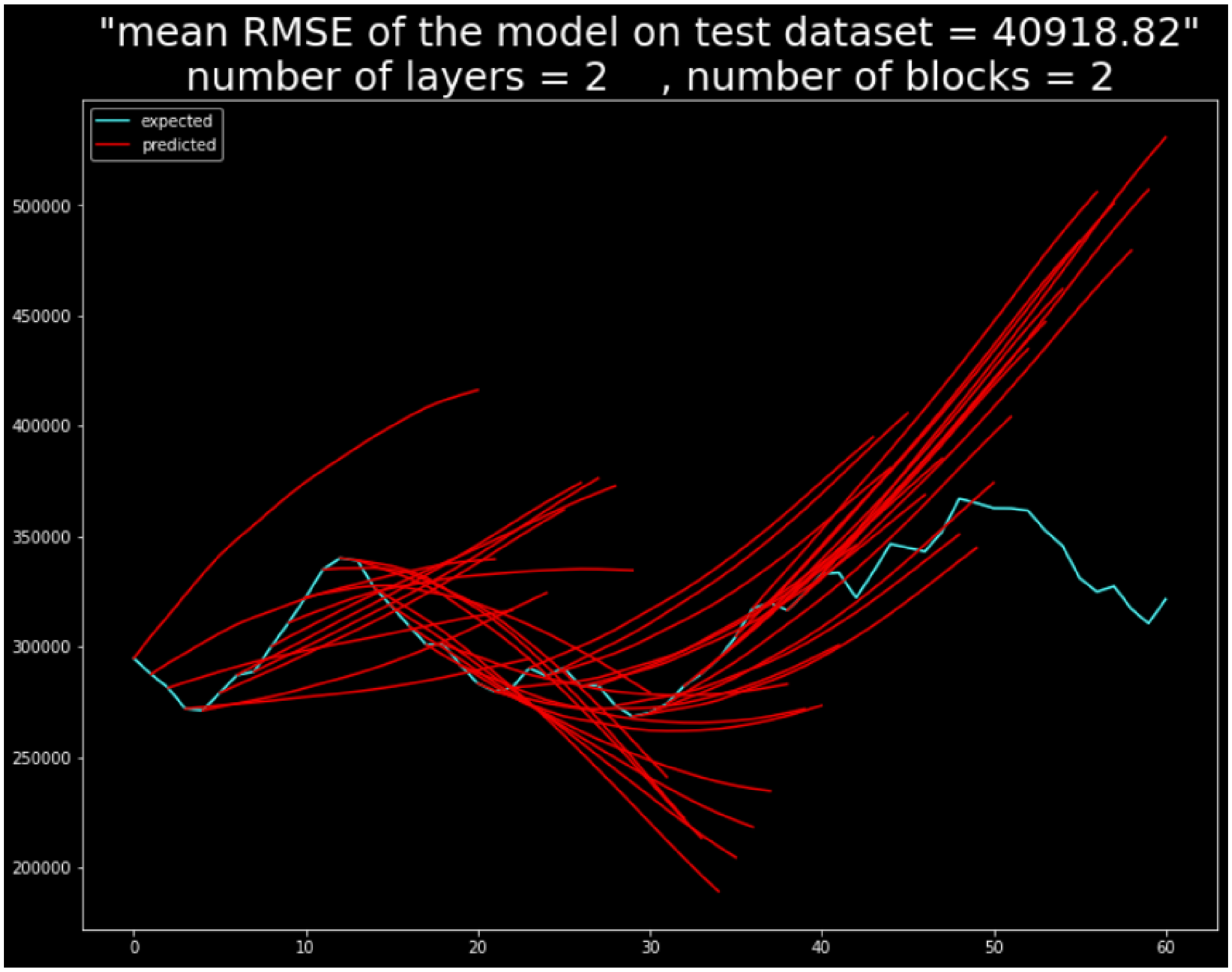
```
t+1 RMSE: 9261.11
t+2 RMSE: 19736.11
t+3 RMSE: 26993.57
t+4 RMSE: 32296.03
t+5 RMSE: 38908.86
t+6 RMSE: 47032.76
t+7 RMSE: 54498.35
t+8 RMSE: 59295.61
t+9 RMSE: 64359.64
t+10 RMSE: 70706.42
t+11 RMSE: 77433.48
t+12 RMSE: 83721.51
t+13 RMSE: 90554.54
t+14 RMSE: 99011.54
t+15 RMSE: 109264.39
t+16 RMSE: 118263.27
t+17 RMSE: 126704.44
t+18 RMSE: 136278.20
t+19 RMSE: 146192.34
t+20 RMSE: 154534.64
```

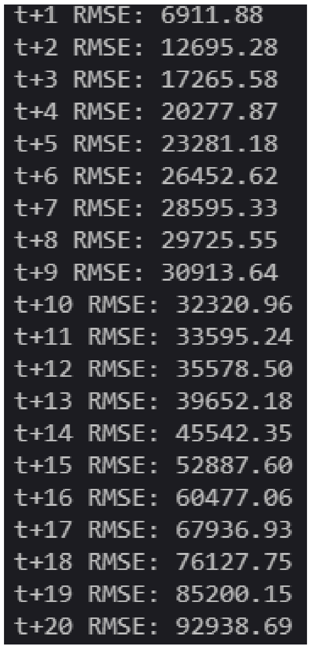
‘2 LSTM layers with 1 unit (Stacked LSTM Model)’



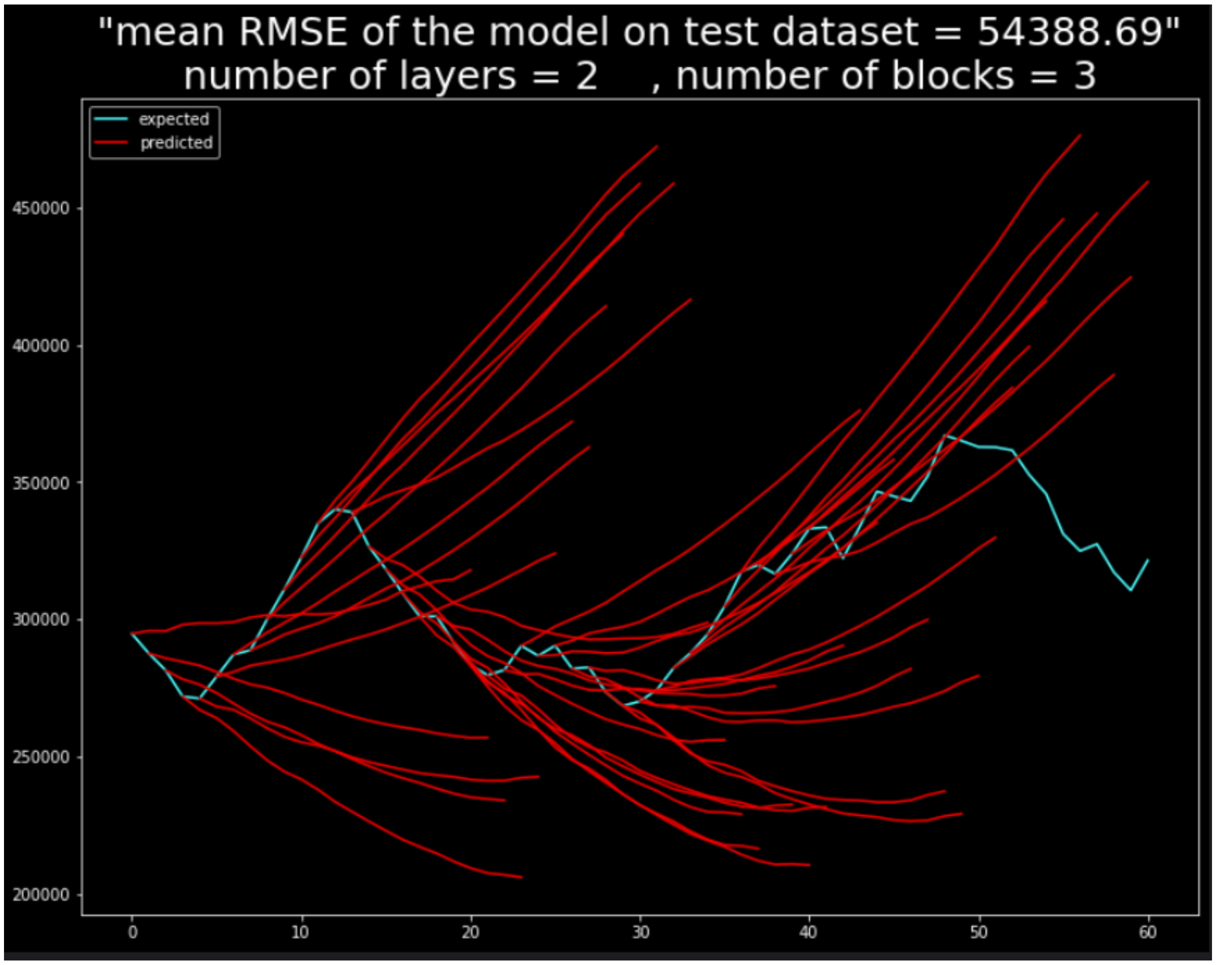

```
t+1 RMSE: 6445.20
t+2 RMSE: 12533.91
t+3 RMSE: 18753.87
t+4 RMSE: 25302.28
t+5 RMSE: 32267.10
t+6 RMSE: 39200.04
t+7 RMSE: 45044.35
t+8 RMSE: 50941.39
t+9 RMSE: 55766.60
t+10 RMSE: 59415.15
t+11 RMSE: 61930.08
t+12 RMSE: 63311.10
t+13 RMSE: 64717.04
t+14 RMSE: 66494.33
t+15 RMSE: 69667.42
t+16 RMSE: 73031.05
t+17 RMSE: 76479.53
t+18 RMSE: 80108.82
t+19 RMSE: 83427.80
t+20 RMSE: 85628.74
```

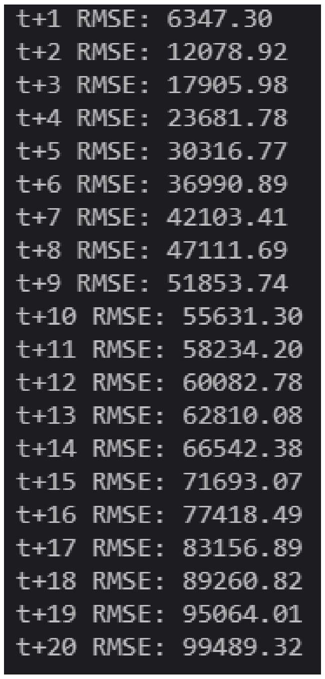
‘2 LSTM layers with 2 units each (Stacked LSTM Model)’




```
t+1 RMSE: 6911.88
t+2 RMSE: 12695.28
t+3 RMSE: 17265.58
t+4 RMSE: 20277.87
t+5 RMSE: 23281.18
t+6 RMSE: 26452.62
t+7 RMSE: 28595.33
t+8 RMSE: 29725.55
t+9 RMSE: 30913.64
t+10 RMSE: 32320.96
t+11 RMSE: 33595.24
t+12 RMSE: 35578.50
t+13 RMSE: 39652.18
t+14 RMSE: 45542.35
t+15 RMSE: 52887.60
t+16 RMSE: 60477.06
t+17 RMSE: 67936.93
t+18 RMSE: 76127.75
t+19 RMSE: 85200.15
t+20 RMSE: 92938.69
```

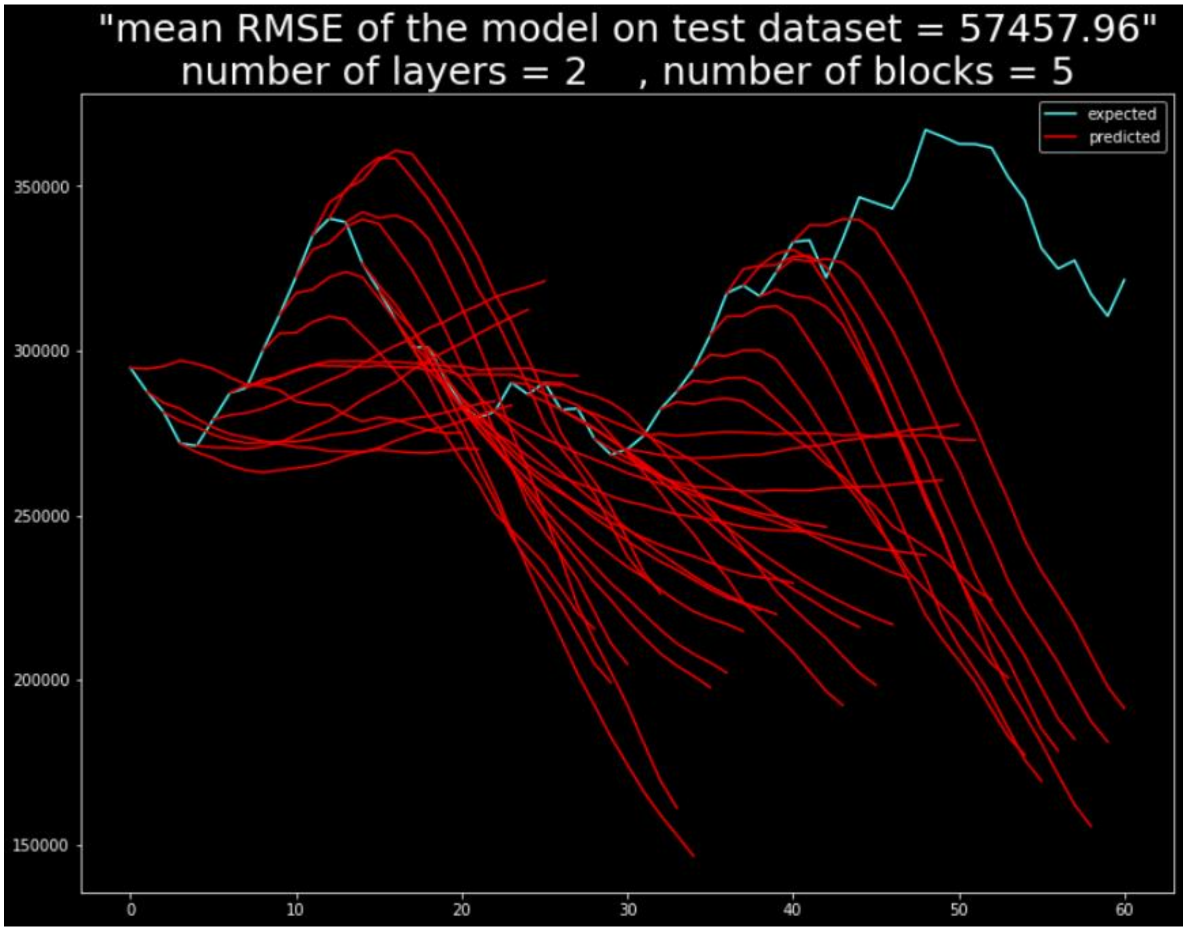
‘2 LSTM layers with 3 units each (Stacked LSTM Model)’

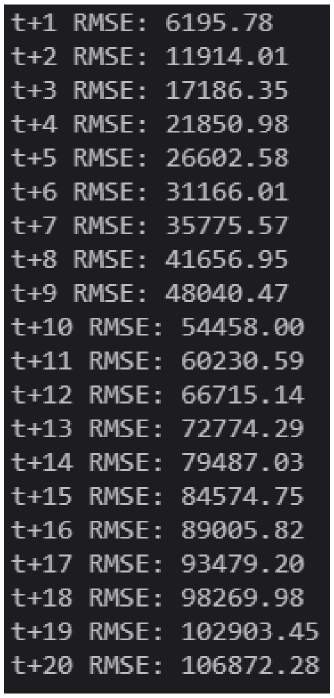




```
t+1 RMSE: 6347.30
t+2 RMSE: 12078.92
t+3 RMSE: 17905.98
t+4 RMSE: 23681.78
t+5 RMSE: 30316.77
t+6 RMSE: 36990.89
t+7 RMSE: 42103.41
t+8 RMSE: 47111.69
t+9 RMSE: 51853.74
t+10 RMSE: 55631.30
t+11 RMSE: 58234.20
t+12 RMSE: 60082.78
t+13 RMSE: 62810.08
t+14 RMSE: 66542.38
t+15 RMSE: 71693.07
t+16 RMSE: 77418.49
t+17 RMSE: 83156.89
t+18 RMSE: 89260.82
t+19 RMSE: 95064.01
t+20 RMSE: 99489.32
```

‘2 LSTM layers with 5 units each (Stacked LSTM Model)’

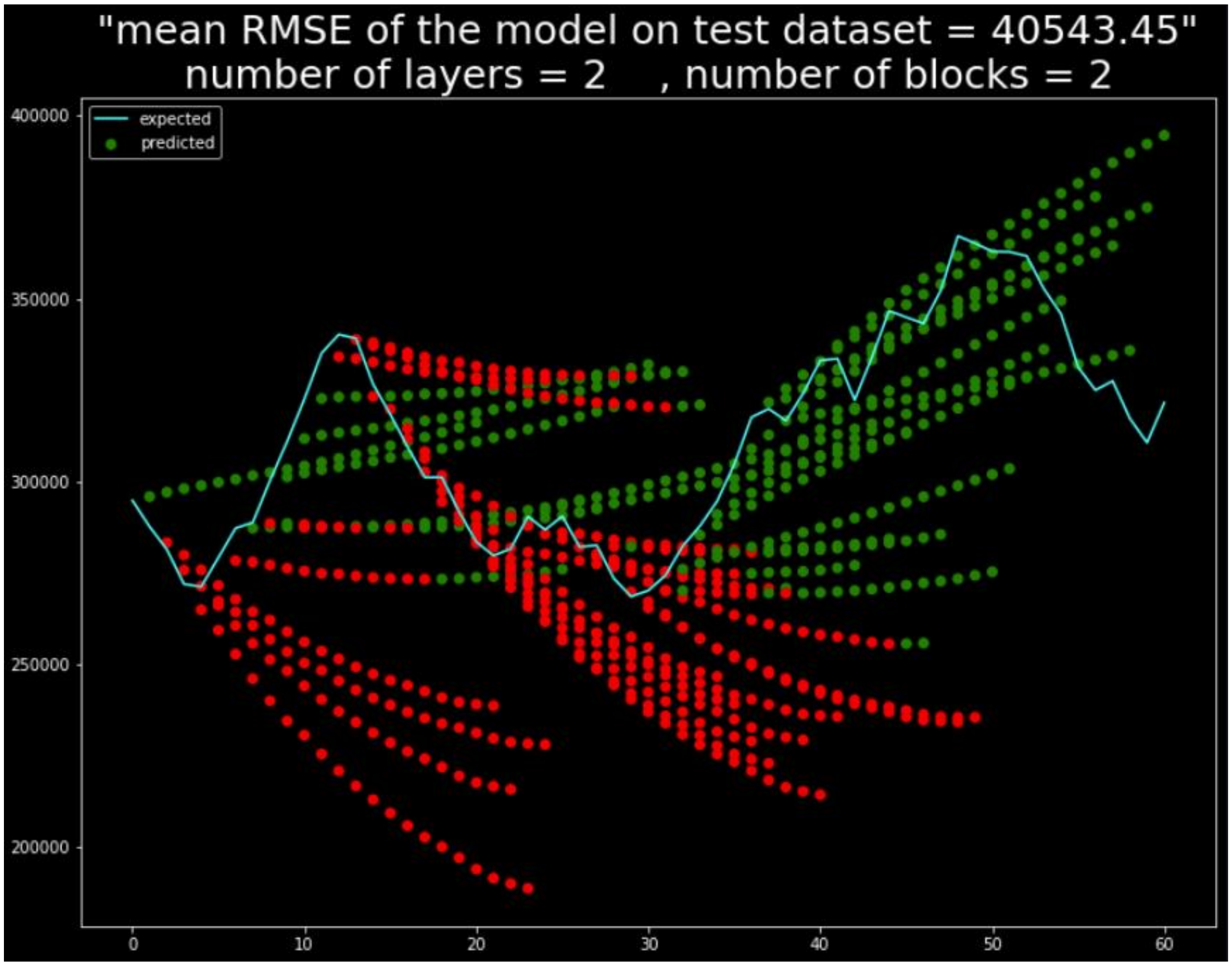




```
t+1 RMSE: 6195.78
t+2 RMSE: 11914.01
t+3 RMSE: 17186.35
t+4 RMSE: 21850.98
t+5 RMSE: 26602.58
t+6 RMSE: 31166.01
t+7 RMSE: 35775.57
t+8 RMSE: 41656.95
t+9 RMSE: 48040.47
t+10 RMSE: 54458.00
t+11 RMSE: 60230.59
t+12 RMSE: 66715.14
t+13 RMSE: 72774.29
t+14 RMSE: 79487.03
t+15 RMSE: 84574.75
t+16 RMSE: 89005.82
t+17 RMSE: 93479.20
t+18 RMSE: 98269.98
t+19 RMSE: 102903.45
t+20 RMSE: 106872.28
```

As it can be figured out from the 'rmse' losses obtained above, the Stacked LSTM model with 2 LSTM layers consisting of two neurons each reached the least error; therefore, it will be considered as the best model in part.

The best model predictions' visualization, as it is required in the question, is shown in figure below:



One of the most noticeable facts about the model defined in this question (compared to that of the question 3) is that as the predictions go further in time, the corresponding error increases significantly (further time-steps from the input results in more error in prediction). One possible solution to overcome this problem is to consider a sequence of inputs when predicting a sequence of outputs. For example, in our case, it could be better to consider the last 20 observations for forecasting the next 20 time-steps. This approach is known as seq2seq multi-step forecasting which results in higher accuracy.

“Finish