A **Recurrent Neural Network (RNN)** is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. It is a form of generalized feed forward network that has internal memory which can be used to process the input data.

In this assignment, a recurrent neural network (RNN) is designed and trained on S&P 500 data. The model is tested by predicting the next one,two,three or four values using a sliding sampling window of width 180 days.

The models are also re-tested on noise-corrupted data to observe the performance. The goal of this project is to optimize the RNN to give the best possible results.

DATASET

The S&P 500 dataset is collected from the Yahoo Finance website between the time frame of January 4, 1960 to December 31, 2020. The table from the website is scraped and stored into a csv file. Shiller P/E ratio is one of the standard metrics used to evaluate whether a market is overvalued, undervalued, or fairly-valued. It can be downloaded from the Shiller-PE website.

The Shiller P/E ratio data was collected monthly from 1960 to 2020. We use linear interpolation method to convert this monthly data to daily data. The formula for linear interpolation is as follows:

$y = y1 + ((x – x1) / (x2 – x1)) * (y2 – y1)$ where,

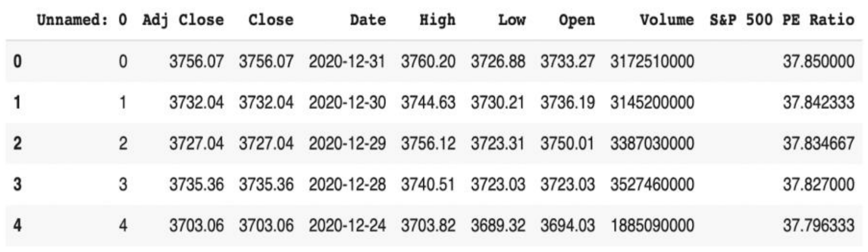
X = known value

y= unknown value

(x1,y1) = coordinates below the known value

(x2,y2) = coordinates above the known value

S&P 500 data and Shiller P/E ratio data were then merged and stored in finalDS.csv. The data was then divided into training and test sets.



	Unnamed: 0	Adj Close	Close	Date	High	Low	Open	Volume	S&P 500 PE Ratio
0	0	3756.07	3756.07	2020-12-31	3760.20	3726.88	3733.27	3172510000	37.850000
1	1	3732.04	3732.04	2020-12-30	3744.63	3730.21	3736.19	3145200000	37.842333
2	2	3727.04	3727.04	2020-12-29	3756.12	3723.31	3750.01	3387030000	37.834667
3	3	3735.36	3735.36	2020-12-28	3740.51	3723.03	3723.03	3527460000	37.827000
4	4	3703.06	3703.06	2020-12-24	3703.82	3689.32	3694.03	1885090000	37.796333

The generated dataset is a time series data of length N, defined as $CP_0, CP_1, \ldots, CP_{N-1}$ in which CP_i is the close price on day 'i', $0 \leq i < N$. The entire dataset is divided into fixed window size of W = 180. Therefore, whenever the window is moved to the right by size W , there is no overlapping between data in all the sliding windows. Then the data values are normalized within range of [0,1]. Then the data values are split into test and train datasets.

RNN MODEL

The architecture employed to deploy this model is the **LSTM model**. ***LSTM*** stands for Long Short Term Memory. Long Short-Term Memory (LSTM) networks are a ***modified version*** of recurrent neural networks, which makes it easier to remember past data in memory. The ***vanishing gradient*** problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. This focuses on a single dependent variable. The basic assumption behind **the univariate prediction approach** is that the value of a time-series at time-step t is closely related to the values at the previous time-steps t-1, t-2, t-3 and so on.
Univariate models are easier to develop than multivariate models. The dependent variable in stock market forecasting is usually the closing or opening price of a finance asset. A forecasting model that is trained solely on the basis of price developments attempts.
It trains the model by using back-propagation. In an LSTM network,**three gates** are present:

i. Input gate — discover which value from input should be used to modify the memory. **Sigmoid** function decides which values to let through **0,1.** and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from**-1** to **1**

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

ii. Forget gate — discover what details to be discarded from the block. It is decided by the **sigmoid function.** it looks at the previous state(**ht-1**) and the content input(**Xt**) and outputs a number between **0(***omit this*)and **1(***keep this***)**for each number in the cell state **Ct−1**.

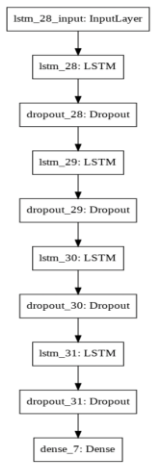
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

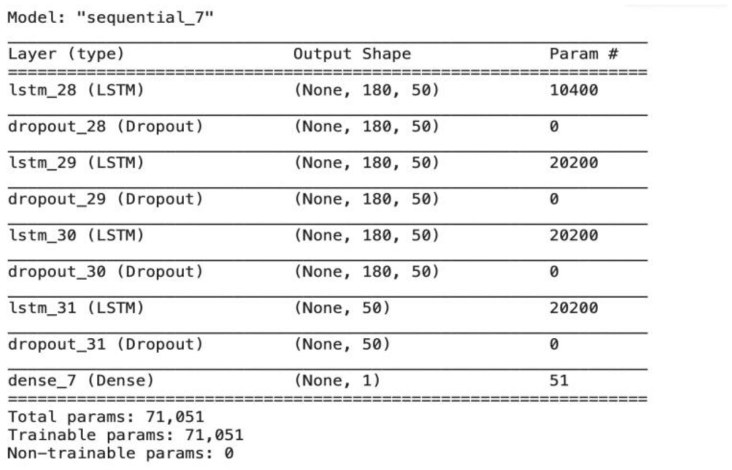
iii. Output gate — the input and the memory of the block is used to decide the output. **Sigmoid** function decides which values to let through **0,1.** and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from**-1** to **1** and multiplied with output of **Sigmoid.**

$$o_t = \sigma\left(W_o \ [\, h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

The model is trained on the entire data set from 1960 to 2014 and tested on data from 2015 to 2020. The RNN built is a regression model that predicts the future Stock Close Price. The resulting output of the predicted price is then compared to the original price values to evaluate the model.

RNN MODEL ARCHITECTURE





```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_28 (LSTM)               (None, 180, 50)           10400
_________________________________________________________________
dropout_28 (Dropout)         (None, 180, 50)           0
_________________________________________________________________
lstm_29 (LSTM)               (None, 180, 50)           20200
_________________________________________________________________
dropout_29 (Dropout)         (None, 180, 50)           0
_________________________________________________________________
lstm_30 (LSTM)               (None, 180, 50)           20200
_________________________________________________________________
dropout_30 (Dropout)         (None, 180, 50)           0
_________________________________________________________________
lstm_31 (LSTM)               (None, 50)                20200
_________________________________________________________________
dropout_31 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 51
=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
```