

# **PREDICTION OF STOCK MARKET DATA AND BUILDING A STRATEGY TO FIND RETURN ON INVESTMENT USING DEEP LEARNING METHODS**

**Project Report**

Done by

**JAGANNATH E**

Guided by

**Mr. SANJEEV KUMAR JHA**

Joint Director (Tech.)

In partial fulfillment of the requirement for

**POST GRADUATE DIPLOMA IN DATA SCIENCE AND ANALYTICS**



**NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION  
TECHNOLOGY [NIELIT] CHENNAI**

(An Autonomous Scientific Society of Ministry of Electronics & Information Technology)

**Government of India**

No. 25, Gandhi Mandapam Road, Chennai – 600025, Tamil Nadu, India.

# **DECLARATION**

I hereby declare that the project work entitled

## **PREDICTION OF STOCK MARKET DATA AND BUILDING A STRATEGY TO FIND RETURN ON INVESTMENT USING DEEP LEARNING METHODS**

Done at

**NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION  
TECHNOLOGY [NIELIT] CHENNAI**

(An Autonomous Scientific Society of Ministry of Electronics & Information Technology)

**Government of India**

No. 25, Gandhi Mandapam Road, Chennai – 600025, Tamil Nadu, India.

**Under the Guidance Of**

**Mr. SANJEEV KUMAR JHA**

**Joint Director (Tech.)**

**Name of the Student**

**Signature**

JAGANNATH E

**Place:** Chennai

**Date:**

# ACKNOWLEDGEMENT

First of all, I express my sincere gratitude to the almighty whose blessings helped me to materialize this project.

I would like to thank **Mr. Martin KM**, Director In-Charge, NIELIT Chennai for providing all the facilities required.

My sincere thanks to my project guide **Mr. Sanjeev Kumar Jha**, Joint Director (Tech.), NIELIT Chennai, for spending his precious time for me and giving his valuable ideas and suggestions, which helped me to complete the project successfully.

I express my deep gratitude to **Mr. Bharath C**, for providing me with necessary technical support and motivation.

Finally I would like to express my sincere thanks to all of my friends, colleagues, parents and all those who have directly assisted during this project.

## **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 Technical analysis and LSTM based Machine learning to predict stock values. Factors considered are open, close, low, high and volume

The ultimate goal of this project is to provide insights into the stock market world and build a successful strategy to smartly invest on selected stocks that yield better results or more profit than the others on basis of technical indicators and a model of Recurrent Neural Network – Long Short-Term Memory using TensorFlow deep learning framework. Since using deep neural networks is a data-driven approach, it is crucial to understand the dataset.

The evidence presented in this project supports that deep neural networks are powerful and credible method for building successful stock market strategy.

## **TABLE OF CONTENTS**

<b>S. No</b>	<b>List of contents</b>	<b>Page No</b>
1.	INTRODUCTION	1
2.	TOOLS AND PACKAGES USED	4
3.	DATABASE USED	9
4.	DATA EXPORTING	10
5.	ANALYSIS OF THE DATASET	13
6.	TECHNICAL ANALYSIS	18
7.	FORECASTING	25
8.	BUILDING AND TRAINING THE LSTM MODEL	27
9.	TRADING STRATEGY	29
10.	CONCLUSION	33
11.	CODE	34
12.	REFERENCES	48

# 1. INTRODUCTION

Predicting the Stock Market has been the goal of investors since its existence. Everyday billions of dollars are traded on the exchange, and behind each dollar is an investor hoping to profit in one way or another. Entire companies rise and fall daily based on the behaviour of the market. Should an investor be able to accurately predict market movements, it offers tantalizing promises of wealth and influence. It is no wonder then that the Stock Market and its associated challenges find their way into the public imagination every time it misbehaves. The 2008 financial crisis was no different, as evidenced by the flood of films and documentaries based on the crash. If there was a common theme among those productions, it was that few people knew how the market worked or reacted. Perhaps a better understanding of stock market prediction might help in the case of similar events in the future.

Despite its prevalence, Stock Market prediction remains a secretive and empirical art. Few people, if any, are willing to share what successful strategies they have. A chief goal of this project is to add to the academic understanding of stock market prediction. The hope is that with a greater understanding of how the market moves, investors will be better equipped to prevent another financial crisis. The project will evaluate some existing strategies from a rigorous scientific perspective and provide a quantitative evaluation of new strategies. It is important here to define the scope of the project. Although vital to any investor operating in the real world, no attempt is made in this project at portfolio management. Portfolio management is largely an extra step done after an investor has made a prediction on which direction any particular stock will move. The investor may choose to allocate funds across a range of stocks in such a way to minimize his or her risk. For instance, the investor may choose not to invest all of their funds into a single company lest that company takes unexpected turn. A more common approach would be for an investor to invest across a broad range of stocks based on some criteria he has decided on before. This project will focus exclusively on predicting the daily trend (price movement) of individual stocks. More so, the project will analyze the accuracies of these predictions.

Additionally, a distinction must be made between the trading algorithms studied in this project, these algorithms operate on the order of fractions of a second. The algorithms presented in this report will operate on the order of days and will attempt to be truly predictive of the market.

Many investors analyze stocks based on their fundamentals – such as their revenue, valuation or industry trends – but fundamental factors aren't always reflected in the market price. Technical analysis seeks to predict price movements by examining historical data, mainly price and volume. It helps traders and investors navigate the gap between intrinsic value and market price by leveraging techniques like statistical analysis and behavioral economics. Technical analysis helps guide traders to what is most likely to happen given past information. Most investors use both technical and fundamental analysis to make decisions.

Neural Networks is one of the most popular machine learning algorithms at present. It has been decisively proven over time that neural networks outperform other algorithms in accuracy and speed. With various variants like CNN (Convolution Neural Networks), RNN (Recurrent Neural Networks), Auto Encoders, Deep Learning etc. neural networks are slowly becoming for data scientists or machine learning practitioners what linear regression was one for statisticians. It is thus imperative to have a fundamental understanding of what a Neural Network is, how it is made up and what is its reach and limitations.

*Recurrent networks ... have an internal state that can represent context information. ... [they] keep information about past inputs for an amount of time that is not fixed a priori, but rather depends on its weights and on the input data.*

...

*A recurrent network whose inputs are not fixed but rather constitute an input sequence can be used to transform an input sequence into an output sequence while taking into account contextual information in a flexible way.*

— Yoshua Bengio, et al., [Learning Long-Term Dependencies with Gradient Descent is Difficult](#), 1994.

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behaviour required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field.

In this project, I hope to give insights into LSTMs using the words of research scientists that developed the methods and applied them to new and important problems. There are few that are better at clearly and precisely articulating both the promise of LSTMs and how they work than the experts that developed them. The LSTM model I'm going to create will help building a strategy for investors to invest smartly in the ever expanding world of stock market.



## 2. TOOLS AND PACKAGES USED

### 2.1 PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.



Some of the packages used in Python are as followed,

#### 2.1.1 NumPy

NumPy is the fundamental package for scientific computing with Python. It contains a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and FORTRAN code, useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

#### 2.1.2 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The

name is derived from the term "panel data", an econometric term for data sets that include observations over multiple time periods for the same individuals. some of the features of pandas are data frame object for data manipulation with integrated indexing, tools for reading and writing data between in-memory data structures and different file formats, data alignment and integrated handling of missing data, reshaping and pivoting of data sets, label-based slicing, fancy indexing, and subsetting of large data sets, data structure column insertion and deletion, group by engine allowing split-apply-combine operations on data sets, data set merging and joining, hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure. time series-functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging and provides provides data filtration.

### **2.1.3 Matplotlib**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

### **2.1.4 Seaborn**

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures. Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

### **2.1.5 Tqdm**

Tqdm mean "Progress" in Arabic it instantly make your loops show a smart progress meter. Tqdm works on any platform (Linux, Windows, Mac, FreeBSD, NetBSD, Solaris/SunOS), in any console or in a GUI, and is also friendly with IPython/Jupyter notebooks. Tqdm does not require any dependencies, just Python and an environment supporting carriage return \r and line feed \n control characters.

### **2.1.6 Scikit-learn**

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

### **2.1.7 TA -lib**

It is a Technical Analysis library to financial time series datasets (open, close, high, low, volume). You can use it to do feature engineering from financial datasets. It is built on Python Pandas library.

### **2.1.8 Keras**

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the Xception deep neural network model.

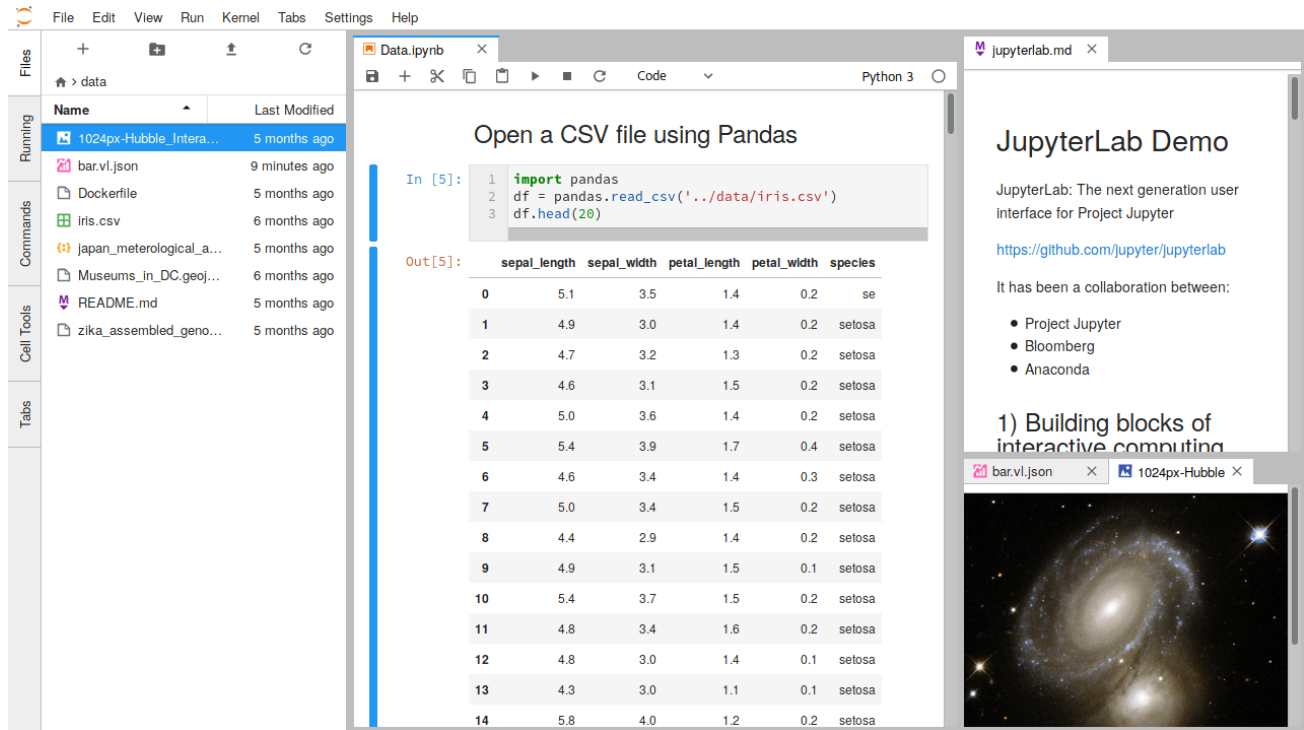
In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library.[5] Chollet explained that Keras was conceived to be an interface rather than a standalone machine-learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used. Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0.

### **2.1.9 TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

## 2.2 JUPYTER LAB

JupyterLab is a next-generation web-based user interface for Project Jupyter.



JupyterLab enables you to work with documents and activities such as Jupyter notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner. You can arrange multiple documents and activities side by side in the work area using tabs and splitters.

Documents and activities integrate with each other, enabling new workflows for interactive computing, for example:

- Code Consoles provide transient scratchpads for running code interactively, with full support for rich output. A code console can be linked to a notebook kernel as a computation log from the notebook, for example.
- Kernel-backed documents enable code in any text file (Markdown, Python, R, LaTeX, etc.) to be run interactively in any Jupyter kernel.
- Notebook cell outputs can be mirrored into their own tab, side by side with the notebook, enabling simple dashboards with interactive controls backed by a kernel.

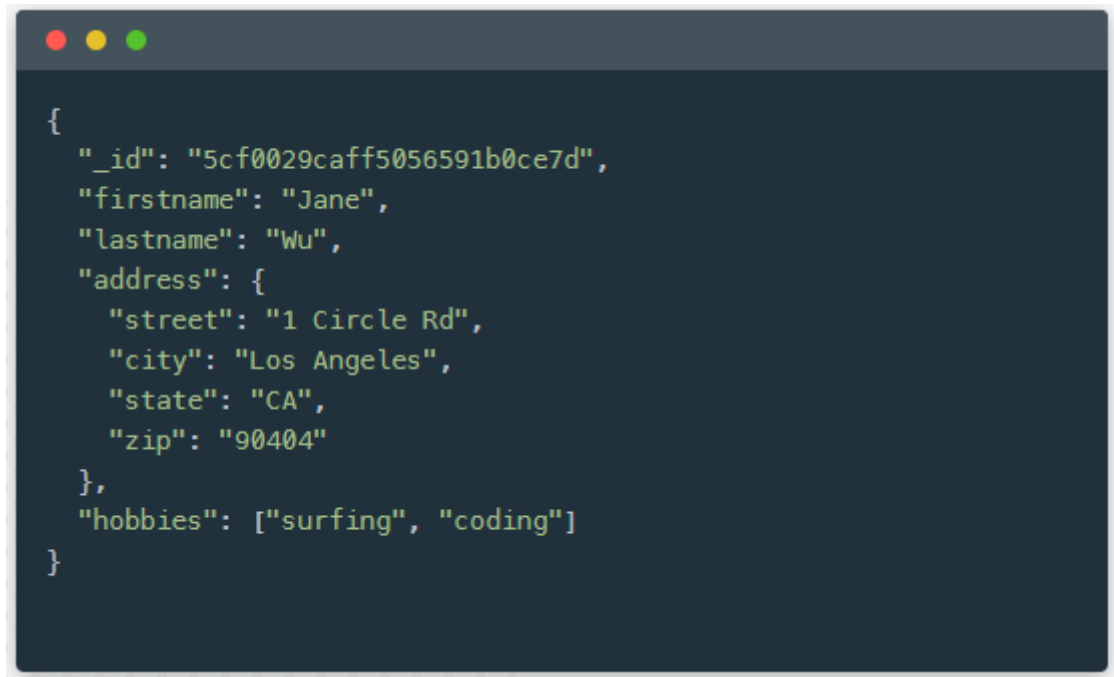
- Multiple views of documents with different editors or viewers enable live editing of documents reflected in other viewers. For example, it is easy to have live preview of Markdown, Delimiter-separated Values, or Vega/Vega-Lite documents.

JupyterLab also offers a unified model for viewing and handling data formats. JupyterLab understands many file formats (images, CSV, JSON, Markdown, PDF, Vega, Vega-Lite, etc.) and can also display rich kernel output in these formats.

## 3. DATABASE USED

### 3.1 MONGODB

MongoDB is a document database, which means it stores data in JSON-like documents. This is one of the most natural way to think about data, and is much more expressive and powerful than the traditional row/column model.



```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  },
  "hobbies": ["surfing", "coding"]
}
```

MongoDB's document data model makes it easy to build on, since it supports unstructured data natively and doesn't require costly and time-consuming migrations when application requirements change. MongoDB's documents are encoded in a JSON-like format, called BSON, which makes storage easy, is a natural fit for modern object-oriented programming methodologies, and is also lightweight, fast and traversable.

In addition, MongoDB supports rich queries and full indexes, distinguishing it from other document databases that make complex queries difficult or require a separate server layer to enable them. Its other features include automatic sharding, replication, and more.

I have used MongoDB to store my data and retrieve it whenever I wanted to and also used the MongoClient to connect with Python.

## 4. DATA EXPORTING

The data we needed to work our analysis with is found as raw day to day stock information from an open source Yahoo Finance. <https://in.finance.yahoo.com/most-active>

yahoo

finance

Search for news, symbols or companies

Q

Sign in

Mail

CryptocurrenciesCalendarWorld IndicesStocks: GainersStocks: LosersStocks: Most ActivesCurrenciesTop Mutual FundsCommoditiesMost Traded Options by Volume

<input type="checkbox"/> Symbol	Name	Price (intraday)	Change	% change	Volume	Avg vol (3-month)	Market Cap ▾	PE ratio (TTM)	52-week range
<input type="checkbox"/> RELIANCE.NS	Reliance Industries Limited	1,581.00	+43.10	+2.80%	13.466M	7.738M	10.022T	22.22	1,095.30 - 1,817.55
<input type="checkbox"/> RELIANCE.BO	Reliance Industries Limited	1,580.65	+42.95	+2.79%	617,437	408,103	10.021T	22.22	1,095.65 - 1,917.80
<input type="checkbox"/> TCS.NS	Tata Consultancy Services Limited	2,219.10	-19.70	-0.88%	3.279M	3.271M	8.327T	25.70	1,870.00 - 2,296.20
<input type="checkbox"/> TCS.BO	Tata Consultancy Services Limited	2,218.05	-20.35	-0.91%	71,372	124,838	8.324T	25.69	1,874.00 - 2,296.00
<input type="checkbox"/> HDFCBANK.NS	HDFC Bank Limited	1,278.15	-9.50	-0.74%	8.457M	5.18M	7T	28.65	1,011.00 - 1,305.50
<input type="checkbox"/> HDFCBANK.BO	HDFC Bank Limited	1,277.85	-9.75	-0.76%	185,105	384,230	6.999T	28.65	1,011.50 - 1,304.10
<input type="checkbox"/> HINDUNILVR.BO	Hindustan Unilever Limited	2,059.20	+11.35	+0.55%	47,301	60,806	4.458T	67.38	1,648.70 - 2,187.00
<input type="checkbox"/> HINDUNILVR.NS	Hindustan Unilever Limited	2,060.30	+11.05	+0.54%	994,231	1.266M	4.457T	67.42	1,650.00 - 2,190.00
<input type="checkbox"/> HDFC.BO	Housing Development Finance Corporation Limited	2,453.95	-28.25	-1.14%	25,981	118,265	4.243T	20.21	1,821.55 - 2,499.95
<input type="checkbox"/> HDFC.NS	Housing Development Finance Corporation Limited	2,453.95	-28.15	-1.13%	2.25M	2.965M	4.243T	20.21	1,820.00 - 2,499.90
<input type="checkbox"/> 715023.BO	IL&FS Financial Services Limited	8,998,760,448.00	+86.03	+0.00%	N/A	0	2,390,682.701T	2,672,634.368.00	990.00 - 9,480.00
<input type="checkbox"/> ICICIBANK.NS	ICICI Bank Limited	532.05	-5.10	-0.95%	18.308M	22.228M	3.442T	52.08	336.15 - 552.20
<input type="checkbox"/> ICICIBANK.BO	ICICI Bank Limited	530.90	-6.10	-1.14%	3.213M	926,820	3.436T	51.97	336.25 - 552.40
<input type="checkbox"/> INFY.NS	Infosys Limited	767.85	+0.85	+0.11%	3.761M	11.391M	3.256T	1,416.70	615.10 - 847.00
<input type="checkbox"/> INFY.BO	Infosys Limited	767.85	+1.00	+0.13%	88,333	706,624	3.256T	1,416.70	615.00 - 847.40
<input type="checkbox"/> KOTAKBANK.NS	Kotak Mahindra Bank Limited	1,698.10	-3.45	-0.20%	2.295M	2.196M	3.245T	39.45	1,210.00 - 1,734.80

Out of all the stock details available, I have considered the top 1980 stocks with most market cap relevant to the Indian stock market. I have taken the top 10 most market valid stocks for an example, they are:

	A	B
1	RELIANCE.NS	Reliance Industries Limited
2	RELIANCE.BO	Reliance Industries Limited
3	TCS.NS	Tata Consultancy Services Limited
4	TCS.BO	Tata Consultancy Services Limited
5	HDFCBANK.NS	HDFC Bank Limited
6	HDFCBANK.BO	HDFC Bank Limited
7	HDFC.BO	Housing Development Finance Corporation Limited
8	HDFC.NS	Housing Development Finance Corporation Limited
9	HINDUNILVR.NS	Hindustan Unilever Limited
10	HINDUNILVR.BO	Hindustan Unilever Limited

## 4.1 IMPORTING RAW DATA INTO LINUX

The next step in exporting this data includes scraping all the above mentioned details into my Linux System. As these details of numbers and values mentioned, however is not what I wanted to be working with, I will remove them using **shellscript** except for the first and most significant column for this entire project, tickers/symbols of these stock values.

File	Edit	View	Search	Terminal	Help										
RELIANCE.NS	Reliance Industries Limited	1,532.20	-3.10	-0.20%	3.801M	7.591M	9.721T	21.89							
RELIANCE.BO	Reliance Industries Limited	1,532.75	-2.60	-0.17%	65.184k	390,140	9.72T	21.89							
TCS.NS	Tata Consultancy Services Limited	2,193.20	+35.55	+1.65%	2.728M	3.298M	8.23T	25.40							
TCS.BO	Tata Consultancy Services Limited	2,193.75	+36.30	+1.68%	82,365	129,384	8.233T	25.40							
HDFCBANK.NS	HDFC Bank Limited	1,270.85	-15.90	-1.24%	1.670M	5.439M	6.96T	28.49							
HDFCBANK.BO	HDFC Bank Limited	1,271.00	-15.65	-1.22%	775,452	202,911	6.959T	28.49							
HDFC.BO	Housing Development Finance Corporation Limited	2,448.00	-18.20	-0.74%	9,543k	151,925	4.233T	20.16							
HDFC.NS	Housing Development Finance Corporation Limited	2,448.75	-17.65	-0.72%	638.545k		3.146M	4.233T	20.16						
HINDUNILVR.NS	Hindustan Unilever Limited	1,926.50	-11.55	-0.60%	610.112k		1.348M	4.17T	63.03						
HINDUNILVR.BO	Hindustan Unilever Limited	1,926.30	-11.85	-0.61%	6,837	68,019	4.17T	63.04							
ICICIBANK.BO	ICICI Bank Limited	538.80	-1.90	-0.35%	368,854	745,944	3.485T	52.74							
ICICIBANK.NS	ICICI Bank Limited	538.75	-1.85	-0.34%	2.820M	23.041M	3.485T	52.74							
KOTAKBANK.NS	Kotak Mahindra Bank Limited	1,660.05	-11.50	-0.69%	384.359k		2.224M	3.172T	38.57						
KOTAKBANK.BO	Kotak Mahindra Bank Limited	1,660.50	-10.95	-0.66%	6,376	89,903	3.172T	38.58							
INFY.NS	Infosys Limited	744.55	+9.85	+1.34%	3.203M	11.254M	3.158T	1,460.29							
INFY.BO	Infosys Limited	744.80	+10.10	+1.37%	110.481k	699,218	3.157T	1,460.00							
SBIN.BO	State Bank of India	335.80	-3.50	-1.03%	287.030k		1.618M	2.999T	24.43						
SBIN.NS	State Bank of India	335.85	-3.45	-1.02%	8.200M	36.322M	2.997T	24.42							
ITC.NS	ITC Limited	238.60	-1.25	-0.52%	2.571M	12.298M	2.933T	20.85							
ITC.BO	ITC Limited	238.70	-1.10	-0.46%	79,497	429,781	2.933T	20.86							
BAJFINANCE.NS	Bajaj Finance Limited	4,221.40	-24.65	-0.58%	315.140k		1.253M	2.54T	49.73						
BAJFINANCE.BO	Bajaj Finance Limited	4,221.70	-26.60	-0.63%	7,384	54,336	2.54T	49.73							
BHARTIARTL.NS	Bharti Airtel Limited	455.95	+0.75	+0.16%	1.648M	16.004M	2.339T	N/A							
BHARTIARTL.BO	Bharti Airtel Limited	456.35	+1.35	+0.30%	50,187	659,114	2.341T	N/A							
MARUTI.BO	Maruti Suzuki India Limited	7,308.00	-23.80	-0.32%	4,999	46,890	2.208T	36.06							
MARUTI.NS	Maruti Suzuki India Limited	7,307.00	-22.85	-0.31%	171.406k		849,031	2.208T	36.06						
STAN.BO	Standard Chartered PLC	63.00	+2.50	+4.13%	3	1,651	2.281T	3,461.54							
AXISBANK.BO	Axis Bank Limited	749.40	-7.85	-1.04%	40.589k	305,920	2.113T	41.77							
AXISBANK.NS	Axis Bank Limited	749.75	-7.20	-0.95%	1.837M	8.577M	2.114T	41.77							
LT.NS	Larsen & Toubro Limited	1,334.55	-10.75	-0.80%	724.512k		2.854M	1.873T	19.82						
LT.BO	Larsen & Toubro Limited	1,334.35	-10.65	-0.79%	23,440	153,281	1.873T	19.82							
ASIANPAINT.NS	Asian Paints Limited	1,766.80	-23.85	-1.33%	479,819	1.107M	1.695T	65.43							
ASIANPAINT.BO	Asian Paints Limited	1,767.45	-22.80	-1.27%	8,009	60,742	1.695T	65.45							

The processed data will look something like this..

```
File Edit View Search Terminal Help
RELIANCE.NS
RELIANCE.BO
TCS.NS
TCS.BO
HDFCBANK.NS
HDFCBANK.BO
HDFC.BO
HDFC.NS
HINDUNILVR.NS
HINDUNILVR.BO
ICICIBANK.BO
ICICIBANK.NS
KOTAKBANK.NS
KOTAKBANK.BO
INFY.NS
INFY.BO
```



## 4.2 SCRAPPING DATASET THROUGH THE SYMBOLS

I have saved the processed symbol information as a csv file and imported into python as a list.

```
[2]: import pandas as pd
import numpy as np

# Importing tickers/Symbols field from the scrapped stockdata

df = pd.read_csv('/home/nielit/Desktop/datasample.csv',header=None)
df.columns=["ticker"]
df.head()
```

[2]:

	ticker
0	RELIANCE.NS
1	RELIANCE.BO
2	TCS.NS
3	TCS.BO
4	HDFCBANK.NS

Then, I have imported pandas datareader library which will enable me to access the required data from the yahoo finance website.

```
[10]: # Collecting the stock market data from open source yahoo finance API for the following stock tickers for past 2 years

data = list()
for i in tqdm(tic):
    try:
        data.append(pdr.get_data_yahoo(symbols=i, start=datetime(2018, 1, 2), end=datetime(2019, 12, 31)))
    except:
        pass
```

100%|██████████| 1980/1980 [57:10<00:00, 1.24s/it]

Using these 1980 symbols and the yahoo finance API we have obtained the open, high, low, close, volume fields (OHLCV) which are basic properties that define a stock market.

We will perform our analysis using these fields. The entirety of this project is done by taking the past 24 months market cap of these symbols which will enable us to train our model as train, validation and test data. And also since, recent outcomes yield better result at least to an extent in this ever changing stock market research.

## 5. ANALYSIS OF THE DATASET

Each data frame rows are indexed by date. There are 1980 tickers with observations over 488 market days with 926833 entries, 2018-01-02 to 2020-01-01 are no null values within them.

```
[14]: df.head()
```

```
[14]:
```

	High	Low	Open	Close	Volume	Adj Close	Symbol
Date							
2018-01-02	919.549988	906.400024	913.000000	911.150024	4342815.0	900.578369	RELIANCE.NS
2018-01-03	926.000000	913.049988	925.000000	914.799988	6175312.0	904.185974	RELIANCE.NS
2018-01-04	921.799988	915.700012	918.150024	920.299988	4118581.0	909.622131	RELIANCE.NS
2018-01-05	926.900024	920.250000	921.799988	923.250000	3401905.0	912.537903	RELIANCE.NS
2018-01-08	931.000000	923.500000	926.099976	928.549988	4035417.0	917.776428	RELIANCE.NS

```
[7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 926833 entries, 2018-01-02 to 2020-01-01
Data columns (total 7 columns):
High      926833 non-null float64
Low       926833 non-null float64
Open      926833 non-null float64
Close     926833 non-null float64
Volume    926833 non-null float64
Adj Close 926833 non-null float64
Symbol    926833 non-null object
dtypes: float64(6), object(1)
memory usage: 56.6+ MB
```

```
[8]: print("There are {} Symbols with observations over {} days.".format(df.Symbol.unique().size, df.index.unique().size))
```

```
There are 1961 Symbols with observations over 488 days.
```

```
[9]: # Check for null values
```

```
df.isna().sum()
```

```
[9]: High      0
Low        0
Open       0
Close      0
Volume     0
Adj Close  0
Symbol     0
dtype: int64
```

Now, let's build dataframe for open, high, low, close, and volume as desired.

```
[10]: open_val = df[["Symbol", "Open"]].pivot(columns = "Symbol", values = "Open")
high_val = df[["Symbol", "High"]].pivot(columns = "Symbol", values = "High")
low_val = df[["Symbol", "Low"]].pivot(columns = "Symbol", values = "Low")
close_val = df[["Symbol", "Close"]].pivot(columns = "Symbol", values = "Close")
volume = df[["Symbol", "Volume"]].pivot(columns = "Symbol", values = "Volume")

[11]: # Open value dataframe
open_val.head()
```

Symbol	AARTIDRUGS.BO	AARTIDRUGS.NS	AARTIIND.BO	AARTIIND.NS	AARVEEDEN.BO	AARVEEDEN.NS	AAVAS.NS	ABAN.BO	ABAN.NS	ABANSENT.BO	...
Date											
2018-01-02	102.150002	73.900002	NaN	607.200012	40.500000	197.600006	1081.000000	132.100006	327.649994	69.400002	...
2018-01-03	100.000000	76.199997	NaN	603.349976	42.500000	194.949997	1117.199951	128.550003	331.799988	67.699997	...
2018-01-04	109.000000	77.949997	NaN	608.000000	41.000000	197.350006	1123.000000	128.550003	340.000000	68.699997	...
2018-01-05	105.449997	78.699997	NaN	610.150024	41.099998	194.750000	1118.000000	135.000000	334.799988	69.500000	...
2018-01-08	112.750000	78.300003	NaN	619.700012	42.750000	196.000000	1128.000000	131.699997	337.500000	68.500000	...

5 rows × 1961 columns

Next, three additional data frames were created for: return, next-day return, and high/low (these were requested in the task given to me). These metrics are potential features in addition to the OHLCV values. The dataframe for next day return is shown below, in the same format we will create 2 more dataframes.

We will create a dataframe containing the future close returns at time t, since we are predicting the close returns of next day value.

```
[14]: next_val = (close_val.shift(-1) / close_val - 1)
# The last day return looks something like this
next_val.tail()
```

Symbol	AARTIDRUGS.BO	AARTIDRUGS.NS	AARTIIND.BO	AARTIIND.NS	AARVEEDEN.BO	AARVEEDEN.NS	AAVAS.NS	ABAN.BO	ABAN.NS	ABANSENT.BO	...
Date											
2019-12-26	-0.001067	0.001949	0.005519	0.003541	0.0	-0.010000	-0.004409	0.009558	0.013861	0.044898	...
2019-12-27	-0.005342	0.023346	0.024904	0.000588	0.0	0.030303	0.000473	0.073373	0.018555	0.082031	...
2019-12-30	0.006982	-0.015209	0.002526	-0.001998	0.0	0.022409	-0.007569	0.016538	0.056088	0.055957	...
2019-12-31	0.001067	0.011583	0.001663	0.016724	0.0	0.026027	0.017263	0.000000	-0.021788	0.018803	...
2020-01-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...

5 rows × 1961 columns

## 5.1 EXPLORATORY DATA ANALYSIS

Before analyzing the coverage and quality, let us discuss the hypothesis, scope, motivation and strategy of this task.

### **Hypothesis:**

Based on research [1-3], the patterns in the technical indicators derived from OHLCV of a particular stock (or company) have the predictive power of its future movements.

### **Scope:**

This work limits its scope to hypothesis 1.

### **Motivation:**

This work extends hypothesis 1 to be more generic: unlike the hypotheses employed in [1-3], the future movement of any stock price can be predicted through the movement and pattern of its technical indicators; and the same prediction from movement and pattern can be applicable to stock price of any company.

Based on studies [1-4], the time series nature of the data recommends the use of long short-term memory (LSTM) paradigm of neural networks, which has been shown to be superior to ensemble methods such as random forest and other classification techniques such as the support vector machines.

### **Strategy:**

The strategy here is to build a generic prediction model using the OHLCV data and technical indicators of past 60 (trading) days so as to be able to predict the movement of the stock price of any company.

For this, an LSTM model will be trained using train, validation, and test data to obtain the following metrics: upward or downward movement of the next day stock price. In addition, the random forest classification method will be used for benchmarking and discussion.

## 5.2 COVERAGE AND QUALITY

```
[17]: # Histogram for non-missing values for 488 days

import matplotlib.pyplot as plt
hist = close_val.notna().sum().hist(bins=10)
plt.xlabel("Observations")
plt.ylabel("Symbols")
plt.title("Non-missing values in Symbols")
print("There are {} symbols with full data available.".format(close_val.columns[(close_val.notna().sum() == close_val.shape[0]).values].shape[0]))
plt.show()

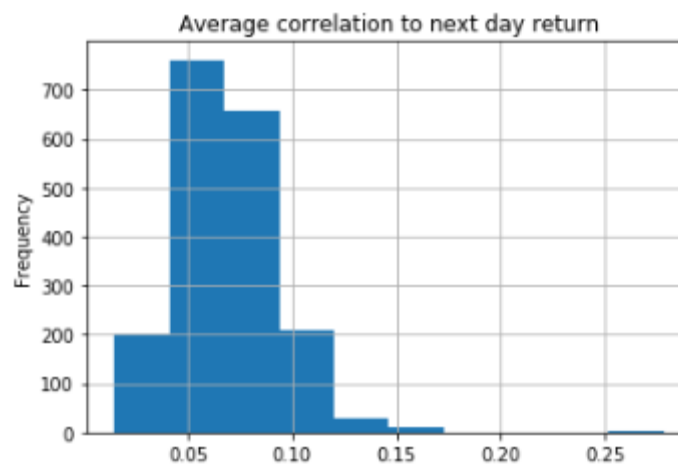
There are 1187 symbols with full data available.
<Figure size 640x480 with 1 Axes>

[18]: close_val.notna().sum()

[18]: Symbol
AARTIDRUGS.BO    488
AARTIDRUGS.NS    487
AARTIIND.BO     172
AARTIIND.NS     488
AARVEEDEN.BO    487
AARVEEDEN.NS    487
AAVAS.NS        488
ABAN.BO         487
ABAN.NS        488
ABANSENT.BO     488
ARR.BO          487
```

Analyzing the quality of the data we have found there are 1187 symbols out of 1980 which consists of full data. All missing values are coming from the price not being registered on the day. After crosschecking with data from the Yahoo Finance's official website, it is found that this is mainly due to the fact that the trade did not take place on that day. As clean result is of paramount importance, tickers with a lot of missing values are disregarded. In this case, only symbols with more than 400 non-missing values are considered.

Secondly, for good predictability, it is important to consider only companies with good correlation to the label, which in this case is the future return movement. Once, the list of tickers that are of interest to us have been shortlisted, we will move on to handling missing data. Also, we are also not interested in the compound indexes as they do not fall within the scope of the approach taken. Therefore, let us remove those tickers as well.



Based on the histogram above, we can see that there are very few tickers with good correlation to future values. For high profitability, we must take the tickers that possess good predictive power. Let us consider the tickers with average correlation value of more than the 90th percentile; and there are 188 of them.

```
[24]: sym90 = corr_val[corr_val.avgcorr > corr_val.avgcorr.quantile(0.90)].symbol.values

open_val = open_val[sym90]
high_val = high_val[sym90]
low_val = low_val[sym90]
close_val = close_val[sym90]
volume = volume[sym90]
next_val = next_val[sym90]
ret_val = ret_val[sym90]
hl_val = hl_val[sym90]

[25]: # Modify original dataframes

close_val = close_val.fillna(method="ffill") # close
close_val = close_val.fillna(method="bfill") # to handle the first row of close
volume = volume.applymap(lambda x: 0 if pd.isna(x) is True else x) # volume
open_val = open_val.fillna(close_val)
high_val = high_val.fillna(close_val)
low_val = low_val.fillna(close_val)

# calculate other dataframes
next_val = (close_val.shift(-1) / close_val) - 1 # future return
ret_val = (close_val / close_val.shift(1)) - 1 # return
hl_val = high_val / low_val # high/Low
```

Now, the missing values are handled. There are a couple of ways to handle the missing values on the day where the trade did not take place:

1. Set volume to 0 and open, high, low, and close values set to previous day's close value with the assumption that no trade occurred on that day for that particular ticker. This approach may introduce outliers in the volume and therefore noise in the data.
2. Interpolate the data OHLCV data using the previous and next available values for smooth transition. Another alternative is to fill in the missing values to be either the mean or median of the column. This, on the other hand, introduces false information.

Here, the first approach is taken as it is theoretically more correct.

## 6. TECHNICAL ANALYSIS

Before going into performing some visualizations and analysis, some technical indicators are examined. There are numerous such transformations in literature [1–3]. The work by Borovkova et al. [1] consolidates some of the key technical indicators which can be used for additional transformations. Typically, they can be categorized into four groups:

We want to create features that possess some level of predictive power which could indicate the future direction of the market. Statistically, these features should have good correlation power to the market movement.

There are many such transformations in Statistics.

According to work by Borovkova, some of the key technical indicators that can be used are categorised into four groups, Momentum, Trend, Volume and Volatility.

Some of the commonly used indicators are:

### **Momentum:-**

1. Money flow index
2. Relative strength index
3. Stochastic oscillator
4. William %R

### **Trend:-**

1. Exponential moving average
2. Moving average convergence-divergence
3. Commodity channel index
4. Ichimoku Indicator

## Volume:-

### 1. Accumulation/distribution index

## Volatility:-

### 1. Bollinger bands

All transformations were done using an open-source technical analysis ta - Python package [5].

The formula to calculate these technical indicators are:

#### Moving average convergence-divergence (MACD)

The moving average convergence-divergence is an indicator that shows the relationship between 12-period EMA and 26-period EMA. This index can be used to measure the trend-following momentum of a security. Accordingly, it is calculated as:

$$\text{MACD} = 12\text{-Period EMA} - 26\text{-Period EMA}$$

where EMA is the exponential moving average, as defined and programmed earlier.

#### William %R

William's %R is another momentum indicator that ranges between -100 and 0 which shows the overbought and oversold of a stock price. For example, a reading above -20 is overbought and a reading below -80 is oversold. Typically, it is calculated over the period of last  $N = 14$  days.

This indicator can be simply calculated as follows:

$$\%R = \frac{\text{Highest High} - \text{Close}}{\text{Highest High} - \text{Lowest Low}}$$

#### Accumulation/distribution index

Accumulation/distribution is a cumulative indicator that makes use of volume and price to determine if the stock is being accumulated or distributed. Typically, a rising A/D line helps confirm a rising price trend whereas a falling A/D line helps confirm a price downtrend.

$$A/D = \text{Previous A/D} + (\text{Close} - \text{Low}) - (\text{High} - \text{Close}) \frac{1}{\text{High} - \text{Low}} - \text{Low} * \text{Volume}$$

#### Bollinger bands

Bollinger bands are two lines that are two standard deviations away from the EMA. It is a volatility indicator. Here, it will be presented in terms of percentage as:

$$BB = \frac{\text{Close} - \text{Lower Band}}{\text{Higher Band} - \text{Lower Band}}$$

Typically, value of above 0.8 gives sell signal; and buy signal for value below 0.2.



### Money flow index (MFI)

Money flow index is calculated based on the price and volume to indicate the strength of money in and out from a particular ticker, or in other words, if a particular stock is overbought or oversold. Typically, when MFI of above 80 indicates overbought and oversold when it is below 20.

The MFI can be calculated as follows:

$$MFI = 100 - \frac{100}{1 + MFR}$$

where

$$MFR = \frac{\text{Positive Money Flow}}{\text{Negative Money Flow}}$$
$$\text{Money Flow} = \left( \frac{\text{High} + \text{Low} + \text{Close}}{3} \right) \text{Volume}$$

and MFR denotes money flow ratio.

### Exponential moving average (EMA)

The exponential moving average is exponentially weighed moving average calculated by exponentially decreasing the weight of observations  $x_t$  with respect to their distance from  $x_t$  using weighted multiplier  $\alpha$ . For example,  $\alpha = 0.1$  gets only 10% of the current value into EMA. Since only a small portion of the current value is taken, most of the old values are preserved. Essentially, EMA is defined as:

$$EMA(x_t, \alpha) = \alpha x_t + (1 - \alpha) EMA(x_{t-1}, \alpha)$$

and

$$\alpha = \frac{2}{N + 1}$$

where  $N$  is number of days in a period. In this study, let us use  $N = 10$  to be consistent with MFI, and hence  $\alpha = 0.13333$  using the close price of day  $t$  as  $x_t$ .

### Relative strength index (RSI)

Just like the MFI, the relative strength index is another momentum indicator, but uses the velocity and magnitude of price movements and also to indicate whether a stock is overbought or oversold. The RSI can be simply calculated as follows:

$$RSI = 100 - \frac{100}{1 + RS}$$

where

$$RS = \frac{\text{Average of up closes}}{\text{Average of down closes}}$$

is the relative strength. For consistency, let us use  $N = 10$  when calculating the average.

### Stochastic oscillator (SO)

The stochastic oscillator (or Stochastic %K) is a momentum indicator calculated using the highest high price over a period of time  $N$ , lowest low price over a period of time  $N$ , and the current close price. The calculation can be simply performed as:

$$\%K = \left( \frac{\text{Close} - \text{Lowest Low}}{\text{Highest High} - \text{Lowest Low}} \right) * 100$$

which gives the result in percentage.

The fast stochastic indicator (Stochastic %D) is normally the 3-period moving average of %K. Both indicators will be calculated below.

### Commodity channel index (CCI)

The commodity channel index is another trend indicator, but the difference in this indicator is that it measures the difference between the current price and the historical average price. This indicator can be calculated as follows:

$$CCI = \frac{1}{0.015} \frac{\text{Typical Price} - \text{SMA}(\text{Typical Price})}{\text{MD}(\text{Typical Price})}$$

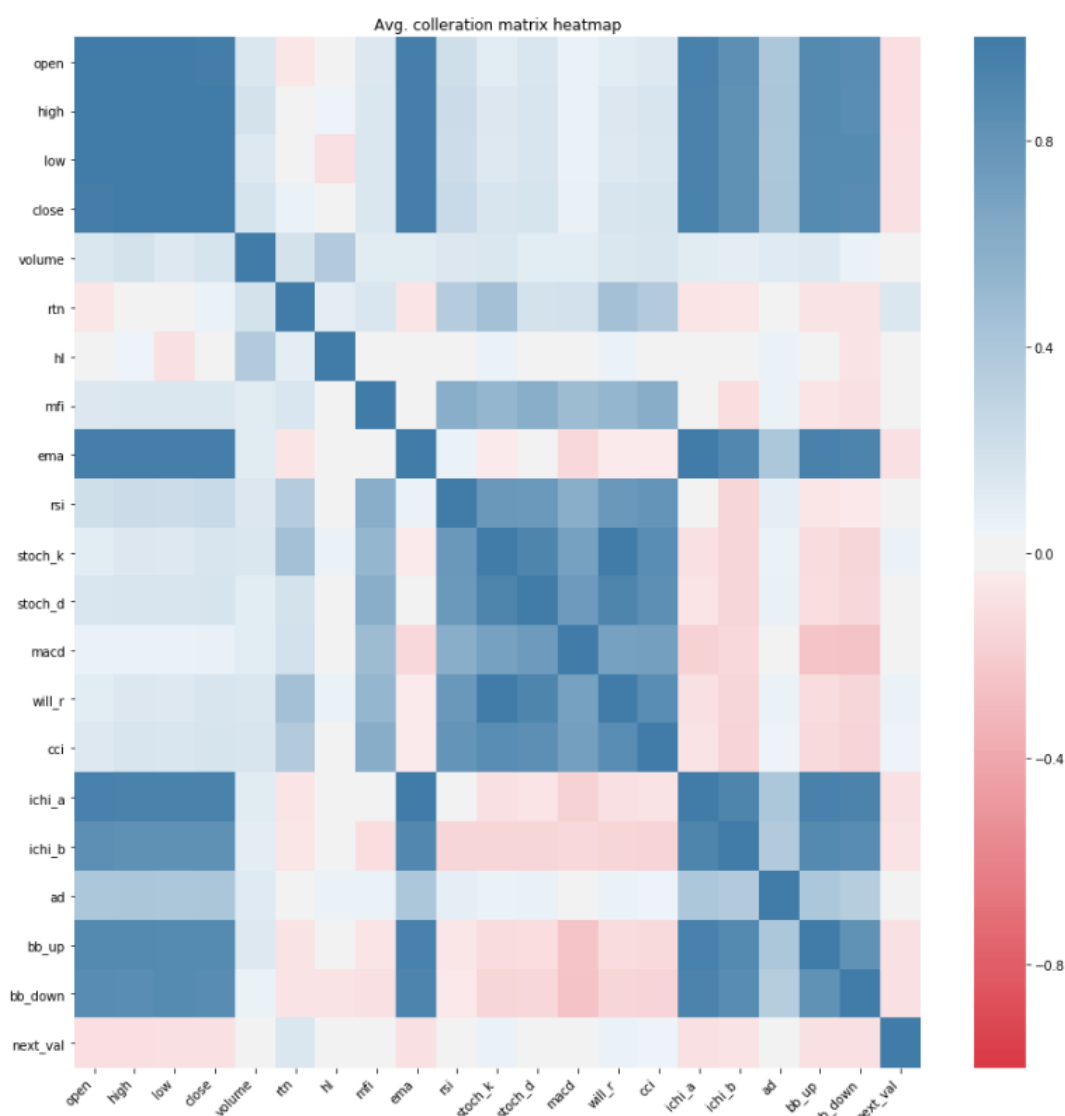
where SMA is the simple moving average and MD is the mead absolute deviation.

When the CCI is above zero it indicates the price is above the historic average. When CCI is below zero, the price is below the historic average. Readings above 100 typically signals buy signal; and sell signal for reading below -100.

Before performing some data analysis, the data is be split to training, validation, and testing sets as follows:

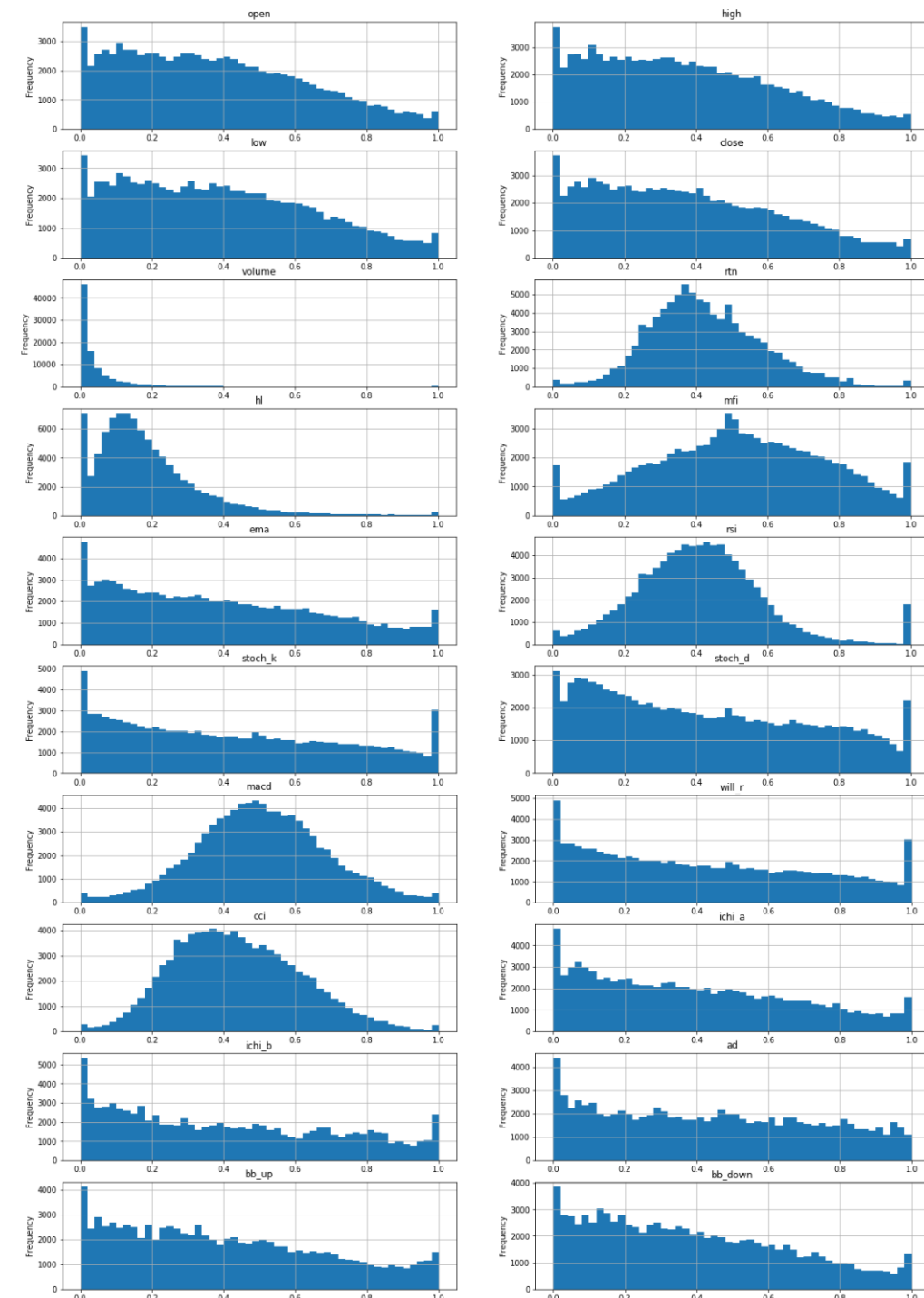
1. Train: 2018-01-02 to 2019-03-01
2. Validation: 2019-03-02 to 2019-07-31
3. Test: 2019-08-01 to 2020-01-01

First, the correlation between all the data frames, or in other words, between features are visualized. Because there are numerous tickers in a data frame, it could be difficult to visualize the correlation matrix for all tickers. Therefore, here, the average correlation value is taken and the correlations are visualized using a heat map.



In addition to the correlation matrix heat map, let's have a look at the histogram of data to look for outliers.

benchm



Analysing the correlation heat map and histograms above.

Open, high, low and close values are highly correlated to each other. This is expected because the values are within close proximity to each other. As the technical indicators are calculated using these values and intrinsically retain its information, open, high, and low values can be removed from feature list.

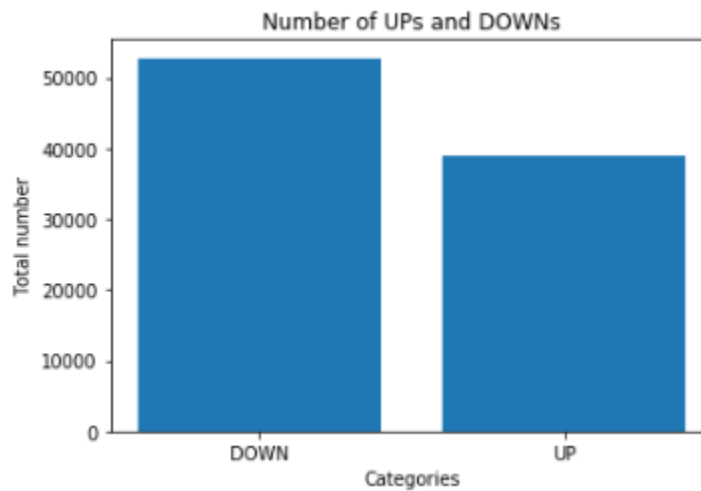
Most of the distributions above are either normal or uniform in shape, except for hl and volume. This suggests that there will be a large number of outliers in those two features. Although the other features with normal distribution may have outliers in the tail section, they can be considered negligible. In addition, observing the correlation matrix, they also appear to have low correlation to future return, which is the metric we will be predicting. Therefore, it justifies leaving hl and volume out from the feature list. Similarly, based on the correlation matrix heatmap, some values appear to not have much correlation to next day value. They are: mfi and ad. To keep the prediction model simple, these features can also be regarded as not useful.

Finally, it also appears that stoch\_k is highly correlated to william\_r. This is expected because the mathematical expression of both the indicators is similar. Here, william\_r will be removed from the feature list.

Based on the discussion above, the final feature list would be: close, rtn, ema, rsi, stoch\_k, stoch\_d, macd, cci, ichi\_a, ichi\_b, bb\_up, bb\_down.

Our strategy is to predict the UP and DOWN movement of a stock together. Based on experience, this is a better strategy compared to predicting the direction of future return as the performance metric based on this can be misleading. For example, if we build a model that minimises the mean-squared-error, that still doesn't mean that the direction of the movement can be correct. If the actual movement is 0.1%, then investing based on 0.5% is better compared to -0.1% prediction although the later might have a error value.

Now, we first need to create a target data frame that categorises into 1 and 0: 1 for UP and 0 for DOWN. Then, let us look at the distribution of the values.



Based on the analysis above, it appears that there are more DOWNS than UPs. Therefore, some balancing is required before sending the data for machine learning. Before going into forecasting section, the final list of data frames should be defined based on the features list finalized earlier.

## 7. FORECASTING

### 7.1 PRE-PROCESSING THE DATA

Based on the EDA done in the previous section, this section builds an LSTM neural network model for the prediction of the next day stock price moment; whether it is going upward or downward.

First, let's split the data to training, validation, and test sets. The training and validation sets will be used during the LSTM network training, while the test set will be used for trading strategy implementation and additional testing of the final model.

```
[45]: # Train set
dfois_train = []
for df in dfois:
    dfois_train.append(df.iloc[df.index < '2019-03-01'])

# Test set
dfois_test = []
for df in dfois:
    dfois_test.append(df.iloc[df.index >= '2019-08-01'])

# Validation set
dfois_eval = []
for df in dfois:
    dfois_eval.append(df.iloc[(df.index >= '2019-03-01') & (df.index < '2019-08-01')])
```

#### Normalizing the data:

Next, let us preprocess the data. Let us normalize all the data to be between 0 and 1. Note that the maximum and minimum is with respect to each ticker in each train data frame

```
[46]: # List of normalisers corresponding to each dataframe
normalisers = []

for i, df in enumerate(dfois[:-1]):
    # create the normaliser object
    normalisers.append(preprocessing.MinMaxScaler(feature_range=(0,1)))

    # columns and indexes
    columns = dfois_train[i].columns
    index_train = dfois_train[i].index
    index_test = dfois_test[i].index
    index_eval = dfois_eval[i].index

    # fit normalise
    normalisers[i].fit(dfois_train[i])

    # transform
    train_data = normalisers[i].transform(dfois_train[i])
    eval_data = normalisers[i].transform(dfois_eval[i])
    test_data = normalisers[i].transform(dfois_test[i])

    # replace list
    dfois_train[i] = pd.DataFrame(train_data, columns=columns, index=index_train)
    dfois_eval[i] = pd.DataFrame(eval_data, columns=columns, index=index_eval)
    dfois_test[i] = pd.DataFrame(test_data, columns=columns, index=index_test)
```

## Scaling the data:

Now, let us scale the data to have mean 0 and unit variance. This scaling is done by training the scaler using data of entire normalized training data frame. This will be used to transform the training, validation, and test sets.

```
[47]: # List of scalers corresponding to each dataframe
scalers = []

# Standardise the data by fitting the train set
for i, _ in enumerate(dfois[:-1]):
    # create the Scaler object
    scalers.append(preprocessing.StandardScaler())

    # columns and indexes
    columns = dfois_train[i].columns
    index_train = dfois_train[i].index
    index_test = dfois_test[i].index
    index_eval = dfois_eval[i].index

    # fit scale
    flat_arr = dfois_train[i].values.reshape(dfois_train[i].shape[0]*dfois_train[i].shape[1],1)
    scalers[i].fit(np.tile(flat_arr, dfois_train[i].shape[1]))

    # transform
    train_data = scalers[i].transform(dfois_train[i])
    eval_data = scalers[i].transform(dfois_eval[i])
    test_data = scalers[i].transform(dfois_test[i])

    # replace list
    dfois_train[i] = pd.DataFrame(train_data, columns=columns, index=index_train)
    dfois_eval[i] = pd.DataFrame(eval_data, columns=columns, index=index_eval)
    dfois_test[i] = pd.DataFrame(test_data, columns=columns, index=index_test)
```

## 60 Time Steps

Let's sequence the data. We are going to use the past 60 days of data for the next day prediction, we need to append the past 60 days worth data and append them in an array. We first write a function and execute onto all three sets of data and then we split them into three to find the train, validation and test dataset.

```
[49]: sequential_data_train = sequence_data(dfois_train)
sequential_data_eval = sequence_data(dfois_eval)
sequential_data_test = sequence_data(dfois_test, shuffle=False) # do not shuffle just this one

# Print the Length
print('Training data length: {}'.format(len(sequential_data_train)))
print('Validation data length: {}'.format(len(sequential_data_eval)))
print('Testing data length: {}'.format(len(sequential_data_test)))
```

/home/nielit/anaconda3/lib/python3.7/site-packages/ipykernel\_launcher.py:17: FutureWarning: Sorting because n of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

```
Training data length: 42864
Validation data length: 7896
Testing data length: 7708
```

## 8. BUILDING AND TRAINING THE LSTM MODEL

Once sequencing is complete, the data was balanced. This was to make sure the LSTM model is not biased towards the downward trends, as observed earlier during data analysis.

A LSTM model using batch size of 512 and 3 epochs were compiled (using Keras in Tensorflow) due to computational power the neural network is handled more gently than such that would be sufficed, in the future the batch size and the respective epochs could be handled more efficiently. The summary of the model is as follows:

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 60, 128)	72192
dropout_4 (Dropout)	(None, 60, 128)	0
batch_normalization_3 (Batch Normalization)	(None, 60, 128)	512
lstm_4 (LSTM)	(None, 60, 128)	131584
dropout_5 (Dropout)	(None, 60, 128)	0
batch_normalization_4 (Batch Normalization)	(None, 60, 128)	512
lstm_5 (LSTM)	(None, 128)	131584
dropout_6 (Dropout)	(None, 128)	0
batch_normalization_5 (Batch Normalization)	(None, 128)	512
dense_2 (Dense)	(None, 32)	4128
dropout_7 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 2)	66

```
Total params: 341,090  
Trainable params: 340,322  
Non-trainable params: 768
```

None



```
[57]: history = model.fit(train_x, train_y,
                        batch_size=BATCH_SIZE,
                        epochs=EPOCHS,
                        validation_data=(validation_x, validation_y))
```

Train on 36910 samples, validate on 5848 samples  
 WARNING:tensorflow:From /home/nielit/anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/math\_grad.py:1250: add\_dispatch\_support.<1  
 sorflow.python.ops.array\_ops) is deprecated and will be removed in a future version.  
 Instructions for updating:  
 Use tf.where in 2.0, which has the same broadcast rule as np.where  
 Epoch 1/3  
 36910/36910 [=====] - 232s 6ms/sample - loss: nan - acc: 0.5000 - val\_loss: nan - val\_acc: 0.5000  
 Epoch 2/3  
 36910/36910 [=====] - 178s 5ms/sample - loss: nan - acc: 0.5000 - val\_loss: nan - val\_acc: 0.5000  
 Epoch 3/3  
 36910/36910 [=====] - 176s 5ms/sample - loss: nan - acc: 0.5000 - val\_loss: nan - val\_acc: 0.5000

Essentially, the LSTM models will have 3 LSTM layers and dense layer with activation and then finally a dense output layer with softmax activation function as this is a classification problem. The softmax activation function will output the probability of the classification.

To avoid over-fitting, dropout layers with are added. As additional measure, regularization could also be added if deemed necessary so that the weights optimized during training do not get too large. However, in this case, they are not necessary.

An adaptive first-order gradient-based optimization known as Adam is used for optimizing the model. Another famous optimizer would be the stochastic gradient descent known as SGD. As this is a classification problem, accuracy calculated using sparse categorical cross-entropy and loss functions are used as the performance metric. All metrics specified above are hyper-parameters for this model. A hyper-parameter search using a grid of batch size, epochs, unit values and optimization methods can be performed, but due to the hardware limitations, only these initial hyper-parameters are used

```
Test loss: nan
Test accuracy: 0.56227297

Classification report:
              precision    recall  f1-score   support

     0.0         0.56      1.00      0.72       4334
     1.0         0.00      0.00      0.00       3374

 accuracy          0.28      0.50      0.36       7708
 macro avg         0.28      0.50      0.36       7708
 weighted avg         0.32      0.56      0.40       7708
```

## 9. TRADING STRATEGY

Let us define a simple trading strategy. Of all 188 tickers used, open position using \$1 for each ticker based on the predicted movement by the LSTM network, and close position the next day. This means that every day, an investment of \$188 is made and position must be closed the next day.

Note that, as the LSTM network requires 60 days of past data, the trading is simulated from 2019-08-01. The cumulative profit and loss (PnL) will be plotted.

The accuracy score based on the test set shows that the results are in line with research whereby the accuracy on test set is lower than that of the training and validation set. The prediction of upward movement has low precision compared to that of the downward trend, which means that there are more false positives when predicting the upward movement. Often, the recall score is inversely proportional to the precision. Therefore, the best measure is the f1-score, which is acceptable for developing a trading strategy.

```
[131]: # build index list
pred_index = dfois_test[0].index[dfois_test[0].index >= dfois_test[0].head(60).index[-1]]

# build prediction dataframe - 1 is BUY and 0 is SELL
df_pred = pd.DataFrame(pred.reshape(dfois_test[0].shape[0]-SEQ_LEN+1,dfois_test[0].shape[1], order='F'),
                        index = pred_index,
                        columns = dfois_test[0].columns)

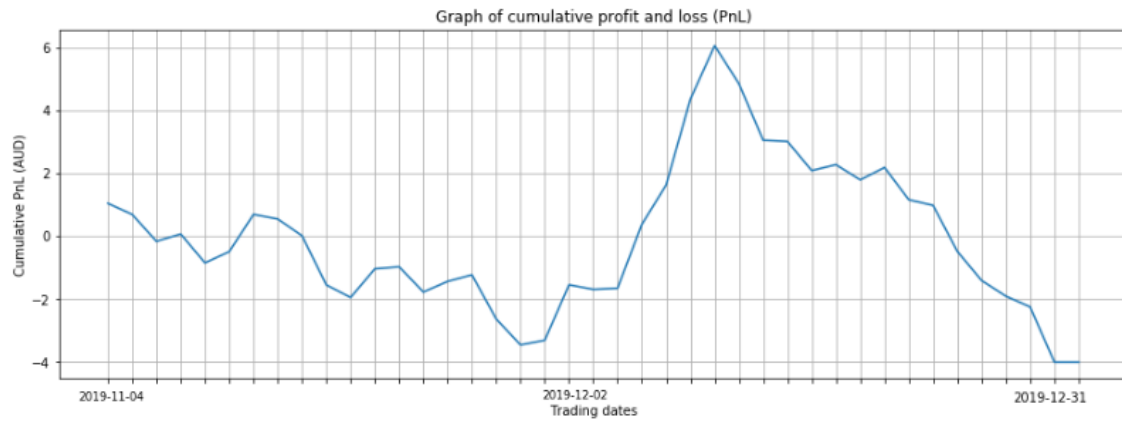
[191]: # dataframe for right prediction - 1 is CORRECT and 0 is WRONG
df_right = (df_pred.astype(bool) == df_target.loc[pred_index].astype(bool)).astype(int)

# dataframe for wrong prediction - 1 is WRONG and 0 is CORRECT
df_wrong = (~df_right.astype(bool)).astype(int)

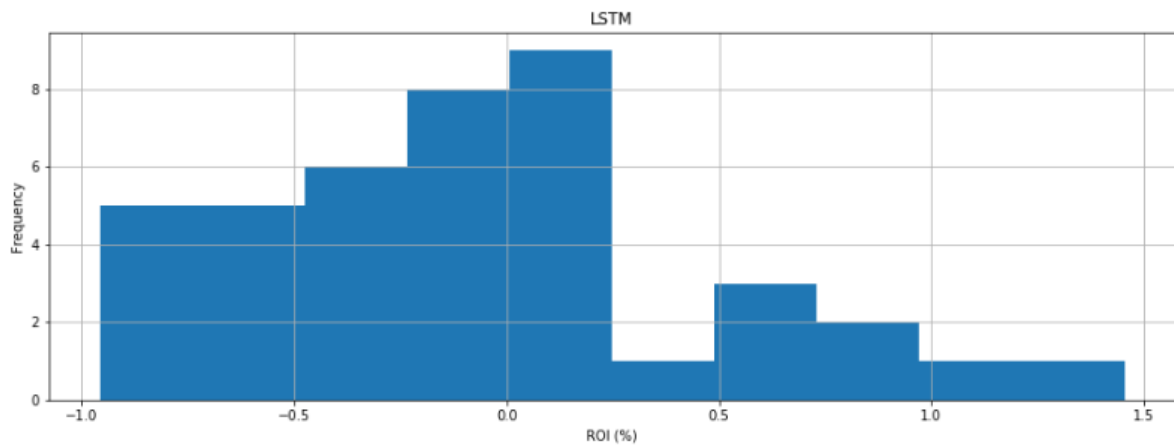
# dataframe for profit
df_profit = df_right*next_val.loc[pred_index].abs()

# dataframe for loss
df_loss = df_wrong*next_val.loc[pred_index].abs()

plt.subplots(figsize=(15,5))
plt.plot((df_profit.sum(axis=1) - df_loss.sum(axis=1)).cumsum())
plt.grid()
plt.xlabel('Trading dates')
plt.ylabel('Cumulative PnL (AUD)')
plt.title('Graph of cumulative profit and loss (PnL)')
plt.show()
```



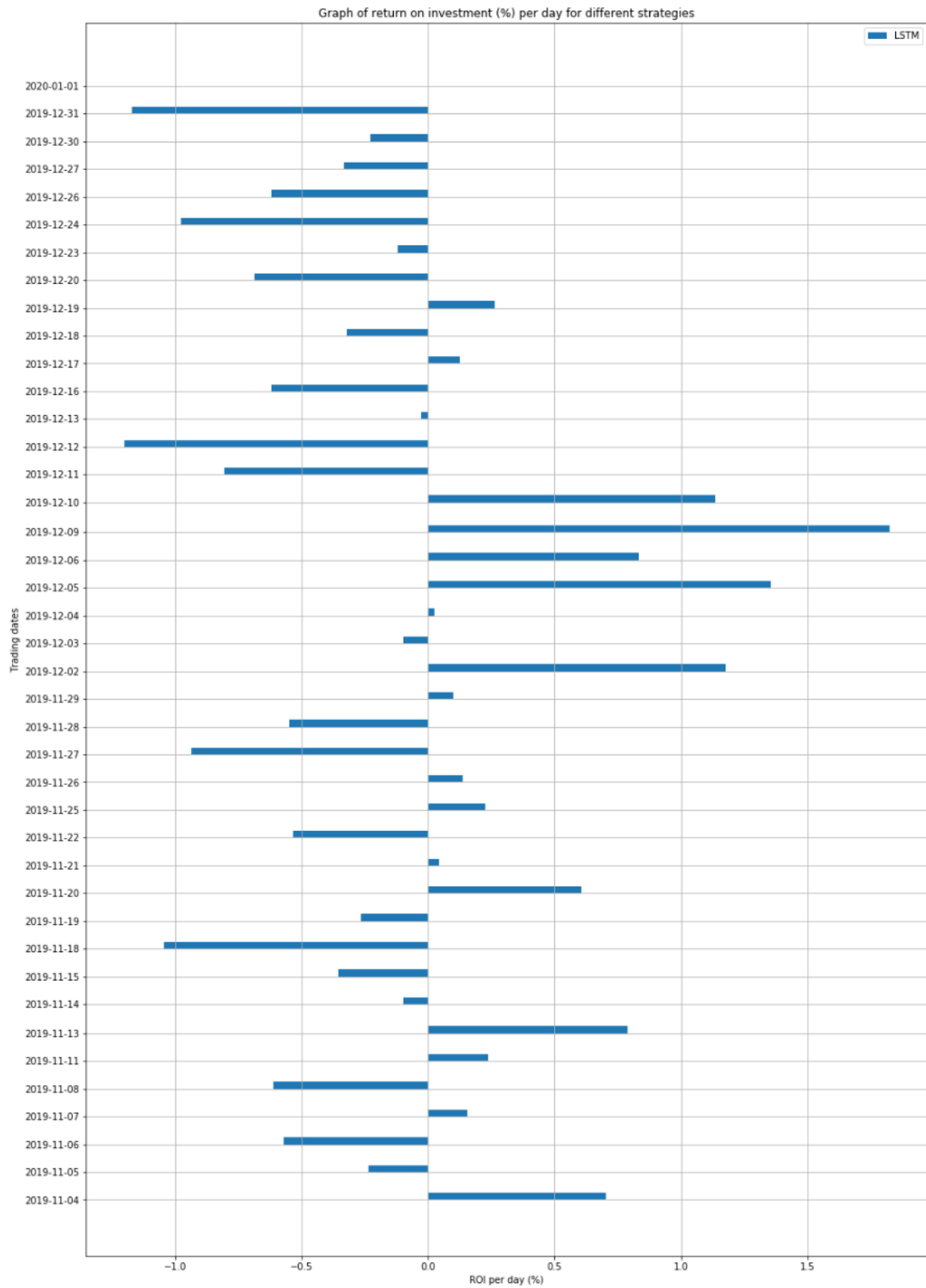
Following the simple strategy, based on 44 days of trading and \$188 investment per day, the total highest per day profit made is about \$6. On average, this is a 0.25% daily return on investment (ROI).



```
[206]: from datetime import timedelta

# pnl of lstm
pnl_lstm = (df_profit.sum(axis=1) - df_loss.sum(axis=1))

# plotting the roi per day
plt.subplots(figsize=(16, 24))
plt.barh((pnl_lstm/150*100).index,(pnl_lstm/150*100).values, label = "LSTM", align='edge', height=0.25)
plt.xlabel('ROI per day (%)')
plt.ylabel('Trading dates')
plt.title('Graph of return on investment (%) per day for different strategies')
plt.legend()
plt.grid()
plt.show()
```



```
[214]: from scipy import stats

# calculating the confidence
conf_lstm = stats.percentileofscore((pnl_lstm/188*100).values, 2) - stats.percentileofscore((pnl_lstm/188*100).values, 0)

print('Confidence of employed approaches in making positive ROI based on test set:')
print('LSTM: {}'.format(conf_lstm))

Confidence of employed approaches in making positive ROI based on test set:
LSTM: 41.46341463414634%
```

Confidence of employed approach in making positive ROI based on test set:

**LSTM:** 41.46341463414634%

This confidence level can be increased by tweaking the neural network to yield more and better strategy on predicting the Return on Investment. With more computational power and more training the neural network I'm sure this strategy can come up with better numbers using this Long Short-Term Memory model.

## 10. CONCLUSION

A historical data for top companies in the Yahoo Finance was given for the task of predicting the next day movement of the securities and subsequently coming up with a trading strategy by using just 488 days of end of day information of 1980 stocks.

The data was first extracted, analysed, cleaned and features were engineered. Following that, a *long short-term memory* (LSTM) network model was built for securities with good predictability.

The results shows that the LSTM model provides a reasonable accuracy for a successful trading strategy for the stock price of 188 Indian companies that has good predictability. Furthermore, the result shows that with *41.46341463414634%* confidence, the LSTM model provides a positive daily return on investment based on the testing data.

## 11. CODE

*# Data Acquisition*

*Install the following necessary packages*

```
#pip install pandas
#pip install numpy
#pip install scipy
#pip install yahoo_finance_api2
#pip install pandas-datareader
#pip install ta
#pip install seaborn
#pip install sklearn
```

```
import pandas as pd
import numpy as np
```

*# Importing tickers/Symbols field from the scrapped stockdata*

```
df = pd.read_csv('/home/nielit/Desktop/datasample.csv',header=None)
df.columns=["ticker"]
df.head()
```

```
len(df.ticker)
```

```
len(df.ticker.unique())
```

*# Check for duplicate values*

```
df_dup = df[df.duplicated()]
df_dup
```

*# Drop the duplicate values*

```
df = df.drop(df.index[300])
```

```
tic = list(df.ticker)
```

*# Import necessary packages*

```
import pandas_datareader as pdr
from datetime import datetime
```

```
from tqdm import tqdm
```

*# Collecting the stock market data from open source yahoo finance API for the following stock tickers for past 2 years*

```
data = list()
for i in tqdm(tic):
    try:
        data.append(pdr.get_data_yahoo(symbols=i, start=datetime(2018, 1, 2), end=datetime(2019,
```

```

12, 31)))
    except:
        pass

for i in range(len(data)):

    data[i]["Symbol"] = tic[i]

df = pd.DataFrame(data[0])

for i in np.arange(1,len(data)):
    df = df.append(data[i])

df.isnull().sum()

```

*Our required dataframe df looks something like this..*

```

df.head()

# Saving the dataframe in local

df.to_csv("/home/nielit/Desktop/StockData2.csv")

# Loading the dataframe back from local

df_stock = pd.read_csv("/home/nielit/Desktop/StockData2.csv")

# The stock data is in the exact format to store in a database

df_stock.info()

# Using MongoDB as my database

from pymongo import MongoClient
from random import randint
try:
    client_mongo = MongoClient()
    print("Connected successfully!!!")
except:
    print("Could not connect to MongoDB")

# Creating a database db and collection stock

db = client_mongo.db_6
collection = db.stock

#### Importing the dataset into database using MongoClient

for i in df_stock.values:
    collection.insert_one({"Date":i[0], "High":i[1], "Low":i[2], "Open":i[3], "Close":i[4], "Volume":i[5], "Adj
    Close":i[6], "Symbol":i[7]})

```



*The stock data is now stored in MongoDB*

*### Exporting the dataset from MongoClient back here*

```
df = pd.DataFrame(list(db.stock.find()))
df.head()
```

*# Reformatting the data back into our desired form from MongoDB*

```
df = df.drop(['_id'],axis=1)
df = df[['Date','High','Low','Open','Close','Volume','Adj Close','Symbol']]
df = df.set_index('Date')
df.head()
```

```
import pandas as pd
import numpy as np
df = pd.read_csv(r"C:\Users\JAGANNATH\Desktop\Project\Project\StockData2.csv")
df = df.sort_values(by='Date')
df['Date'] = pd.to_datetime(df['Date'])
df = df.set_index(['Date'])
df.head()
```

```
df.info()
```

```
print("There are {} Symbols with observations over {} days.".format(df.Symbol.unique().size,
df.index.unique().size))
```

*# Check for null values*

```
df.isna().sum()
```

```
open_val = df[["Symbol", "Open"]].pivot(columns = "Symbol", values = "Open")
high_val = df[["Symbol", "High"]].pivot(columns = "Symbol", values = "High")
low_val = df[["Symbol", "Low"]].pivot(columns = "Symbol", values = "Low")
close_val = df[["Symbol", "Close"]].pivot(columns = "Symbol", values = "Close")
volume = df[["Symbol", "Volume"]].pivot(columns = "Symbol", values = "Volume")
```

*# Open value dataframe*

```
open_val.head()
```

*# Close value dataframe*

```
close_val.head()
```

```
close_val.info()
```

```
next_val = (close_val.shift(-1) / close_val - 1)
```

*# The last day return looks something like this*

```
next_val.tail()
```

```

ret_val = (close_val / close_val.shift(1)) - 1

# The current day return looks something like this

ret_val.head()

# Dataframe with ratio of high/low for each symbols

hl_val = high_val / low_val
hl_val.head()

## Data Pre-processing

# Histogram for non-missing values for 488 days

import matplotlib.pyplot as plt
hist = close_val.notna().sum().hist(bins=10)
plt.xlabel("Observations")
plt.ylabel("Symbols")
plt.title("Non-missing values in Symbols")
print("There are {} symbols with full data
available.".format(close_val.columns[(close_val.notna().sum() ==
close_val.shape[0]).values].shape[0]))
plt.show()

close_val.notna().sum()

# Lets' take Symbols with more than 400 non-missing values

valid_sym = close_val.columns[(close_val.notna().sum() >= 400).values]
open_val = open_val[valid_sym]
high_val = high_val[valid_sym]
low_val = low_val[valid_sym]
close_val = close_val[valid_sym]
volume = volume[valid_sym]
next_val = next_val[valid_sym]
ret_val = ret_val[valid_sym]
hl_val = hl_val[valid_sym]

# The valid symbols looks something like this..

valid_sym
.

corr_val = pd.DataFrame()

for Symbol in valid_sym:
    df = pd.concat([open_val[Symbol], high_val[Symbol], low_val[Symbol], close_val[Symbol],
volume[Symbol], next_val[Symbol], ret_val[Symbol], hl_val[Symbol]], axis=1,
keys=["open", "high", "low", "close", "volume", "next_val", "ret_val", "hl_val"])
    corr_val = corr_val.append({"symbol": Symbol, "avgccorr": df.corr().drop("next_val", axis =

```

```
1).loc['next_val'].abs().mean()), ignore_index = True)
```

```
# Histogram of values of average correlation to next day return
corr_val.avgcorr.hist();
plt.xlabel("Average correlation")
plt.ylabel("Frequency")
plt.title("Average correlation to next day return")
plt.show()
```

Based on the histogram above, we can see that there are very few symbols with good correlation to next day values. For high profitability, we are taking the symbols that possess good predictive power.

Let's consider the tickers with average correlation of more than the 90%.

```
sym90 = corr_val[corr_val.avgcorr > corr_val.avgcorr.quantile(0.90)].symbol.values
```

```
open_val = open_val[sym90]
high_val = high_val[sym90]
low_val = low_val[sym90]
close_val = close_val[sym90]
volume = volume[sym90]
next_val = next_val[sym90]
ret_val = ret_val[sym90]
hl_val = hl_val [sym90]
```

```
# Modify original dataframes
```

```
close_val = close_val.fillna(method="ffill") # close
close_val = close_val.fillna(method="bfill") # to handle the first row of close
volume = volume.applymap(lambda x: 0 if pd.isna(x) is True else x) # volume
open_val = open_val.fillna(close_val)
high_val = high_val.fillna(close_val)
low_val = low_val.fillna(close_val)
```

```
# calculate other dataframes
next_val = (close_val.shift(-1) / close_val) - 1 # future return
ret_val = (close_val / close_val.shift(1)) - 1 # return
hl_val = high_val / low_val # high/low
```

```
## Technical Analysis
```

```
# Technical Analysis package
```

```
import ta
```

```
##### 1. Money flow index
```

```
import ta
```

```
mfi = pd.DataFrame()
```

```
for Symbol in close_val.columns:
```

```

temp = ta.momentum.money_flow_index(high=high_val[Symbol], low=low_val[Symbol],
close=close_val[Symbol], volume=volume[Symbol], fillna=True)
mfi = pd.concat([mfi, temp], axis=1,sort=True)

```

```

# renaming the columns
mfi.columns = close_val.columns

```

#### ##### 2. Relative Strength Index

```

rsi = close_val.apply(ta.momentum.rsi, fillna=True)

```

#### ##### 3. Stochastic oscillator

```

stoch_k = pd.DataFrame()
stoch_d = pd.DataFrame()

```

```

for Symbol in close_val.columns:
    temp = ta.momentum.stoch(high=high_val[Symbol], low=low_val[Symbol],
close=close_val[Symbol], fillna=True)
    stoch_k = pd.concat([stoch_k, temp], axis=1,sort=True)

```

```

temp = ta.momentum.stoch_signal(high=high_val[Symbol], low=low_val[Symbol],
close=close_val[Symbol], fillna=True)
stoch_d = pd.concat([stoch_d, temp], axis=1,sort=True)

```

```

# renaming the columns
stoch_k.columns = close_val.columns
stoch_d.columns = close_val.columns

```

#### ##### 4. William %R

```

will_r = pd.DataFrame()

```

```

for Symbol in close_val.columns:
    temp = ta.momentum.wr(high=high_val[Symbol], low=low_val[Symbol], close=close_val[Symbol],
fillna=True)
    will_r = pd.concat([will_r, temp], axis=1,sort=True)

```

```

# renaming the columns
will_r.columns = close_val.columns

```

#### ##### 5. Exponential moving average

```

ema = close_val.apply(ta.trend.ema_indicator, fillna=True)

```

#### ##### 6. Moving average convergence-divergence

```

macd = close_val.apply(ta.trend.macd_diff, fillna=True)

```

#### ##### 7. Commodity channel index

```

cci = pd.DataFrame()

for Symbol in close_val.columns:
    temp = ta.trend.cci(high=high_val[Symbol], low=low_val[Symbol], close=close_val[Symbol],
fillna=True)
    cci = pd.concat([cci, temp], axis=1, sort=True)

# renaming the columns
cci.columns = close_val.columns

##### 8. Ichimoku Indicator

ichi_a = pd.DataFrame()
ichi_b = pd.DataFrame()

for Symbol in close_val.columns:
    temp = ta.trend.ichimoku_a(high=high_val[Symbol], low=low_val[Symbol], fillna=True)
    ichi_a = pd.concat([ichi_a, temp], axis=1, sort=True)

    temp = ta.trend.ichimoku_b(high=high_val[Symbol], low=low_val[Symbol], fillna=True)
    ichi_b = pd.concat([ichi_b, temp], axis=1, sort=True)

# renaming the columns
ichi_a.columns = close_val.columns
ichi_b.columns = close_val.columns

##### 9. Accumulation/distribution index

ad = pd.DataFrame()

for Symbol in close_val.columns:
    temp = ta.volume.acc_dist_index(high=high_val[Symbol], low=low_val[Symbol],
close=close_val[Symbol], volume=volume[Symbol], fillna=True)
    ad = pd.concat([ad, temp], axis=1, sort=True)

# renaming the columns
ad.columns = close_val.columns

##### 10. Bollinger bands

bb_up = close_val.apply(ta.volatility.bollinger_hband, fillna=True)
bb_down = close_val.apply(ta.volatility.bollinger_lband, fillna=True)

## Data Analysis

# Listing all the above needed dataframes

dfois = [open_val, high_val, low_val, close_val, volume, ret_val, hl_val,
mfi, ema, rsi, stoch_k, stoch_d, macd, will_r, cci,
ichi_a, ichi_b, ad, bb_up, bb_down, next_val]
dfois_str = ['open', 'high', 'low', 'close', 'volume', 'rtn', 'hl', 'mfi', 'ema', 'rsi', 'stoch_k', 'stoch_d', 'macd',
'will_r', 'cci', 'ichi_a', 'ichi_b', 'ad', 'bb_up', 'bb_down', 'next_val']

```

```

import seaborn as sns

corr_val = np.empty([len(dfois),len(dfois)])

# Loop over every dataframe and find correlation

for i, df1 in enumerate(dfois):
    for j, df2 in enumerate(dfois):
        corr_val[i][j] = df1.corrwith(df2).mean()

fig, ax = plt.subplots(figsize=(15,15))
df_corr = pd.DataFrame(corr_val, columns=dfois_str, index=dfois_str)

# Heatmap using seaborn

sns.heatmap(df_corr, vmin=-1, vmax=1, center=0, cmap=sns.diverging_palette(10, 240, n=500),
ax=ax)

ax.set_xticklabels(ax.get_xticklabels(), rotation=45, horizontalalignment='right');

ax.set_title('Avg. colleration matrix heatmap');

from sklearn import preprocessing

# Normalise and draw subplots for each feature

fig, axs = plt.subplots(10,2, figsize=(20, 30))
for i, ax in enumerate(axs.flatten()):
    if i >= 20:
        pass
    else:
        nomaliser = preprocessing.MinMaxScaler(feature_range=(0,1))
        df_val = nomaliser.fit_transform(dfois[i])
        ax.hist(df_val.reshape(df_val.shape[0]*df_val.shape[1]), bins=50)
        ax.set_title(dfois_str[i])
        ax.set_ylabel('Frequency')
        ax.grid()

# Create target dataframe
df_target = (next_val > 0).astype(int)

# Look at the proportion
values, counts = np.unique(df_target.values.reshape(df_target.shape[0]*df_target.shape[1]),
return_counts=True)
plt.bar(values,counts,tick_label=['DOWN','UP'])
plt.title('Number of UPs and DOWNS')
plt.ylabel('Total number')
plt.xlabel('Categories')
plt.show()

# List all the dataframes of interest

```

```

dfois = [close_val, ret_val, ema, rsi, stoch_k, stoch_d, macd, cci, ichi_a, ichi_b, bb_up, bb_down,
df_target]
dfois_str = ['close', 'rtn', 'ema', 'rsi', 'stoch_k', 'stoch_d', 'macd', 'cci', 'ichi_a', 'ichi_b', 'bb_up',
'bb_down', 'target']

## Prediction

##### Pre-processing the data

# Train set
dfois_train = []
for df in dfois:
    dfois_train.append(df.iloc[df.index < '2019-03-01'])

# Test set
dfois_test = []
for df in dfois:
    dfois_test.append(df.iloc[df.index >= '2019-08-01'])

# Validation set
dfois_eval = []
for df in dfois:
    dfois_eval.append(df.iloc[(df.index >= '2019-03-01') & (df.index < '2019-08-01')])

# List of normalisers corresponding to each dataframe
normalisers = []

for i, df in enumerate(dfois[:-1]):
    # create the normaliser object
    normalisers.append(preprocessing.MinMaxScaler(feature_range=(0,1)))

    # columns and indexes
    columns = dfois_train[i].columns
    index_train = dfois_train[i].index
    index_test = dfois_test[i].index
    index_eval = dfois_eval[i].index

    # fit normalise
    normalisers[i].fit(dfois_train[i])

    # transform
    train_data = normalisers[i].transform(dfois_train[i])
    eval_data = normalisers[i].transform(dfois_eval[i])
    test_data = normalisers[i].transform(dfois_test[i])

    # replace list
    dfois_train[i] = pd.DataFrame(train_data, columns=columns, index=index_train)
    dfois_eval[i] = pd.DataFrame(eval_data, columns=columns, index=index_eval)
    dfois_test[i] = pd.DataFrame(test_data, columns=columns, index=index_test)

# List of scalers corresponding to each dataframe
scalers = []

```

```

# Standardise the data by fitting the train set
for i, _ in enumerate(dfois[:-1]):
    # create the Scaler object
    scalers.append(preprocessing.StandardScaler())

    # columns and indexes
    columns = dfois_train[i].columns
    index_train = dfois_train[i].index
    index_test = dfois_test[i].index
    index_eval = dfois_eval[i].index

    # fit scale
    flat_arr = dfois_train[i].values.reshape(dfois_train[i].shape[0]*dfois_train[i].shape[1],1)
    scalers[i].fit(np.tile(flat_arr, dfois_train[i].shape[1]))

    # transform
    train_data = scalers[i].transform(dfois_train[i])
    eval_data = scalers[i].transform(dfois_eval[i])
    test_data = scalers[i].transform(dfois_test[i])

    # replace list
    dfois_train[i] = pd.DataFrame(train_data, columns=columns, index=index_train)
    dfois_eval[i] = pd.DataFrame(eval_data, columns=columns, index=index_eval)
    dfois_test[i] = pd.DataFrame(test_data, columns=columns, index=index_test)

##### 60 Time Steps

from collections import deque
import random

# Look at the past 60 days
SEQ_LEN = 60

def sequence_data(df_list, shuffle=True):
    # list containing the data
    sequential_data = []

    for Symbol in close_val.columns:
        # initialise dataframe
        df_ticker = pd.DataFrame()

        # concatenate the dataframes
        for df in df_list:
            df_ticker = pd.concat([df_ticker, df[Symbol]], axis=1)

        prev_days = deque(maxlen=SEQ_LEN)
        # for values in every row
        for i in df_ticker.values:
            # remove the targets
            prev_days.append([n for n in i[:-1]])
            # append when sequence length is reached

```



```

        if len(prev_days) == SEQ_LEN:
            sequential_data.append([np.array(prev_days), i[-1]])

    # shuffle - we do not need to do this for test set
    if shuffle == True:
        random.shuffle(sequential_data)

    return sequential_data

sequential_data_train = sequence_data(dfois_train)
sequential_data_eval = sequence_data(dfois_eval)
sequential_data_test = sequence_data(dfois_test, shuffle=False) # do not shuffle just this one

# Print the length
print('Training data length: {}'.format(len(sequential_data_train)))
print('Validation data length: {}'.format(len(sequential_data_eval)))
print('Testing data length: {}'.format(len(sequential_data_test)))

# balance train and evaluation data
def balance_data(sequential_data):
    ups = []
    downs = []

    # separate the sequence into ups and downs
    for seq, target in sequential_data:
        if target == 0:
            downs.append([seq, target])
        elif target == 1:
            ups.append([seq, target])

    # shuffle to randomise
    random.shuffle(ups)
    random.shuffle(downs)

    # get the shorter length
    lower = min(len(ups), len(downs))

    # truncate the list to shorter length
    ups = ups[:lower]
    downs = downs[:lower]

    # merge and shuffle
    sequential_data = ups+downs
    random.shuffle(sequential_data)

    return sequential_data

# separate train and target data
def separate_data(sequential_data):
    X = []
    y = []

```

```

# loop over every row in sequential data
for seq, target in sequential_data:
    X.append(seq)
    y.append(target)

return np.array(X), y

# perform balancing by calling the function
train_x, train_y = separate_data(balance_data(sequential_data_train))
validation_x, validation_y = separate_data(balance_data(sequential_data_eval))

import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, Dropout, LSTM, BatchNormalization
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.callbacks import ModelCheckpoint, ModelCheckpoint

# Batch size and epochs
BATCH_SIZE = 512
EPOCHS = 3

# Build LSTM prediction model
model = Sequential()

model.add(LSTM(128, input_shape=(train_x.shape[1:]), activation='tanh', return_sequences=True))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(LSTM(128, activation='tanh', return_sequences=True))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(LSTM(128, activation='tanh'))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(2, activation='softmax'))

# Compile model
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(lr=0.001, decay=1e-6),
              metrics=['accuracy'])

# Model summary
print(model.summary())

history = model.fit(train_x, train_y,
                    batch_size=BATCH_SIZE,
                    epochs=EPOCHS,

```

```

validation_data=(validation_x, validation_y))

# save the model
model.save("lstm.l1")

# load the model
model = load_model("lstm.l1")

from sklearn.metrics import classification_report

# test performance using test set
test_x, test_y = separate_data(sequential_data_test)

# get the prediction
pred = model.predict_classes(test_x)

# get prediction probability
pred_proba = model.predict(test_x)

# accuracy using test
score = model.evaluate(test_x, test_y, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
print("")

# classification report
print('Classification report:')
print(classification_report(test_y, pred))

# build index list
pred_index = dfois_test[0].index[dfois_test[0].index >= dfois_test[0].head(60).index[-1]]

# build prediction dataframe - 1 is BUY and 0 is SELL
df_pred = pd.DataFrame(pred.reshape(dfois_test[0].shape[0]-SEQ_LEN+1,dfois_test[0].shape[1],
order='F'),
                        index = pred_index,
                        columns = dfois_test[0].columns)

# dataframe for right prediction - 1 is CORRECT and 0 is WRONG
df_right = (df_pred.astype(bool) == df_target.loc[pred_index].astype(bool)).astype(int)

# dataframe for wrong prediction - 1 is WRONG and 0 is CORRECT
df_wrong = (~df_right.astype(bool)).astype(int)

# dataframe for profit
df_profit = df_right*next_val.loc[pred_index].abs()

# dataframe for loss
df_loss = df_wrong*next_val.loc[pred_index].abs()

plt.subplots(figsize=(15,5))

```

```

plt.plot((df_profit.sum(axis=1) - df_loss.sum(axis=1)).cumsum())
plt.grid()
plt.xlabel("Trading dates")
plt.ylabel("Cumulative PnL (AUD)")
plt.title("Graph of cumulative profit and loss (PnL)")
plt.show()

```

```

from datetime import timedelta

```

```

# pnl of lstm
pnl_lstm = (df_profit.sum(axis=1) - df_loss.sum(axis=1))

# plotting the roi per day
plt.subplots(figsize=(16, 24))
plt.barh((pnl_lstm/150*100).index,(pnl_lstm/150*100).values, label = "LSTM", align='edge',
height=0.25)
plt.xlabel("ROI per day (%)")
plt.ylabel("Trading dates")
plt.title("Graph of return on investment (%) per day for different strategies")
plt.legend()
plt.grid()
plt.show()

```

Following the simple strategy, based on 40 days of trading and 188 investment per day, the total profit made is about 6. Considering the numbers and the linear trend of the cumulative PnL, I would say that this is a successful strategy.

```

fig, (ax1) = plt.subplots(1,1, figsize=(15, 5), sharey=True)

```

```

ax1.hist((pnl_lstm/188*100).values, bins=10)
ax1.set_title("LSTM")
ax1.set_ylabel('Frequency')
ax1.set_xlabel('ROI (%)')
ax1.grid()

```

```

from scipy import stats

```

```

# calculating the confidence
conf_lstm = stats.percentileofscore((pnl_lstm/188*100).values, 2) -
stats.percentileofscore((pnl_lstm/188*100).values, 0)

print('Confidence of employed approaches in making positive ROI based on test set:')
print('LSTM: {}'.format(conf_lstm))

```

## 12. REFERENCES

- [1] Borovkova S, Tsiamas I. An ensemble of LSTM neural networks for high-frequency stock market classification. *Journal of Forecasting*. 2019;1–20. <https://doi.org/10.1002/for.2585>
- [2] Zhai Y, Hsu A, Halgamuge SK. Combining News and Technical Indicators in Daily Stock Price Trends Prediction. *Lecture Notes in Computer Science*. 2007; [https://link.springer.com/chapter/10.1007/978-3-540-72395-0\\_132](https://link.springer.com/chapter/10.1007/978-3-540-72395-0_132)
- [3] Hegazy O, Soliman OS, Salam MA. A Machine Learning Model for Stock Market Prediction. *International Journal of Computer Science and Telecommunications*. 2013; 17–23. [https://www.ijcst.org/Volume4/Issue12/p4\\_4\\_12.pdf](https://www.ijcst.org/Volume4/Issue12/p4_4_12.pdf)
- [4] Vargas MR, dos-Anjos CEM, Bichara GLG, Evsukoff AG. Deep Learning for Stock Market Prediction Using Technical Indicators and Financial News Articles. 2018; <https://doi.org/10.1109/IJCNN.2018.8489208>
- [5] Padial DL. Technical Analysis Library in Python. 2019; <https://github.com/bukosabino/ta>
- [6] Selvin S, Vinayakumar R, Gopalakrishnan E.A, Menon VK, Soman K.P. Stock price prediction using LSTM, RNN and CNN-sliding window model. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2017; <https://doi.org/10.1109/ICACCI.2017.8126078>
- [7] Chen S, He H. Stock Prediction Using Convolutional Neural Network. *IOP Conference Series: Materials Science and Engineering*. 2018; <https://doi.org/10.1088/1757-899X/435/1/012026>
- [8] Weng B, Lu L, Wang X, Megahed FM, Martinez W. Predicting Short-Term Stock Prices Using Ensemble Methods and Online Data Sources. *Expert Systems with Applications*. 2018; <https://doi.org/10.1016/j.eswa.2018.06.016>