A Project Report

on

Stock Market Prediction System

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mr. C. Ramesh Kumar
Assistant Professor
Department of Computer Science and Engineering

Submitted By

Neerabh Kumar Nirala
18SCSE1140020

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA
DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **" STOCK MARKET PREDICTION "** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021,** under the supervision of **Mr. C. Ramesh Kumar Assistant Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other place.

18SCSE1140020 - NEERABH KUMAR NIRALA

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Mr. C. Ramesh Kumar, Assistant Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of, **18SCSE1140020 - NEERABH KUMAR NIRALA** has been held on _____ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

Stock Market Prediction System

ABSTRACT

The prediction of a stock market direction may serve as an early recommendation system for short-term investors and as an early financial distress warning system for long-term shareholders. Forecasting accuracy is the most important factor in selecting any forecasting methods. Research efforts in improving the accuracy of forecasting models are increasing since the last decade. The appropriate stock selections that are suitable for investment is a very difficult task. The key factor for each investor is to earn maximum profits on their investments. In this project Regression a machine learning technique and long short term memory technique is used. Abstract-In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Machine learning itself employs different models to make prediction easier and authentic. The paper focuses on the use of Regression and LSTM based Machine learning to predict stock values. Factors considered are open, close, low, high and volume. You would like to model stock prices correctly, so as a stock buyer you can reasonably decide when to buy stocks and when to sell them to make a profit. You need good machine learning models that can look at the history of a sequence of data and correctly predict what the future elements of the sequence are going to be. A correct prediction of stocks can lead to huge profits for the seller and the broker. Frequently, it is brought out that prediction is chaotic rather than random, which means it can be predicted by carefully analyzing the history of the respective stock market. Machine learning is an efficient way to represent such processes. It predicts a market value close to the tangible value, thereby increasing the processes. It predicts a market value close to the tangible value, thereby increasing the accuracy. Introduction of machine learning to the area of stock prediction has appealed to many researchers because of its efficient and accurate measurements.

TABLE OF CONTENTS

1.1 Introduction(Overview)

There are over 2.2 million Hong Kong stock investors, who contributed about 15% of the cash market trading value in 2016. The total cash market trading turnover is around HK$1.6 trillion.In particular, retail investors have made buy or sell investment decisions worth a total turnover of $240 billion for the year of 2016 [1]. In Hong Kong, there are a lot of investment decisions that involve a large sum of money being made.

Retail investors spend a lot of time finding investment opportunities. Wealthier investors could seek professional financial advisory services, but for typical retail investors, the costs are prohibitive. Thus, retail investors have to figure out the market themselves and make informed decisions on their own. This makes investment very stressful in modern societies.

Unfortunately, humans are irrational in nature. Without quantitative, data-driven models,decisions get swayed by cognitive biases or personal emotions, resulting in unnecessary losses.Even if investors are cautious enough, most do not have sufficient skills to process a huge volume of data required to make good judgments. Institutional investors rely on sophisticated models supported by technologies to avoid traps, but retail investors do not have access to such technologies and often find themselves falling behind the market.

Without access to quantitative and data-driven models, one obvious approach retail investors could use to evaluate the market is through simple indicators, for example, linear regression and exponential moving average (EMA) (Figure 1.1). Two important indicators are the 20-day EMA and the 50-day EMA. When the 20-day EMA rises above the 50-day EMA, the stock is likely to trend upward, and vice versa. Another obvious approach retail investors might use to predict the stock market is to draw a linear regression line that connects the maximum or minimum of candlesticks.

Inspired by the increasing popularity of deep learning algorithms for forecasting applications,these algorithms might serve as potential tools to find hidden patterns in the trend of stock prices. This information could be useful to provide extra insights for retail investors when making investment decisions. Therefore, this final year project

aims to investigate the usefulness of deep learning algorithms in predicting stock prices and democratize such technologies through an easy to use interface for the general public.

1.2 Objectives

1.2.1 Introduction

The ultimate goal of our application is to serve retail investors as a third party investment tool that uses machine learning to help them navigate in the fast-changing stock market. The project aims to introduce and democratize the latest machine learning technologies for retail investors.No prediction is 100% accurate. Therefore, the upper bound and lower bound of the stock prices will be displayed to illustrate the trading range the investors should be looking at. This application serves as a supplementary quantitative tool for investors to see the market at a different perspective with the help of technology.

This project is divided into 2 parts, namely a research component and an application component, aiming to provide retail investors with stock price predictions using different machine learning models in a good user experience way for reference.

1.2.2 Research

This project will investigate how different machine learning techniques can be used and will affect the accuracy of stock price predictions. Different models, from linear regression to dense and recurrent neural networks are tested. Different hyperparameters are also tuned for better performance.

The search space for all neural network architectures and hyperparameter combinations is huge, and with limited time in conducting this project, apart from manually trying different reasonable combinations, the team optimizes the models with evolution algorithm, replicating AutoML techniques from other researches with promising results in the financial context.

1.2.3 Application

This project aims to provide stock price predictions based on the latest machine learning technologies to all retail investors. A mobile web application is developed to provide predictions in an intuitive way. Different models' performance and accuracy can

also be compared. The application also serves as another user interface (UI) in visualizing results from the research apart from Jupyter notebooks with lots of tables and graphs.

1.3 Literature Survey

1.3.1 Stock Price Predictions

From the research paper “Machine Learning in Stock Price Trend Forecasting” written by Y. Dai and Y. Zhang in Stanford University, they used features like PE ratio, PX volume, PX EBITDA, 10-day volatility, 50-day moving average, etc. to predict the next-day stock price and a long-term stock price [2]. The machine learning algorithms used in the research are Logistic Regression, Gaussian Discriminant Analysis, Quadratic Discriminant Analysis, and SVM. The accuracy ratio is defined as the number of days that the model correctly classified the testing data over the total number of testing days. With the short term model predicting the next day stock price, it has very low accuracy, the Quadratic Discriminant Analysis is the best among all models, it scored a 58.2% accuracy. With the long term model predicting the next n days stock prices, the longer the time frame, the better in the accuracy for SVM. With a time window of 44 days, the SVM model’s accuracy reached 79.3%. Apart from that, it was found that by increasing the number of features, the accuracy increased. When all of the 16 features were used, the accuracy of the model reached 79%, while it fell to 64% when only 8 features were used, and 55% if only 1 feature was used. Our project will also investigate how the timeframe would affect the accuracy of price predictions of different models. As models have to reach a certain threshold to have significance for the users to work as a reference, it is essential for us to optimize our model to figure out what the optimal parameters and model structure are for our stock price prediction purpose.The research paper “Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques” written by J. Patel, S. Shah, P.Thakkar, and K. Kotecha for the “Expert Systems with Applications” international journal demonstrated a way to use trend deterministic data to predict stock price movement [3]. They conducted experiments in using 10 technical indicators’ signals as inputs, then they use prediction models to predict whether the stock will go up or down in the coming 10 days, Technical analysis indicators include SMA, EMA, Momentum, Stochastic SK, Stochastic SK, MACD, RSI, etc. The prediction models they have used include ANN, SVM, Random Forest, and Naive Bayesian models. The model outputs “up” or “down” movement signals. Experiments have shown random forest scored the highest performance with 83.56% accuracy with their inputs.

B. Wanjawa and L. Muchemi demonstrated the potential in predicting stock prices using ANN,as shown in the research paper “ANN Model to Predict Stock Prices at Stock

Exchange Markets”[4]. They used 70% of the training data to predict the stock prices for the next 60 days. Through Optimizations, they were able to predict the actual closing prices within 0.71% mean absolute percentage error (MAPE), with the highest variance -3.2% among all of the 62 days. This Demonstrated a high potential for using machine learning to accurately predict stock prices.This is one of the key components in our application where algorithms have to be designed to have high accuracy, such that the platform could be useful for retail investors.

1.3.2 Neural Network

A neural network attempts to learn a function that maps the input features to the output predictions, serving as a universal function approximator [5]. It consists of a network of neurons, each of which represents a weighted sum of inputs. Outputs from neurons are fit into activation functions which introduce non-linearity to the system, and then passed to some other neurons. In a typical dense feedforward neural network, the network consists of layers of neurons stacked together, with neurons between individual layers fully connected.

Optimization of neural networks is usually done through backpropagation with gradient descent, which essentially propagates the error from the output layer back to the input layer,while computing the gradient of the error against each parameter in the process.

1.3.3 Recurrent Neural Network

Recurrent neural network [5] is a type of neural network where connections between neurons allow temporal, sequential information to be stored and processed in the network. One typical architecture is formed by feeding the output of the current unit back to the input with a time delay so that the network can use the information in processing the next input. Various techniques have been developed over the years to train such a type of network. One of the popular approaches is backpropagation through time (BPTT) [6], whose central idea is to unroll the recurrent network into a feedforward network, where each layer represents a timestep. Backpropagation with gradient descent could then be performed to optimize the network, just like how we optimize a feedforward network. Unfortunately, it has been shown that techniques like BPTT result in either vanishing or exploding gradients [7]. Vanishing gradients lead to unrealistically

long training time, and sometimes training is infeasible while exploding gradients result in fluctuating weights, which leads to unstable training. Both are undesirable in neural network training. Thus, new training methods and architectures are needed to mitigate the problems.

1.3.4 Long Short-Term Memory (LSTM)

Long short-term memory [8] was first introduced by Hochreiter and Schmidhuber in 1997 to address the aforementioned problems. Long-short term memory tackles the problem of learning to remember information over a time interval, by introducing memory cells and gate units in the neural network architecture. A typical formulation involves the use of memory cells, each of which has a cell state that stores previously encountered information. Every time an input is passed into the memory cell, and the output is determined by a combination of the cell state (which is a representation of the previous information), and the cell state is updated. When another input is passed into the memory cell, the updated cell state and the new input can be used to compute the new output.

1.3.5 Gated Recurrent Unit (GRU)

Gated recurrent unit [9] follows the same architecture as long short-term memory, except that it simplifies the design of the memory cell, by reducing the structure to contain only two gates,the reset gate, which controls how much information to forget when taking in the new information, and the update gate, which controls the proportion of cell state updated by the contribution. Although it has been shown that LSTM is more powerful than GRU [10], GRU has the advantage of lower training time and may perform better on smaller datasets [11].

1.3.6 Evolution Algorithm

Researches have shown that large-scale evolution can auto-generate neural network model architectures and hyperparameters with performance comparable with state-of-the-art human-designed models. In a research in 2017 [12], a large-scale evolution for discovering image classification neural networks was run. It started with a huge population of randomized simple 1-layer models, then slowly evolved the population by removing a poor model and generating a new model by mutating some

parameters of a good model in each iteration. After hundreds of hours of running the algorithm with huge computing power, most models in the population achieved state-of-the-art results on CIFAR datasets. In each iteration, only a simple mutation that changed 1 parameter was applied, which allowed searching in a large search space. The paper showed the possibility of finding good models by using lots of computational power to replace human-machine learning experts and has set the foundation of democratizing machine learning with AutoML.

2.1 PROBLEM FORMULATION

Investors are familiar with the saying, "buy low, sell high" but this does not provide enough context to make proper investment decisions. Before an investor invests in any stock, he needs to be aware of how the stock market behaves. Investing in a good stock but at a bad time can have disastrous results, while investment in a mediocre stock at the right time can bear profits. Financial investors of today are facing this problem of trading as they do not properly understand which stocks to buy or which stocks to sell in order to get optimum profits. Predicting long term value of the stock is relatively easier than predicting on a day-to-day basis as the stocks fluctuate rapidly every hour based on world events.

We aim to predict the daily adjusted closing prices of Vanguard Total Stock Market ETF (VTI), using data from the previous N days (ie. forecast horizon=1). We will use three years of historical prices for VTI from 2015-11-25 to 2018-11-23, which can be easily downloaded from yahoo finance.

We will split this dataset into 60% train, 20% validation, and 20% test. The model will be trained using the train set, model hyper parameters will be tuned using the validation set, and finally the performance of the model will be reported using the test set. Below plot shows the adjusted closing price split up into the respective train, validation and test sets.

To evaluate the effectiveness of our methods, we will use the root mean square error (RMSE) and mean absolute percentage error (MAPE) metrics. For both metrics, the lower the value, the better the prediction.

2.2 Last Value

In the Last Value method, we will simply set the prediction as the last observed value. In our context, this means we set the current adjusted closing price as the previous day's adjusted closing price. This is the most cost-effective forecasting model and is commonly used as a benchmark against which more sophisticated models can be compared. There are no hyperparameters to be tuned here.

2.3 Moving Average

In the moving average method, the predicted value will be the mean of the previous N values. In our context, this means we set the current adjusted closing price as the mean of the adjusted closing price of the previous N days.The hyperparameter N needs to be tuned.

3.1 REQUIRED TOOLS

- JUPYTER NOTEBOOK
- DATASET FROM YAHOO FINANCE

3.2 PYTHON LIBRARIES

- Pandas
- Numpy
- Scikit Learn
- Matplotlib
- Pandas datareaders
- Keras
- Math

These following Libraries can be installed by pip command in the terminal and can be used using the import_function.

COMPLETE WORK PLAN LAYOUT

DATASET INFORMATION

Stock prices come in several different flavors. They are,

- Open: Opening stock price of the day
- Close: Closing stock price of the day
- High: Highest stock price of the data
- Low: Lowest stock price of the day

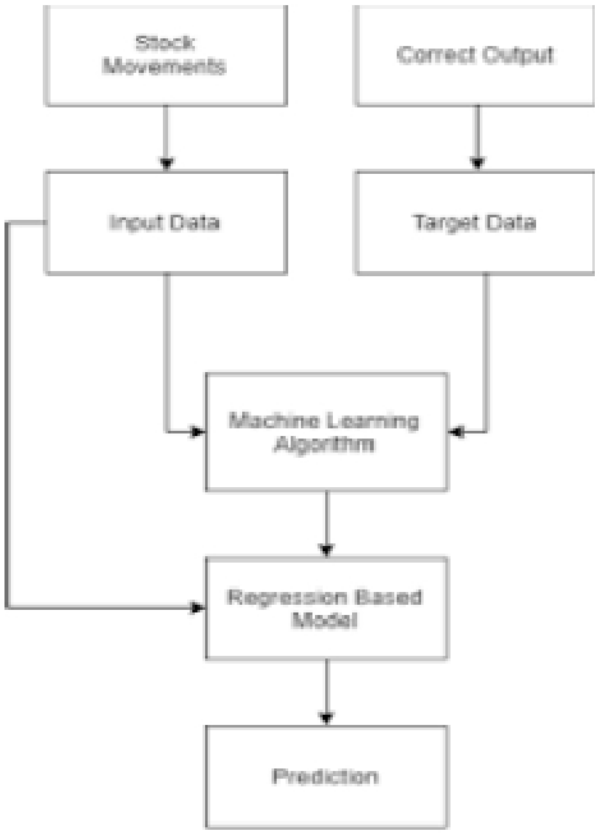
MODEL 1

Stock market prediction seems a complex problem because there are many factors that have yet to be addressed and it doesn't seem statistical at first. Butby proper use of machine learning techniques, one can relate previous data to the current data and train the machine to learn from it and make appropriate assumptions. Machine learning as such has many models but this paper focuses on two most important of them and made the predictions using them.

$$V=a + bk + error$$

Regression is used for predicting continuous values through some given independent values . The project is based upon the use of a linear regression algorithm for predicting correct values by minimizing the error function as given in Figure1. This operation is called gradient descent. Regression uses a given linear function for predicting continuous values: Where,V is a continuous value; K represents known independent values; and, a, b are coefficients. Work was carried out on csv format of data through the panda library and calculated the parameter which is to be predicted, the price of the stocks with respect to time. The data is divided into different train sets for cross validation to avoid
over-fitting. The test set is generally kept 20% of the whole dataset. Linear regression as given by the above equation is performed on the data and then predictions are made, which are plotted to show the results of the stock market prices vs time

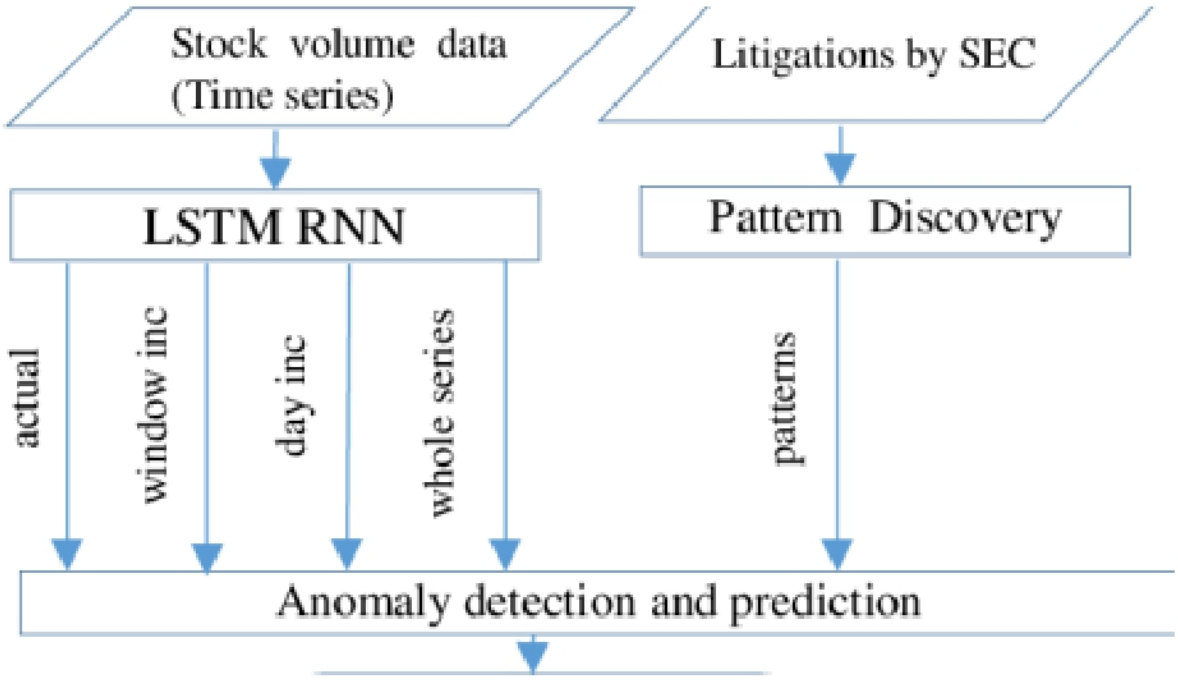
FLOW CHART


Stock Movements
Correct Output
Input Data
Target Data
Machine Learning Algorithm
Regression Based Model
Prediction

MODEL 2 based on LSMT

LSTM is the advanced version of Recurrent-NeuralNetworks (RNN) where the information belonging to previous state persists. These are different from RNNsas they involve long term dependencies and RNNs works on finding the relationship between the recent and the current information. This indicates that the interval of information is relatively smaller than that of LSTM. The main purpose behind using this model in stock market prediction is that the predictions depend on large amounts of data and are generally dependent on the long term history of the market . So LSTM regulates error by giving an aid to the RNNs through retaining information for older stages making the prediction more accurate . Since stock market involves processing of huge data, the gradients with respect to the weight matrix may become very small and may degrade the learning rate of the system. This corresponds to the problem of Vanishing Gradient. LSTM prevents this from happening. The LSTM consists of remembering the cell, input gate, output gate and a forget gate. The cell remembers the value for long term propagation and the gates regulate them . In this paper,a sequential model has been made which involves stacking two LSTM layers on top of each other with the output value of 256. The input to the layer is in the form of two layer and layer. A dropout value of 0.3 has been fixed which means that 0.3 out of total nodes will be frozen during the training process to avoid overfitting of data and increase the speed of the training process. At last, the core dense layer where each neuron is connected to every other in the next layer is added providing input of 32 parameters to the next core layer which gives output as 1. The model is compiled with a mean square cost function to maintain the error throughout the process and accuracy is chosen as a metric for the prediction.

FLOW CHART



STEPS IN PREPARATION OF MODELS

- Download data from yahoo finance
- Data exploration using pandas
- Splitting Dataset into training and test sets
- Normalizing the data using MINMAXSCALER
- Predicting by Averaging
- Defining hyperparameters
- Defining inputs and outputs
- Parameters for LSTM and regression
- Calculating LSTM OUTPUT and feeding it to regression layer to get final prediction
- Predict related calculations
- Visualizing the prediction

IMPLEMENTATION

```
In [44]: import math
         import pandas_datareader as web
         import numpy as np
         import pandas as pd
         from sklearn.preprocessing import MinMaxScaler
         import matplotlib.pyplot as plt
```

```
         # LOADING THE DATASET
```

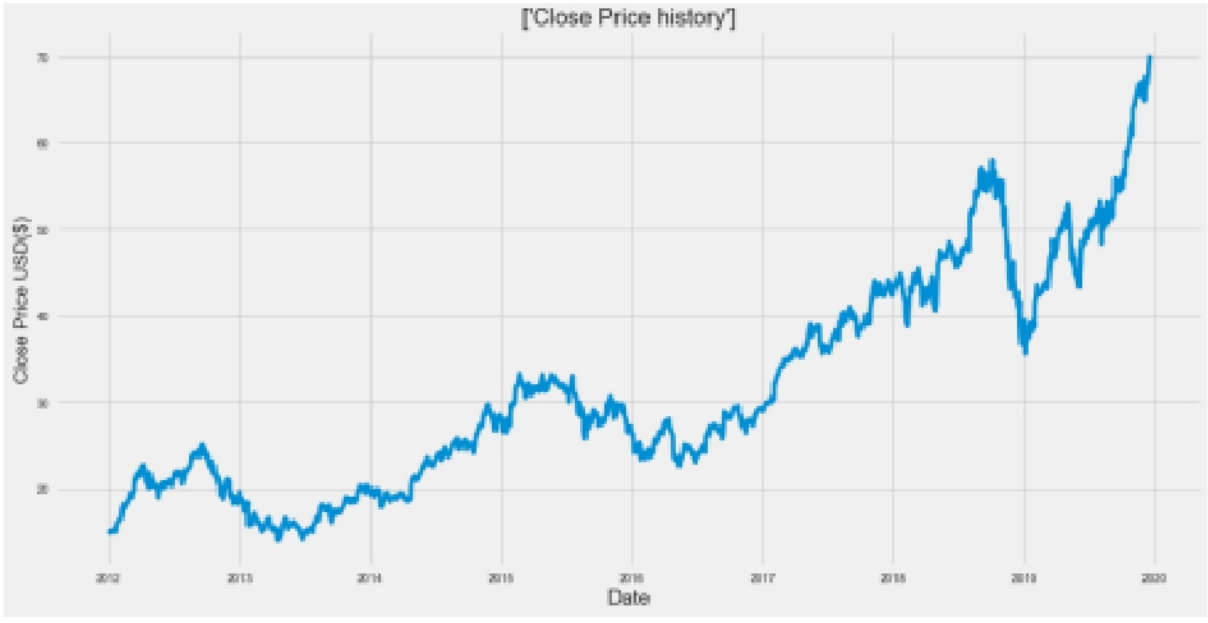
```
In [14]: df=web.DataReader('AAPL',data_source='yahoo',start='2012-01-01',end='2019-12-17')
         df
```

Date	High	Low	Open	Close	Volume	Adj Close
2012-01-03	14.732142	14.607142	14.621428	14.686786	302220800.0	12.891425
2012-01-04	14.810000	14.617143	14.642858	14.765715	260022000.0	12.959831
2012-01-05	14.948215	14.738214	14.819643	14.929643	271269600.0	13.103293
2012-01-06	15.098214	14.972143	14.991786	15.085714	318292800.0	13.036158

```
In [15]: df.shape

         (2003, 6)
```

```
In [10]: #VISUALIZE THE CLOSING PRICE HISTORY
```

```
In [19]: plt.figure(figsize=(16,8))
         plt.title(['Close Price history'])
         plt.plot(df['Close'])
         plt.xlabel('Date',fontsize=18)
         plt.ylabel('Close Price USD($)')
         plt.style.use('fivethirtyeight')
         plt.show()
```



In [20]:
```
#create a new dataframe with only the close column
```

In [21]:
```
data=df.filter(['Close'])
dataset=data.values
```

In [25]:
```
training_data_len=math.ceil(len(dataset)* .8)
training_data_len
```

```
1603
```

In [26]:
```
#SCALE THE DATA
```

In [27]:
```
scaler=MinMaxScaler(feature_range=(0,1))
scale_data=scaler.fit_transform(dataset)
scale_data
```

```
array([[0.01316589],
       [0.01457866],
       [0.01348085],
       ...,
       [0.99755134],
       [1.        ]])
```

In [ ]:
```
#Create the training dataset
#Create the Scaled training data set

train_data=scale_data[0:training_data_len,:]

#spliting the data into X_train and Y_train sets
x_train=[]
y_train=[]

for i in range(60,len(train_data)):
    x_train.append(train_data[i-60:i,0])
    y_train.append(train_data[i,0])
    if i<=60:
        print(x_train)
        print(y_train)
```

In [33]:
```
#converting into the numpy array
x_train,y_train=np.array(x_train),np.array(y_train)
```

reshape data int 3 D dimnesion as lstm accepts 3 Dinmension values

In [38]:
```
x_train.shape
```

```
(1543, 60)
```

In [41]:
```
x_train=np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
x_train.shape
```

```
(1543, 60, 1)
```

In [55]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import LSTM
```

Build the LSTM model

In [57]:
```
model=Sequential()
model.add(LSTM(50, return_sequences=True, input_shape= (x_train.shape[1],1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

In [58]:
```
#Compile the model
model.compile(optimizer='adam',loss='mean_squared_error')
```

In [59]:
```
model.fit(x_train,y_train,batch_size=1,epochs=1)
```

```
1543/1543 [==============================] - 18s 12ms/step - loss: 7.3970e-04
```

In [60]:
```
#creating the testing the dataset
#Create a new array containing scaled values from index 1543
```

In [96]:
```
test_data=scale_data[training_data_len - 60: , :]
```

In [97]:
```
#Create the data sets x_test and y_test
x_test=[]
y_test=dataset[training_data_len:,:]

for i in range(60,len(test_data)):
    x_test.append(test_data[i-60:i, 0])
```

In [98]:
```
x_test=np.array(x_test)
```

In [99]:
```
x_test= np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
```

get the model prediction

In [100]:
```
predictions=model.predict(x_test)
predictions=scaler.inverse_transform(predictions)
```

get the root mean squared error(RMSE)

In [101]:
```
rmse=np.sqrt(np.mean(predictions-y_test)**2)
rmse
```

```
2.0013076514654543
```

plot the data

In [102]:

```
train=data[:training_data_len]
valid=data[training_data_len:]
valid['Predictions']=predictions
```

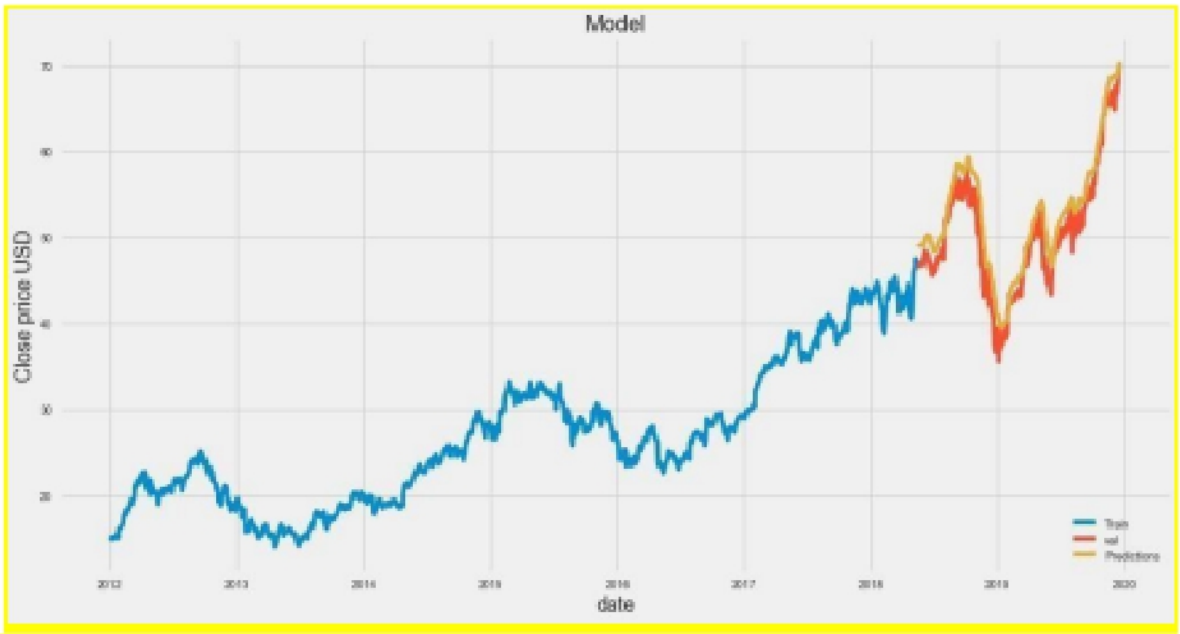
```
c:\python37\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

visualize the data

In [104]:

```
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('date',fontsize=18)
plt.ylabel('Close price USD',fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Train','val','Predictions'],loc='lower right')
```

PLOTING THE GRAPH FOR PREDICTED PRICE WITH ACTUAL VALUE



In [105]:
```
valid
```

Date	Close	Predictions
2018-05-17	46.747501	48.874680
2018-05-18	46.577499	48.959053
2018-05-21	46.907501	48.975101
2018-05-22	46.790001	48.999039
2018-05-23	47.090000	49.007614
...	...	...
2019-12-11	67.692497	69.118729
2019-12-12	67.864998	69.347015
2019-12-13	68.787498	69.590561
2019-12-16	69.964996	69.928429
2019-12-17	70.102501	70.411484

400 rows x 2 columns

PREDICTING THE CLOSING PRICE USING LAST 60 DAYS DATA

In [106]:
```
#get the qoute
apple_quote=web.DataReader('AAPL',data_source='yahoo',start='2012-01-01',end='2019-12-17')
#create a new dataframe
new_df=apple_quote.filter(['Close'])
#get the last 60 day closing price values and convert the dataframe to an array
last_60_days=new_df[-60:].values
last_60_days_scaled=scaler.transform(last_60_days)
```

In [107]:
```
#create an empty list
X_test=[]
#append the last 60 lasts
X_test.append(last_60_days_scaled)
#convert the X_test set to numpy array
X_test=np.array(X_test)
#reshape the data
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
#get the predicted price
pred_price=model.predict(X_test)
#undo scaling
pred_price=scaler.inverse_transform(pred_price)
print(pred_price)
```

```
[[70.909615]]
```

4.Methodology

4.1 Unit Test

The unittest module from Python [37] is used to implement all unit tests, as it is available by default in Python and integrates well with existing Python codes.

Unit tests are done for the build dataset script, which transforms the raw input data into feature vectors usable for training and testing, as well as model score calculations. Unit tests are conducted because the components are error-prone, calculation intensive. Also, they exhibit garbage-in-garbage-out properties, that the model will be completely wrong if it receives the wrong input, and if the model scores are wrong, the final buy-sell recommendation will be totally incorrect.

In particular, unit tests are written for the functions to build the dataset for training and prediction and the function to build the snakes. Combinations of input options are tested,including n-day stock price lookback as well as n-day moving average. Correctness is ensured by asserting the feature vectors' shapes, as well as starting and ending elements.

For model score calculation unit tests, different scenarios are emulated, including the case when the model accurately predicts all the stock prices, the case when the model predicts all the stock prices wrongly by a very large magnitude, the case when the model predicts the trend correctly but underestimates the trend, as well as the case when the model predicts the trend correctly but overestimates the trend.

4.2 Tools Used for Testing

Various tools have been used to assist in the development of the mobile application. In Particular, Chrome Mobile Emulator is used to simulate the mobile view while developing the mobile application on desktop/laptop computers. After the application is deployed to the cloud, mobile phones with different operating systems and browsers, including Google Pixel running Android 9 (Google Chrome) and iPhone 7 running iOS 12.1 (Safari), are used to verify the user experience is consistent across different devices with different resolutions.

5. Methodology - Evaluation

The project's objective is to provide a third-party investment tool to investors with democratized machine learning technologies. The success of the project is primarily determined by two factors, namely, whether the investment tool provides useful, accurate stock price predictions to investors, and whether investors can use and understand the predictive information provided by the machine learning technologies. The first factor is evaluated by the model scores described in 2.2.7. However, the evaluation of the second factor is based on user experience. External users have to be involved in the evaluation. For this purpose, hallway testing is used.

Hallway testing involves allowing users who have not been involved in the development of the project to test the application and give constructive feedback about how users feel about the application. Users participating in the tests are asked a set of questions about the usability and whether they understand the information presented by the mobile application. This would give indications about whether the democratization of the machine learning technologies succeeds.

6. Findings

6.1 General Findings

The following are some general findings from testing out different machine learning models.it shows that the 1-day interval historical predictions line follows closely with the historical prices. The graph looks like the prediction line is just 1 day shifting from the historical prices, similar to a shifted and smoothed out historical prices line. Therefore, the shape of the historical predictions line is similar to the shape of the exponential moving averages (EMA), where the price changes from t to t+1 heavily depend on the direction and magnitude of changes from t-1 to t, followed by decreasing importance from earlier historical prices. Other models in predicting stock prices of other stocks also show similar results.

It shows that the 10-day interval historical predictions line does not follow closely with the historical prices but could demonstrate the trend. For example, historical predictions 1, 2, 3, 4, 7, 8, 9, 10 provided insights on the correct market direction, yet the magnitude did not match the actual price movements. A possible reason for this error can be the 10-day interval prediction has to predict more values while having less data compared to the case of 1-day interval prediction, which for 1-day interval prediction, data of close prices until previous day are available. Therefore, a longer period of interval prediction could be subject to greater changes in market fundamentals, including market news, macroeconomic factors,earning reports, etc. Other models in predicting stock prices of other stocks also show similar results.Although price reflects all available information, the magnitude of price changes in the future might need other data for forecasting purposes, such as market sentiment, company announcement, retail and institutional investors' attention on the company, etc. This is one of the possible explanations of why the 10-day interval prediction might have a large difference to actual values as there are potential shifts in market momentum. Therefore, the price might be too compact and other information is required to make a more accurate prediction.

2 scores are used to measure the performance of historical predictions, trend score and accuracy score, introduced in 2.2.7. The higher the trend score means that the model is more accurate in trend prediction and could provide more meaningful price movement direction insights. The score representations used in this application could be useful for the user to interpret the errors of predictions in a quantifiable way. The higher the accuracy score means that the model could follow the actual stock prices more accurately. It shows that all best models generated from the evolution algorithm experiment have a trend score ranging from 6-7 but have an accuracy score ranging from 1-2 on the test set. This finding matches the earlier findings, that the trend could be predictable, especially for less volatile stocks, but exact price, especially further into the future, could hardly be predicted accurately.Despite common research findings that recurrent neural networks in general perform better than dense feedforward neural networks at predicting time-series data such as stock prices, in this project feedforward neural network outperforms recurrent neural networks. One possible explanation is that training a recurrent neural network requires more data than the dense neural network in general, as recurrent neural networks have more parameters. As the models are trained using only daily stock prices dating back 20 years (or less if the stock is listed fewer than 20 years), there might not be enough data for training the recurrent networks to a good performance.

6.2 Prediction Approach Findings

As mentioned in 2.2.1, 2 approaches are tested in predicting the stock prices for the next 10 days, predicting all 10-day stock prices directly and predicting each stock price one at a time.The 2 different approaches frame the problem totally differently, which introduces a significant language bias.

According to the results (e.g. Figure 6.2a and 6.2b), for most stocks, most models that predict 10-day stock prices directly have a higher error than predicting individual stock price. However, the errors in predicting different days in the future are relatively constant for models that predict 10-day stock prices directly, while the error increases with the time from now for models that predict stock prices one day at a time.

One possible explanation for such observation is that the 2 problem framing approaches drive the model to learn different abstractions. For models that predict 10-day stock prices directly, it will learn the abstraction over 10 days. It is assumed that the correlation between the predicted stock price and today's and earlier stock prices decreases when predicting the future. Since the learned abstractions need to be applicable throughout 10 days, the high error from further prediction because of low correlation is propagated to other closer predictions. It results in a constantly higher error for predicting all days.

On the other hand, predicting stock prices one at a time allows the model to learn the relationships between more correlated data points. It can be observed from the results that the first-day prediction is more accurate compared to the first prediction from models that predict 10 days directly.

However, not all 10-day predictions have a lower error, the error for further predictions are higher. As mentioned in 6.1, most models, especially those predicting stock price individually, behave like an EMA, which puts more emphasis on more recent historical prices. Although this allows the model to accurately trace recent price movements, when predicting future stock prices iteratively the next predicted stock price is most strongly influenced by the

previous prediction instead of real data. This results in a reinforcement effect where predictions further ahead reinforce the unverified trend that the model predicts, and the errors amplify and propagate to subsequent predictions.

6.3 Accuracy Findings

6.3.1 Baseline Investor

The baseline for model accuracy comparison is from a hypothesized investor who adopts a trading strategy of predicting the stock price will either go up or down by the holding period return calculated from historical data. There will also be a corresponding error for this strategy. Assume the hypothesized investor always correctly predicts the stock price movement direction. The baseline strategy error is defined as:

If the hypothesized investor always predicts the wrong stock price movement, then the error is the maximum of the 2 terms.

6.3.2 Findings

The errors from 6.2 are compared with the baseline strategy introduced (Figure 6.3a and 6.3b).It is found that most models for most stocks achieve comparable performance in terms of error as the baseline strategy. Some models for some stocks have a slightly lower error than the baseline, and some have higher.Since the baseline error is calculated based on the assumption that the hypothesized investor always makes a correct prediction on the price movement direction, despite only having marginal improvement on the accuracy or even lower in some cases, the machine learning models trained successfully predicted the trend of the price movement, which agrees with the

6.4 Model Architecture and Hyperparameters Search with

Evolution Algorithm Findings

The evolution algorithm experiment conducted has shown promising results in searching model architectures and hyperparameters. Multiple experiments are run, for different stocks, different neural network types and different inputs. From the prediction error recorded over each evolution iteration all experiments have shown that the evolution algorithm successfully finds better models over time. Hand-designed models based on the team's intuition and basic knowledge could not achieve an error rate lower than that achieved by the algorithm's explored models. One interesting and unexpected observation from the evolution algorithm results is that a number of best models found are fairly simple. The found models are 1 or 2 layers deep with a linear activation function. In the case of having stacks of linear layers, the model is mathematically equivalent to a linear regression over features. There are multiple possible explanations for this observation.

First, the evolution algorithm hyperparameters used limits the search space for possible model architectures and hyperparameters. Due to computational power constraints, only a small population with 10 models is used. This limits the variety of models explored by the algorithm as the variance within the population is small. Moreover, under constraints, each experiment is run for 100 iterations only, which also limits the exploration. On average, the whole population is only 10 steps or mutations away from the original random population, which may not be significant enough for deep exploration. The evolution algorithm itself, together with its hyperparameters, introduces a search bias, the algorithm defines the possible explored models and the search path to achieve them.

Another possible explanation is that the dataset size is relatively small with just daily stock prices. Larger deep neural networks with more complicated architectures and thousands or even millions of weights require much more data to train and learn from.

A final possible explanation is that the stock market is at least weakly efficient, i.e., stock prices follow random walk given historical price data, and patterns with predictive power could not be found just from raw price data. If stock prices follow a random walk, a good predicting method is to put strong weights at very recent prices, and hope the actual price will fluctuate closely around it, which is very similar to a linear model.

Only using stock price data and simple derivatives like moving averages introduces a language bias, as price movements are also highly dependent on news and sentiment. Although stock prices reflect information, it is a very compact representation of all information and news. Thus, it is difficult to reverse engineer features or information out from a single number, especially when only daily stock prices are available. Having other information like real-time news sentiment or summary may help to break stock prices down to more granular components for machine learning algorithms to learn from.

6.5 Other Findings

6.5.1 Trend lines

The accuracy of the predictions based on linear trendlines fluctuate very wildly, as they are simply linear interpolations, while real stock prices may fluctuate up and down. In particular, the accuracy of the predictions depends completely on the choice of the day based on which the linear interpolations are made. Since the choice is arbitrary, the predictions based on trend lines are not reliable at all.

6.5.2 Alternative Prediction Method - Skip Predict

To tackle the problem of input bias on the intermediate prediction result and the short term noise of the stock, an alternative prediction method “skip predict” is used.

This method has 2 key advantages. First, this method can decouple the dependency of the prediction result based on the previous day in the original model. With “skip predict”, the input data of n-day before is used for the prediction. For example, if the number of days skipped is 10, this represents that input data would not consider the recent 10 days, and the input data would use the shifted time frame.

Second, the prediction result can all depend on the historical prices and not the intermediate prediction result. This could be one of the methods to solve the reinforcement problem mentioned in 6.2. The error of the first predicted data point would not impact the following predictions result. For example, the incorrect trend of the first predicted data point (t+1) would not serve as the input for the predictions later (t+2, t+3, ..., t+10). This creates an advantage that the error or bias would not accumulate and the result could solely depend on historical data. The hypothesis is that such a method could generate a lower root mean square error compared to the original model the application is using.

6.6 Mobile Application User Experience Testing

To evaluate the user experience of the mobile application, users who have not been involved in the development of the application have been invited to try out the mobile application and give constructive feedback. The major findings are summarized as follows:

6.6.1 Useful Insights for Finding General Trend

Despite the flaws found in the mobile application, users in the test agree that they were able to check out what are the possible movements of the stock prices predicted by the machine learning models and the general directions of the stocks. Users find it might be useful for finding stocks with upside potential for the coming few days.

6.6.2 Unclear Description of the Models

In the mobile application, different models are named after their architectures, such as LSTM, Dense Neural Network, and GRU. However, these technical names are not familiar to users who have no experience in machine learning and cause some confusion among users.

6.6.3 Unclear Presentations of the Prediction Results

Each stock is associated with a model trend score as described in 2.2.7. However, to laymen users, it might not always be clear what these scores represent, as the definitions are not clearly explained. The lack of clarity might confuse users or lower their confidence as they attempt to take actions following the predictions made by the models.

7. Discussion

As mentioned in 5, the success of the project is primarily determined by two factors, namely, whether the investment tool provides useful, accurate stock price predictions to investors, and whether investors can use and understand the predictive information provided by the machine learning technologies. The project's objectives are therefore partially fulfilled.

7.1 Accuracy of Stock Price Predictions

As shown in 6.1, while the 1-day stock price prediction follows closely with actual stock prices, the predictions for stock prices after 10 days deviate considerably from the actual stock prices. This shows that machine learning models fail to provide accurate stock price predictions to retail investors.

Nevertheless, some of the models have been shown to outperform predictions based on random walks as mentioned in 6.2, and therefore might still serve as a reference for more savvy investors, who might be able to compare the results with their own analysis findings to discover meaningful trends.

7.2 Democratization of Machine Learning Technology

Another factor when evaluating the project's success is whether investors can use and understand the predictive information provided by the machine learning technologies using our mobile application. In spite of the confusions found in some parts of the user interface, especially in the advanced user mode, users found useful insights provided by the machine learning models, such as identifying stocks with upside potential. The result is significant, in the sense that users with little background on machine learning technology and stock trading could find potential use cases for the application. The results imply that machine learning technologies could be democratized to serve the interest of the general public. Stock price prediction is a particularly exciting area, because the level of

expertise required to succeed in making profitable short-term investments is considered to be prohibitive for small, retail investors, and trading with help of machine learning is a feat only institutional investors could perform. The application demonstrates one possible way retail investors could use machine learning technologies on their own.

Limitations and Future Scope of the Project

1) Machine-learning methods HAVE been successfully used by various individuals and institutional 'in-house' groups, but most 'public' individuals, such as yourself, will NOT learn of 'THE' SPECIFIC methodologies that have yielded 'lucrative' returns and results.When 'huge' money is involved, and this IS the case when 'dealing with' the financial markets, NO ONE is going to publicly 'share' their 'edge' derived from applying THEIR successful methods to trading hence, you're not likely to hear of, nor see, detailed studies and reports of such successes.

2) MOST 'academic' researchers who publish papers attempting to apply computer processing algorithms to trading markets simply do NOT truly UNDERSTAND the underlying 'dynamics' of market price behaviors, so 'naive' applications of methodologies are attempted and 'researched', with the result that 'less than stellar' outcomes are generated frequently. To be 'effective' in developing 'successful' trading methods requires a rather 'deep' understanding of 'general underlying dynamic behaviors' of what makes the markets 'tick'.

3) Markets (stocks, futures, forex, options, etc) generate data that form (statistically) NON-STATIONARY, time-series of numbers over ANY period of 'time window' that one may want to examine, 'forecast' upon, and trade. 'Prediction' (which is highly 'precise') is essentially impossible, but to a greater or lesser degree, 'forecastability' (less 'precise', but more 'probabilistic') IS applicable to market time-series data, with the exception of what are called 'event shocks', such as USA's 9/11, October of 1987, 'flash crashes', and similar types of 'events'. (From a 'risk-management' standpoint, any 'good' and 'effective' trading strategy/system MUST make provision for such occurrences in order to protect trading capital and prevent financial 'disaster'!)

4) From an engineering (and computer science) perspective, a 'trading system' can be 'thought of' as a 'combined' mathematical/logical TRANSFORM that uses 'appropriately conditioned' time-series 'market' data as input and then attempts to 'functionally' convert this input into a monotonically-increasing 'capital-capture' output time-series. Before attempting to EFFECTIVELY design such a 'transform', one MUST have a relatively 'decent' understanding of the characteristics AND 'character' of the time-series 'input' data to which the 'transform' is to be applied.....MOST researchers don't have an adequate, NOR realistic,market-dynamics UNDERSTANDING hence, their market MODELS are 'inadequate' and THIS is another reason why you rarely see public information of 'successful' machine-learning methods as applied to trading the markets.

9. Conclusion

The scope of Machine Learning is not limited to the investment sector. Rather, it is expanding across all fields such as banking and finance, information technology, media & entertainment, gaming, and the automotive industry. As the Machine Learning scope is very high, there are some of the areas where researchers are working toward revolutionizing the world for the future. Lastly here, but not finally, patterns do frequently recur in market oriented time- series data that can be exploited when designing a transform such as mentioned in the previous paragraph. These patterns and features can certainly,and effectively be discerned by means of machine-learning methods.

10. References

[1] “Survey Finds Hong Kong Securities Market Attracts Wide Range of Investors,” HKEX, 13 July 2017; http://www.hkex.com.hk/news/news-release/2017/170713news?sc_lang=en.

[2] Y. Dai and Y. Zhang, “Machine Learning in Stock Price Trend Forecasting,” Stanford University;

http://cs229.stanford.edu/proj2013/DaiZhang-MachineLearningInStockPriceTrendForecasting.pdf.

[3] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, “Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques,”

Expert Systems with Applications: An International Journal, Vol. 42, Jan. 2015, pp. 259-268

[4] B. Wanjawa and L. Muchemi, “ANN Model to Predict Stock Prices at Stock Exchange Markets,” arXiv:1502.06434 [q-fin.ST], 2014

[5] D. Mandic and J. Chambers, Recurrent Neural Networks for Prediction, Wiley, 2001

[6] R. Williams and D. Zipser, “Gradient-based learning algorithms for recurrent networks and their computational complexity”, in Back-propagation: Theory, Architectures and Appli-Cations, Hillsdale, NJ: Erlbaum, 1992, pp. 433 - 486

[7] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies”, in A Field Guide to Dynamical Recurrent Neural

Networks, S. C. Kremer and J. F. Kolen, eds., IEEE press, 2001

[8] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory”, Neural Computation, vol. 9,no. 8, pp. 1735 - 1780, 1997

[9] K. Cho et al., “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”, arXiv:1406.1078 [cs.CL], 2014

[10] W. Gail, G. Yoav, and Y. Eran, "On the Practical Computational Power of Finite Precision RNNs for Language Recognition", arXiv:1805.04908 [cs.NE], 2018

[11] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". arXiv:1412.3555 [cs.NE]. 2014

[12] E. Real, et al., “Large-Scale Evolution of Image Classifiers,” arXiv:1703.01041 [cs.NE]. Jun 2017.

[13] D. Alajbeg, Z. Bubas and D. Vasic, “Price Distance To Moving Averages And Subsequent Returns”, International Journal of Economics, Commerce and Management, Vol. V, Dec 2017,pp. 33 - 47

[14] Progressive Web Apps, Google. Available:

https://developers.google.com/web/progressive-web-apps/

[15] Neoteric, “Single-page application vs. multiple-page application”, Medium. 2016. Available:

https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

[16] Keras: The Python Deep Learning library, Keras. Available: https://keras.io/.

[17] Documentation of scikit-learn 0.19.2. Available:

http://scikit-learn.org/stable/documentation.html.

[18] Alpha Vantage API Documentation, Alpha Vantage. Available:

https://www.alphavantage.co/documentation/.

[19] NumPy v1.14 Manual, The SciPy community. Available:

https://docs.scipy.org/doc/numpy-1.14.5/.

[20] pandas: powerful Python data analysis toolkit, Pandas. Available:

http://pandas.pydata.org/pandas-docs/version/0.23/.

[21] Hello, Colaboratory - Colaboratory - Google, Google. Available:

https://colab.research.google.com.

[22] TensorBoard: Visualizing Learning, Google. Available:

https://www.tensorflow.org/guide/summaries_and_tensorboard.

[23] Welcome to Flask — Flask 1.0.2 documentation, Pallets Team. Available:

http://flask.pocoo.org/docs/1.0/.

[24] Cloud Storage, Google. Available: https://firebase.google.com/docs/storage/.

[25] Getting Started, The Investors Exchange. Available:

https://iextrading.com/developer/docs/.

[26] Cloud Functions for Firebase, Google. Available:

https://firebase.google.com/docs/functions/.

[27] Cloud Firestore, Google. Available: https://firebase.google.com/docs/firestore/.

[28] React – A JavaScript library for building user interfaces, Facebook Inc; https://reactjs.org/.

[29] React Router: Declarative Routing for React.js, React Training. Available:

https://reacttraining.com/react-router/web/guides/philosophy.

[30] Read Me - Redux, Redux. Available: https://redux.js.org/.

[31] Immutable collections for JavaScript, Facebook Inc. Available:

https://github.com/facebook/immutable-js/.

[32] Introduction - Material Design, Google. Available: https://material.io/design/introduction/.

[33] Material UI, Material UI Team. Available: https://material-ui.com/

[34] Using Google Charts, Google. Available:

https://developers.google.com/chart/interactive/docs/.

[35] Facebook Login, Facebook. Available:

https://developers.facebook.com/docs/facebook-login/

[36] Firebase Authentication, Google. Available: https://firebase.google.com/docs/auth/

[37] unittest — Unit testing framework, Python Software Foundation. Available:

https://docs.python.org/3.6/library/unittest.html

[38] E. Real, A. Aggarwal, Y. Huang, Q. Le, “Regularized Evolution for Image Classifier Architecture Search,” arXiv:1802.01548 [cs.NE]. Feb 2018.

[39] GitHub features: the right tools for the job, Github Inc; https://github.com/features.