Design Oriented Project
Report

Stock Market Prediction

Submitted in partial fulfillment of
the requirements for

BITS F376
in
Birla Institute of Technology and Science

Submitted by

Names of Students	Roll No
Rishab Khincha	2016B5A70517G
Soundarya Krishnan	2016B5A70472G

Under the guidance of
Prof. Kinjal Banerjee

Department of Physics
Birla Institute of Technology and Science Goa
Goa, India – 403726

II Semester 2018

Contents

List of Figures

Chapter 1

Introduction

Predicting the Stock Market has been the one of the main goals of investors. Billions of dollars are traded every day and behind each dollar is an investor hoping to profit in some way. Predicting stock markets has the ability to make entire companies rise and fall. Prediction of stock market data is a growing field and we attempt to see if this can be done by testing various methods. In this report, we will attempt to study and analyse aspects of the SENSEX market. We will start by predicting the closing price of various sectors of SENSEX using Neural Networks. We will do this by only observing the pattern of rise and fall of the prices and not by examining their causes. Next, we will study applications of the Log Periodic Power Law and test it on previous SENSEX crashes. The report describes the various methods used, results obtained, their drawbacks as well as future plans.

Chapter 2

Work Done

2.1 Closing Price Predictions

Here we use a nonlinear autoregressive with exogenous input. In other words, we predict future values of the closing price using past values of the closing price as well as other inputs, namely, opening price, high price and low price. We use the past 4 days data in order to predict the 5th day. The Levenberg Marquardt algorithm was used as it provides the most accurate results. It takes lesser time but uses more memory in general, but the memory usage was not an issue in this case. Bayesian regularisation was not used the dataset didnt have much noise. The Neural Network has 15 hidden neurons and uses a delay of 4 days for prediction. 70% of the data was used for training, and 15% each for validation and testing.

Another side project was to find the effect of changing the number of days used to predict.

No. of days delay	Squared Error
1	1.2582e+06
2	1.5327e+06
4	3.5324e+05
9	5.1254e+06
14	9.5729e+06
21	1.3648e+07

Thus, we see that the error actually increases when we include more than

4 days for training the Neural Network. We see that a possible explanation for this could be that taking lesser than four days leads to under-fitting, whereas, by taking more than 4 days, we may risk over-fitting the data.

This method was performed in real time and the predicted data was put on a Google sheet so as to avoid any chances of unintentional manipulation.

2.1.1 Code

```
FULL=xlsread('SENSEX 1YR.csv');
X_in=FULL(1:248, 1:3);
Y_in=FULL(1:248, 4);

X = tonndata(X_in,false,false);
T = tonndata(Y_in,false,false);

trainFcn = 'trainlm'; % Levenberg-Marquardt backpropagation.

% Create a Nonlinear Autoregressive Network with External Input
inputDelays = 1:4;
feedbackDelays = 1:4;
hiddenLayerSize = 15;
net =
    narxnet(inputDelays,feedbackDelays,hiddenLayerSize,'open',trainFcn);

 [x,xi,ai,t] = preparets(net,X,{},T);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t,xi,ai);

% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y);

% View the Network
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
```

```
%view(netc)
[xc,xic,aic,tc] = preparets(netc,X,{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc);

nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
%view(nets)
[xs,xis,ais,ts] = preparets(nets,X,{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys);


answer=cell2mat(ys);
%figure
Y_in_new=Y_in(4:248+j);

predictions(i)=answer(245+j);
plot(Y_in_new, 'r')
hold on
plot(answer, 'b')
```

2.1.2 Results

We tested this algorithm on 6 sectors of SENSEX. We were able to predict the future closing price of data with an error of about 0.83%. The following tables describe the errors in detail for 3 chosen days.

Table 2.1: 25-01-2018

Sector	Predicted	Actual	Percentage error
SENSEX	36222	36050	0.475
POWER	2374.2	2347.5	1.123
OIL&GAS	16298	16241	0.350
IT	12614	12688	0.587
AUTO	25902	25670.02	0.896
BANKEX	31051	31082.14	0.100

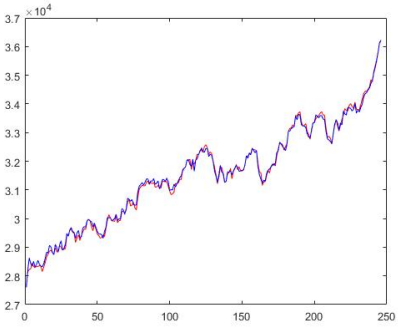
Table 2.2: 29-01-2018

Sector	Predicted	Actual	Percentage error
SENSEX	36226	36283	0.157
POWER	2348.4	2330	0.784
OIL&GAS	16216	16103	0.697
IT	12991	12835	1.201
AUTO	25789	26081	1.132
BANKEX	30741	31125	1.249

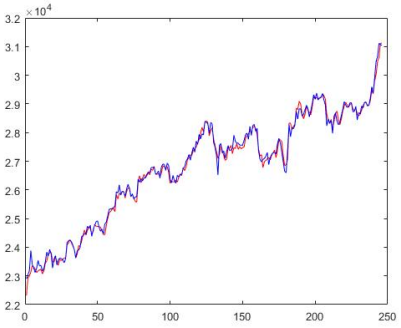
Table 2.3: 30-01-2018

Sector	Predicted	Actual	Percentage error
SENSEX	37032	36033.73	2.770
POWER	2332.6	2327.66	0.212
OIL&GAS	16129	16290.56	0.992
IT	12705	12697.25	0.061
AUTO	26223	25956.89	1.025
BANKEX	31222	30861.34	1.169

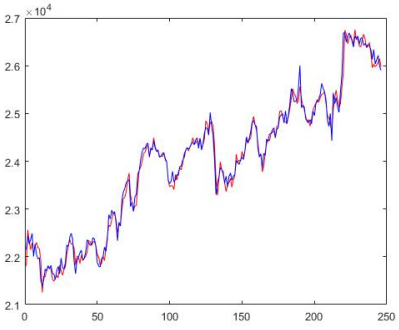
Let us look at these results graphically. The blue line is the original data and the red line is the predicted data.



(a) Sensex 2017



(b) Sensex Bankex 2017



(c) Sensex Auto 2017

2.1.3 Drawbacks:

The main drawback with this type of prediction is that predicting an exact value for what is inherently a probabilistic outcome doesnt make much sense. Running the Neural network several times gave different outputs as it started with a random initialisation. Another possible issue could be that the neural network is overfitting the data and thus will not give us the accurate output on data points outside the training set. Predictions within a particular range of values would be more useful.

2.2 Monte Carlo Simulations

To cover up the drawbacks of the previous method, Monte Carlo Simulation was attempted. Monte Carlo simulation performs risk analysis by building models of possible results by substituting a range of values for any factor that has inherent uncertainty. Before, the Neural Network algorithm predicted a different value each time due to its random initialisation. Through this method, we attempt to rectify this issue by running the NN code multiple times and generating a mean and standard deviation, and then checking if the actual closing price fell in between the error bars. This method was again performed in real time and the predicted data was put on a google sheet so as to avoid any chances of unintentional manipulation.

In the following code, we have run the previous Neural Network code in 2 loops, one for the number of iterations we require to get a satisfactory mean and standard deviation, and one for the extra number of days added. Finally, the data was sent to an excel sheet for ease in analysis.

2.2.1 Code

```
clearvars;
FULL=xlsread('SENSEX 1YR.csv');
Y_last=FULL(:,4);
days=248;
%mc_iter=500;
mc_iter=1;
feed_days=247;
for j=1:days
for i=1:mc_iter
X_in=FULL(1:feed_days-1+j, 1:3);
Y_in=FULL(1:feed_days-1+j, 4);
```

```
X = tonndata(X_in,false,false);
T = tonndata(Y_in,false,false);
inputDelays = 1:4;
feedbackDelays = 1:4;
hiddenLayerSize = 15;
net =
    narxnet(inputDelays,feedbackDelays,hiddenLayerSize,'open',trainFcn);

 [x,xi,ai,t] = preparets(net,X,{},T);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t,xi,ai);

% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y);

% connection from the outout layer.
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
%view(netc)
[xc,xic,aic,tc] = preparets(netc,X,{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc);

nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
%view(nets)
[xs,xis,ais,ts] = preparets(nets,X,{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys);


answer=cell2mat(ys);

predictions(i)=answer(feed_days-4+j);
end
```

```
Final=(predictions);
Final2=Final';
finalmean(j)=mean(Final2);
finalstddev(j)=std(Final2);
end
%output to excel
T=table(Y_last(feed_days+1:feed_days+1+days-1), finalmean',
    finalstddev',abs(Y_last(feed_days+1:feed_days+1+days-1)-finalmean'),abs(Y_last(feed
T.Properties.VariableNames = {'y_in',
    'y_pred','stddev','diff','error'};
filename = 'writetest1.xlsx';
writetable(T,filename,'Sheet',1,'Range','A1');
```

2.2.2 Results

With Monte Carlo Simulation for 2nd Jan 2018 to 22nd Jan 2018, 10 out of 15 instances were within the predicted error margin.

Table 2.4: 30-01-2018

Date	μ	σ	Actual Value	% Error	Difference in values
Jan 2	33627	172.278	33812.26	0.55	185.260
Jan 3	33733	145.885	33793.38	0.18	60.380
Jan 4	33706	114.941	33969.64	0.788	263.640
Jan 5	33928	153.019	34153.85	0.668	225.850
Jan 8	34087	128.910	34352.79	0.78	265.790
Jan 9	34300	173.026	34443.19	0.412	143.190
Jan 10	34413	207.8147	34433.07	0.06	20.07
Jan 11	34378	181.0529	34503.49	0.37	125.49
Jan 12	34476	165.5797	34592.39	0.34	116.39
Jan 15	34590	189.6329	34843.51	0.73	253.51
Jan 16	34780	225.4655	34771.05	0.03	8.95
Jan 17	34966	199.1183	35081.82	0.333	115.82
Jan 18	35273	291.4467	35260.29	0.04	12.71
Jan 19	35428	267.704	35511.58	0.23	83.58
Jan 22	35681	309.2734	35798.01	0.33	117.01

2.2.3 Drawbacks:

The greatest disadvantage of Monte Carlo simulation is that the output is only as good as the inputs. Another great disadvantage is that it tends to underestimate the probability of crises. We want a method to predict crashes by observing the closing prices, by checking how much they grow and oscillate. Thats why next, we move over to the log periodic power law.

2.3 Prediction of crashes using Log Periodic Power Law:

More than predicting the exact closing price of a particular day, it would be more advantageous to predict crashes, so as to help investors save millions. We use the log periodic cost function for the same. In life, we are always interested in the fracture of things. Based on the careful analyses of several earthquakes, physicists have found that just before a crash, there is a transient regime during which the system grows itself, which leads to an exponential growth involving positive feedbacks. This growth is unsustainable and often leads to bursts as crashes. Thus, the intrinsic logic of crashes, be it earthquakes or the financial market, can be approximated by the same function. This function, stated here without deriving, is the following:

$$ln[p(t)] = A + B_0(t - t_c)^{\beta}\{1 + Ccos[\omega ln(t - t_c) + \phi]\}$$

where:

$ln[p(t)]$ = natural logarithm of price p at time t

t_c = the 'critical time' i.e. the time of most probable crash

β = exponential price growth with constraint $0 \leq \beta \leq 1$

ω = oscillation frequency with constraint $2 \leq \omega \leq 20$

ϕ = fixed phase parameter with constraint $0 \leq \phi \leq 2\pi$

A = a constant equal to $ln[p(t)]$ at critical time t_c

B_0 = a constant embodying the scale of power law

C = constant that encapsulates the magnitude of the oscillation around the price growth with constraint $-1 \leq C \leq 1$

The constraints on β are such that they show an acceleration of the log price that the faster than an exponential.

2.3.1 The local minima problem:

However, when practically applied, this cost function poses a problem due to the several local minima that the function can converge to.

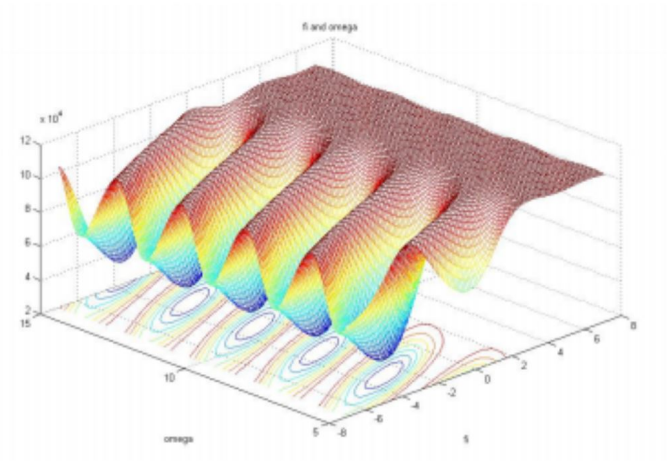


Figure 2.2: Cost function when omega and phi are changing

It is thus very easy to get stuck in a local minimum. Thus, to avoid this, we use an algorithm to find the global minimum.

2.3.2 Code:

This part of the code attempts to find the global minimum of the fitted cost function using MultiSearch and then outputs the parameters estimated. An additional ongoing project is to check the variation of different parameters with time. For example, when the data is taken for a moving range of closing prices, we can observe the variation of parameters when a crash is about to occur, thus predicting the crash more effectively.

```
function [] = plotFunc2


gg=172;
check=xlsread('2006-2008.csv');
ydata=check(1:(length(check)-gg),4);
xdata1=[1:(length(check)-gg)]';
xdata=xdata1./(length(check)-gg);
ydata=log(ydata);
A=8.4;
B=1.3;
tc=1.3;
beta=0.2; %0.2 to 0.8
C=2.2;
```

```
omega=8.5;
phi=2.5;
x0=[A B tc beta C omega phi];
fun=@(x,xdata)
    (x(1)+x(2)*(x(3)-xdata).^x(4).*(1+x(5).*cos(x(6).*log(x(3)-xdata)+x(7))));
% x = lsqcurvefit(fun,x0,xdata,ydata);

p0=[5,-1, 1.05, 0.7, 0.5, 12, pi];
lb = [0,-Inf,1,0,-1,2,0];
ub = [5*max(ydata),0,1.35,1.2,1,30,2*pi];


problem =
    createOptimProblem('lsqcurvefit','x0',p0,'objective',fun,'lb',lb,'ub',ub,'xdata',xd

ms = MultiStart('PlotFcns',@gsplotbestf);
ms.Display='off';
[xmulti,errormulti] = run(ms,problem,50);

% disp(xmulti)

% values=fun(x,xdata);
% error=sum((ydata-values).^2)*100/sum(ydata);
times = linspace(xdata(1),xdata(end));
plot(xdata,ydata,'ko',times,fun(xmulti,times),'b-');
legend('Data','Fitted exponential');
title('Data and Fitted Curve');

end
```

2.3.3 Results:

We had collected historical data of various SENSEX crashes in order to test this cost function. We find t_c and check how far away it is from the actual crash.

Before we begin with any analysis, first we need to define a crash. Chances of misidentifying a crash are high due to the fractal-like nature of stock market prices, and we might mistakenly identify a crash in 2 cases:

- When prices are declining and prices suddenly start increasing by a little, but this is only on a short term and the crash continues.

- When prices are dramatically increasing, and there is a large drop, but the prices continue going up anyway.

We understand this issue and need to find a way to define crashes accurately, and this will be taken up in the future. For the time being, we vaguely explain what we mean by a crash with an example. Following is a time series plot of the 2006-2008 crisis period with the crashes circled.

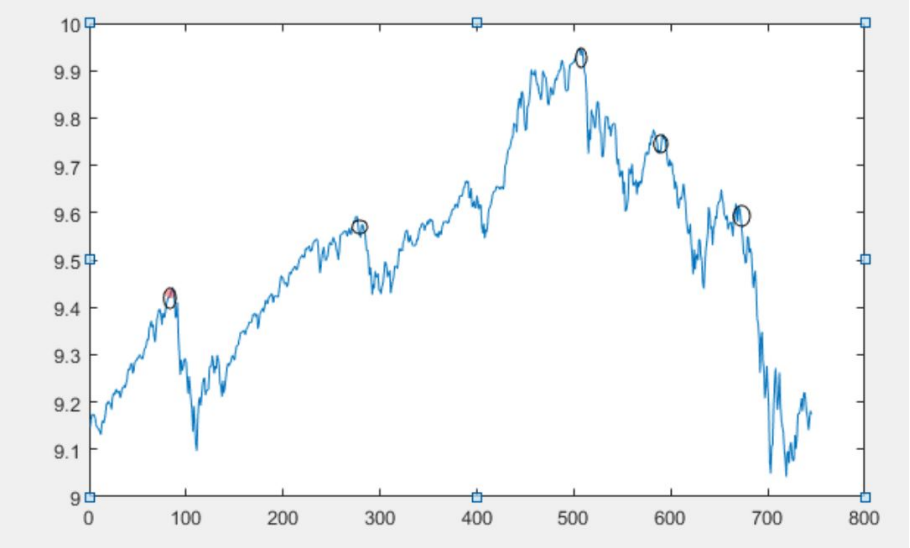


Figure 2.3: Crashes in 2006-2008 crisis period

We proceed to check historical crisis periods to check if the t_c values estimated match the crashes. The results are as follows:

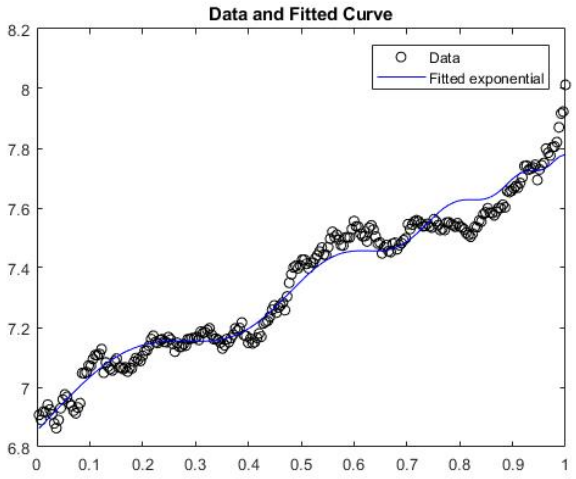


Figure 2.4: 1992 SENSEX Crash

A	B	t_c	β	C	ω	ϕ
7.8503	-0.8725	1.0963	1.0415	0.0941	11.4856	4.0558

Table 2.5: Parameter Values for 1992 crash

Predicted Crash	Actual Crash
22nd April 1992	22nd April 1992

Table 2.6: Crash Date Prediction

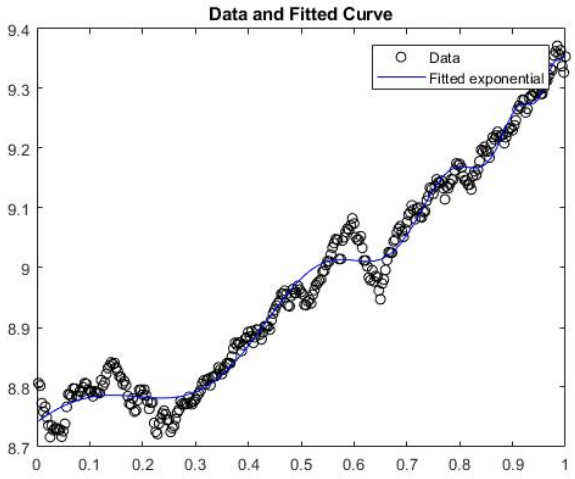


Figure 2.5: 2006 SENSEX crash

Table 2.7: Parameter Values for 2006 crash

A	B	t_c	β	C	ω	ϕ
9.506258	-0.79132	1.053759	0.587741	-0.06967	9.733489	6.218548

Predicted Crash	Actual Crash
9th May 2006	11th May 2006

Table 2.8: 2006 Crash Date Prediction

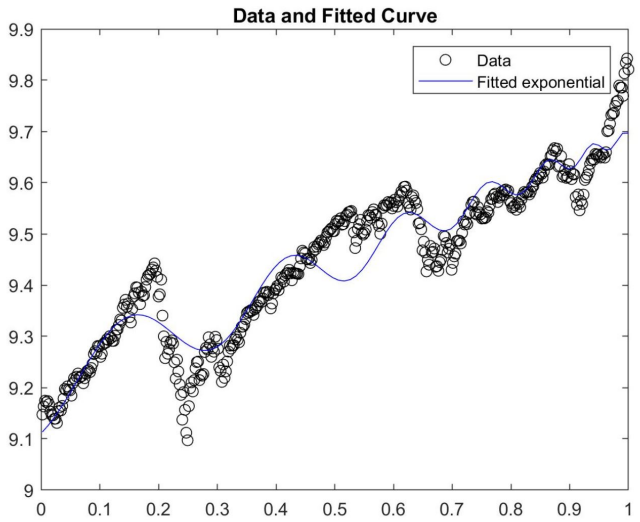


Figure 2.6: 2008 SENSEX crash

A	B	t_c	β	C	ω	ϕ
9.7561	-0.4986	1.1341	1.018	0.1501	19.4192	3.4434

Table 2.9: Parameter Values for 2008 crash

Predicted Crash	Actual Crash
7th January 2008	8th January 2008

Table 2.10: 2008 Crash Date Prediction

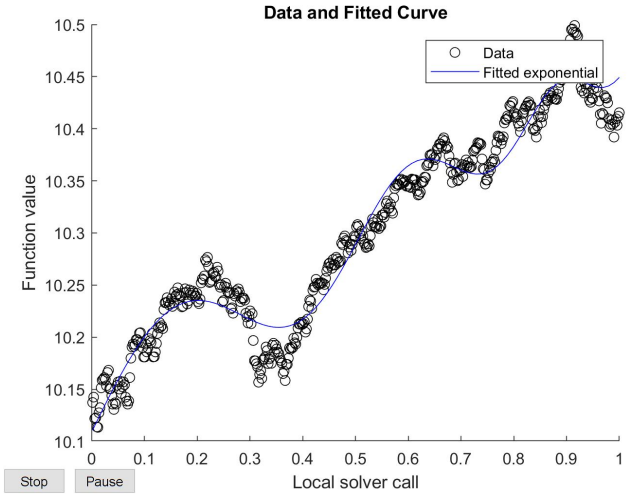


Figure 2.7: Next crash prediction

A	B	t_c	β	C	ω	ϕ
10.5473	-0.3046	1.35	1.2	0.1538	13.1961	0.7815

Table 2.11: Parameter Values for future crash

Predicted Crash	Actual Crash
3rd April 2018	173 days later

Table 2.12: Next crash date prediction

2.3.4 Plotting parameters with time:

An ongoing project is to find how the different parameters change during crisis periods. We're working on this at present and it will help us see how we can predict the crash using parameters other than t_c.

2.3.5 Drawbacks:

- The LPPL was able to predict only those values of tc that are close to 1.
- The accuracy diminishes significantly when the crash time is far away.
- The global search algorithm used still needed to be manipulated to verify the best fit.
- The value of t_c was extremely sensitive to the input. Even slight modification of a single data point was enough to throw the tc haywire.
- The parameters with the least error became complex valued several times.
- The LPPL couldn't be fitted to cryptocurrencies' historical data. This could mean that they follow some other law, or are linear in nature.

Chapter 3

Future Work

Here we discuss several ideas that we plan to take up in the future.

- Plotting various parameters with time to see how they change in crisis periods (as mentioned before).
- A more optimised algorithm for global minima, Genetic Algorithm so as to minimise the manual work in fitting parameters.
- Including traded volumes in stock crash prediction, as the traded volume dramatically changes near the crash period, indicating a crash.
- Including price-earnings ratio in stock crash prediction, so as to give a better feel of whats really going on in the financial market.
- Prediction of stock market behaviour through Natural Language Processing. The internet data can be scraped to find news as well as public opinion about, say, bitcoin, to quantitatively estimate speculation and stock behaviour.

Chapter 4

Conclusion

(CONCLUSION TO BE ADDED)

Acknowledgments

We would like to thank Mr. Chandradew for all his time and patience in guiding us through this project. This would not have been possible without him. We would also like to thank Mr. Kinjal Banerjee for his guidance and giving us the opportunity to do this project.

April 2018
Birla Institute of Technology and Science Goa

References

[1] Monte Carlo Simulations, https://www.investopedia.com/articles/investing/112514/monte-carlo-simulation-basics.asp

[2] What is a bubble? https://www.canterburygroup.com/The-Anatomy-of-a-Market-Bubble-.html

[3] Navigating the World of Crypto https://mynabla.com/2017/11/30/bubble-trouble-exploring-an-lppl-model-for-bitcoin/