**Tri Nit Hackathon**

**Machine Learning ML – 04**

**Introduction**

Stock market remains to be a volatile domain. Its state can change any point of time. Predicting how the stock market will perform is one of the most difficult things to do. There are so many factors involved in the prediction – physical factors vs. psychological, rational, and irrational behavior, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy. Using features like the latest announcements about an organization, their quarterly revenue results, etc., machine learning techniques have the potential to unearth patterns and insights we didn’t see before, and these can be used to make unerringly accurate predictions.

**Problem Statement**

Forecasting stock market prices have always been a challenging task for many business analyst and researchers. Your friend, who is interested in investing in the stock market shares of the well-known company IBM is unable to predict the company's stock market. The rate of his investment and his business opportunities in IBM's Stock market can increase if an efficient algorithm could be

devised to predict the short - term price of an individual stock.

The link below contains a dataset, where the TIME\_SERIES\_DAILY\_ADJUSTED give the stock

market's close value of every day with a date. Your task is to devise a model to predict the 'adjusted close' value of the next day given the stocks of all days until the current day, and developer a front-end UI (either Web app or Mobile app) that can help your friend invest the right amount of money.

**Our Solution to the Problem Statement**

We have used LSTM as a part of our solution to the problem statement. We have split the given dataset into a Training set and a Test set, in which majority of the rows will be present in the Training set. We have created a model in which it will be trained to predict the stock market price (‘adjusted close’) of the 21st day based on the ‘adjusted close’ values of the 20 previous days. We have trained the model by doing this on the entire training dataset. The model first predicts the 21st day adjusted close value based on the 20 previous days, then compares it with the actual value of the 21st day. Then it tries to predict the adjusted close value of the 22nd day based on the previous 20 days again, then again it compares with the actual adjusted close value of the 22nd day. Then it does the same for the 23rd day and so on till it reaches the last day in the dataset. By comparing its predicted value with the actual value, and through training the model through multiple epochs, the model eventually decreases its error and is able to give a very close values of the “adjusted close” when it is tested on the test set. The model is able to predict the ‘adjusted close’ value of the next day given the stocks of all days till the current day, hence our solution to the problem statement. We have used root mean square error evaluation metric to check the error of our model.

**Code Explanation**

Text

Description automatically generated

Importing the required libraries such as numpy, pandas and matplotlib.



Importing the training dataset. Assigning the ‘adjusted close’ values in the Training dataset to the Training feature matrix.



Printing the training\_set feature matrix to check if the values have been imported correctly or not.

Text, letter

Description automatically generated

Feature scaling training\_set using the normalization method. This will make all the values in training\_set to be between 0 and 1. This will allow better training of the model.



Printing the scaled training\_set.

Text, letter

Description automatically generated

Here we are creating a data structure specifying what the LSTM will need to remember when predicting the next stock price, and this is called the number of time steps. It is important to have the correct number of timesteps as a wrong number could lead to overfitting or incorrect predictions. We have chosen 20 as our number of time steps. 20 timesteps means that at each time t, is going to look at 20 stock prices between 20 days before time t and time t, and based on the trends it is capturing during the previous 20 days, it will try to predict the next output (adjusted close value) i.e. the stock price at t+1.

Any number of time steps can be taken, choose the one which gives the best results.

x\_train contains the input of the LSTM and y\_train contains the adjusted close values.



Printing x\_train



Printing y\_train



We are increasing the dimensionality of x\_train to 3 using the reshape function. Before the x\_train has two dimensions i.e. the number of rows and number of columns. The 1 at the end indicates the number of indicators i.e. the Adjusted Close value. This concludes our data pre-processing phase.

Text, letter

Description automatically generated

Importing various functions which will help in building our LSTM model.



Naming our LSTM model as regressor and initializing the sequence of layers.



Adding the first layer to our model. We have chosen number of units as 50 in our first layer. Return sequence is chosen as true as we will be adding more LSTM layers. Third argument is the input shape to the model.

We have chosen the dropout regularization as 0.2 or 20% as it is a recommended and a classic value for LSTM.



Adding a second layer to the LSTM.



Adding a 3rd layer to the LSTM.



Adding the last layer to the LSTM. The default value of return\_sequence is False, that means that we are not adding any more layers to our LSTM.



Adding the output layer. Only one unit is present as the dimensionality of output is one.



It is recommended to use RMSprop optimizer for a LSTM model, but we haven’t used that, and instead used Adam since it is giving better results. The loss function chosen is Mean squared error.



Training our LSTM model on the training data. Number of epochs chosen are 100 and batch size is chosen as 10. The loss function is converging which indicates that the training of model is going well.



Importing the test data set adjusted close values.



Printing the adjust close values of the test data set.

Graphical user interface, text, application, email

Description automatically generated

Taking the latest 20 rows from the training set and appending them with the test set into a new variable called inputs. We need to do this as 20 rows from the training set is required by the model to predict the next day output, i.e. the 21st day and compare it with the value of the test data set. The last value in predicted\_stock\_price is the adjusted\_close value of the next day given the stocks of all days until the current day; this is the output we want.

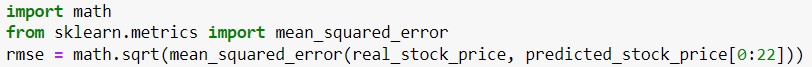


Printing the last value in predicted stock price which is our desired output.

Graphical user interface, application

Description automatically generated

Plotting a graph of the actual values and the predicted values of adjusted close value to compare between them. Note that actual stock price will have only 22 values and predicted stock price will have 23 values, as the model predicted for the next day as well.



Using the Root mean squared function to calculate the error between the actual and the predicted values of the LSTM model. We are calculating error for the 22 rows that are present in the test set and not for the 23rd (next day) value in the predicted stock price as it is a completely new value calculated by our model.



Printing the value of Root mean squared error.

Text

Description automatically generated

Printing the value of Mean Absolute Error.

**Explanation for app.py (website code)**

A picture containing background pattern

Description automatically generated

This portion of code imports necessary libraries for developing the web page and loading our pre-trained Machine Learning model. Followed by, using streamlit functionalities we add a drop bar feature from where we select the company for which we want to predict the stocks. The input field from the drop bar retrieves the dataset of the selected company’s stocks and displays a summary of the high, low, open and close values of the company’s stocks for the past 20 days.

Text

Description automatically generated

This portion of the code performs the data visualization by using streamlit and matplotlib libraries. We plot a graph between closing price vs Time chart. We estimate the moving averages of the closing price of our stocks by calculating the mean of the closing price of the stock for the previous 10 days. We do the same for 20 moving averages by calculating the mean of the previous 20 days of the closing price and plotting it on the same graph. Text

Description automatically generated

This portion of our code block does min-max scaling to the values in the dataset comprising of stocks of the company for which we wanted to know. Then we load our pre-trained ML model and partition the

dataset for the test set. We concatenate the predicted adjusted close of the train and test set and perform scaling and reshaping of inputs, and by using a loop we keep on appending a test data. We load the pre-trained ML model predict on test data and apply inverse transform of the min-max scaler to undo scaling of predicted stock market values.

Text

Description automatically generated

On this last block of our code, we plot the graph between the predicted stock market and the original stock market value of the company to give the user a clear vision of how close the prediction value is to the actual value of the company in the stock market is. In the end, we print the predicted and actual stock market value(adjusted close).

**Results**

**Graphical user interface, text, application

Description automatically generated**

RMSE = 7.017 and MAE=6.22

**Conclusion**

We have got a root mean squared error of 7.0.17 and mean absolute error of 6.22 which is considered as a less error values when it comes to time series prediction models, especially for stock market predictions. This shows the success of our LSTM model, and it also shows that the model can be deployed to practical world stock market prediction applications as well. Our model is displaying an output between 120 – 125 which is very close to the adjusted close values of the training set and the test set, hence we can say that our model is capable of identifying the trends in stock market values until time t, and is capable of predicting the stock market value of time t+1. We have tested our model on other stock price datasets as well such as ‘Green Motors’ and ‘PLC’ and we got less less RMSE and MAE values for these as well.

This concludes our solution to the given ML – 04 problem statement.