Executive Summary

The **Dow Jones Industrial Average** (**DJIA**) is a stock market index that indicates the value of 30 large publicly owned companies based in the United States.The Dow typically makes changes when a company loses market capitalization due to financial distress or when a broader economic shift occurs.The value of the Dow is the sum of the price of one share of stock for each component company. The sum is corrected by a factor called the Dow DIvisor which changes whenever one of the component stocks has a stock split or stock dividend.
[DJIA = SUM (Component stock prices) / Dow Divisor]

Since DJI index values generally form a pattern, we can make a computer understand these patterns and effectively help us predict a new value for the upcoming days.
A Recurrent Neural Network implementing a LSTM model was used to perform the same and the model was analysed by plotting the real values along with the predicted values. The model was saved for further predictions using Pickle.
A MSE Loss of 3.1 was obtained after 100 epochs but the model was able to understand the flow of trends in the data.
Further optimization can be performed by tuning the hyper-parameters.

Problem Analysis

The **Dow Jones Industrial Average** (**DJIA**) is a stock market index that indicates the value of 30 large publicly owned companies based in the United States trading on the New York Stock Exchange (NYSE) and the NASDAQ. When the index launched, it included just 12 companies. The composition of the index changes as the economy changes. The Dow typically makes changes when a company loses market capitalization due to financial distress or when a broader economic shift occurs.

Initially, the Dow was calculated by adding the prices of the twelve Dow component stocks and dividing by twelve with the end result being a simple average. Over time, there have been additions and subtractions to the index, such as mergers and stock splits that had to be accounted for in the index where just calculating the arithmetic mean would not suffice.

DJIA = SUM (Component stock prices) / Dow Divisor
Stock split merely changed the price, not the value of the company. To absorb the effects of price changes from splits, those calculating the DJIA developed the Dow divisor, a number adjusted to account for events like splits that is used as the divisor in the calculation of the average.

The objective is to predict the DJIA by understanding the patterns of the index from the data containing everyday's *opening index values from 29-01-1985 to 08-02-2019*.

Exploratory Analysis

Data Description

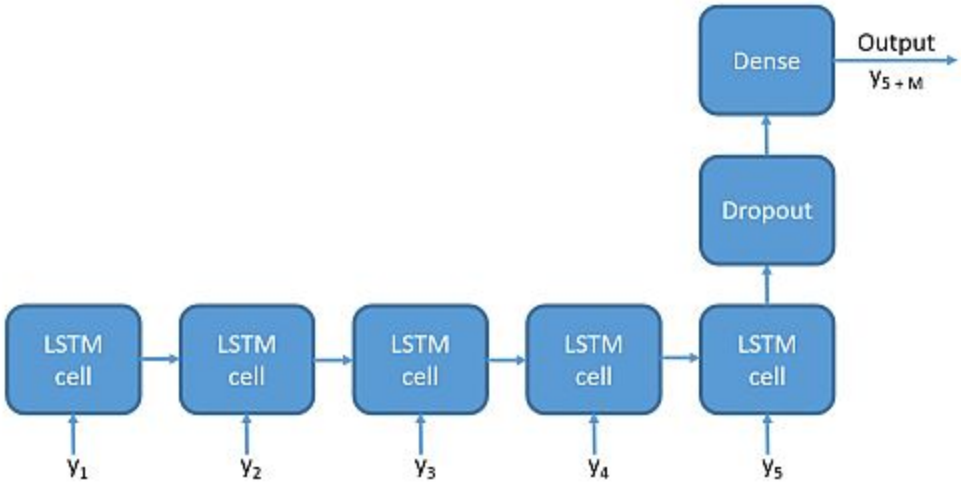
Data Source	https://www.dropbox.com/sh/47hp5s6b3fopyc5/AACYKQ8BHzqu6l_s_KjUujKja?dl=0
Sample Size	8578 Samples
Variables	7

After obtaining a summary of the data using pandas' describe() method (refer to Appendix 1), it is observed that Open has huge a range between the minimum and maximum values. Thus Feature scaling is carried out on the attribute using minmaxscaler (from scitkit-learn).

Analytics Approach

Approach

We predict the subsequent DJI values using a Recurrent Neural Network where we implement an LSTM model to understand the trends in the data. We provide a timestamp of 50 for the LSTM model which means that the model will take into consideration the 50 records before it to understand a trend. Hence, when we generate our x_train and y_train numpy arrays containing the training data,it will initially have a shape of (8491, 50) and (8491,) respectively.



Data Pre-Processing

Once we have our training data it is very important to know that the RNN architecture under Keras accepts an array only of a certain dimension , specifically 3D tensor with shape (batch_size, timesteps, input_dim). In order to convert our existing data to the desired shape, we use np.reshape() in such a way that the new shape is (8491, 50,1)

Building the Model

We initialize a Sequential model using Keras. The input layers are all LSTM layers containing 50 neurons each, whereas the output layer forms a Dense layer containing just one neuron.
We perform Dropout at every layer in order to prevent overfitting. We use the Adam optimizer instead of regular stochastic gradient descent and calculate our Mean Squared Error Loss at every epoch.
When the model is run, at every step, the y_train contains the data of the existing output row, and x_train contains the records of the 50 records before y_train. This way we are able to effectively iterate through the dataset and predict the output for each step.
The model was trained for 100 epochs and the model was stored in a SAV file using pickle in order to ease deployment of the model.

Results And Insights

The test set for the model was taken as the last 28 rows of the dataset. We did this because the data was time-series and at this point the model would have trained enough to predict the test data. (Refer to Appendix 2) There was a Mean Squared Error Loss of 3.1 after 100 epochs after performing some basic hyper-parameter tuning. The predicted values obtained from this model were not accurate enough to even nearly superimpose over the real values, although the model did comprehend the highs and lows and closely mimicked the same over the plotted graph.
The model can be improved by implementing hyper-parameter tuning methods such as Grid Search, or Bayesian Optimization.
NOTE: The model was run locally and on Google Colab and an encoding conflict was encountered by Python in the local environment. If that's the case during execution, it can be overcome by converting our input DataFrame to latin-1 encoding.

Appendix

1. Summary of data

	Open	High	Low	Close	Adj Close	Volume
count	8551.000000	8.551000e+03	8.551000e+03	8.551000e+03	8.551000e+03	8.551000e+03
mean	9669.178980	1.090763e+04	1.079251e+04	1.075096e+04	1.085054e+04	1.350484e+08
std	18398.762228	1.097781e+05	1.097771e+05	1.092601e+05	1.097775e+05	1.187795e+08
min	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.530000e+06
25%	3538.349976	3.550385e+03	3.524380e+03	3.539610e+03	3.538355e+03	2.595000e+07
50%	9807.490234	9.891060e+03	9.738430e+03	9.808040e+03	9.803050e+03	9.989000e+07
75%	12331.014650	1.240329e+04	1.226703e+04	1.233952e+04	1.233441e+04	2.219550e+08
max	1000000.000000	1.000000e+07	1.000000e+07	1.000000e+07	1.000000e+07	9.005100e+08

2. Loss