Configuration Manual

MSc Research Project
MSc in FinTech

Ranjani Chandrasekaran
Student ID: X18108423

School of Computing
National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ranjani Chandrasekaran
Student ID:	X18108423
Programme:	MSc in FinTech **Year:** 2018-2019
Module:	Research Project
Lecturer:	Victor Del Rosal
Submission Due Date:	12th August 2019
Project Title:	Prediction of Litecoin Prices using ARIMA and LSTM
Word Count:	1083 words **Page Count:** 16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……………………………………………………………………………………………………

Date: 12th August 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ranjani Chandrasekaran
Student ID: X18108423

1 Introduction

This user configuration manual provides a step by step account of the product and process requisites to complete the thesis titled "*What is the impact of ARIMA and LSTM in the level of accuracy for prediction of Litecoin prices*?" The steps also include the hardware and software requirements. Further, samples of the codes that are run in the different models and results are provided for effective guidance.

2 Data Gathering

The data collected is from 2014 to 2019 having 1991 observations. It is collected from coinmarketcap.com. Among the different type of prices that is high, low, open and close, close price is considered as the predictor variable. The data is read in CSV format and formatting of date (pre-processing) is executed.

3 System Setup

The hardware system configuration is Intel core i5+ 8th Gen used with a 4GB ram. The software installed is RStudio and RStudio cloud. For the R studio cloud an account is created to implement the neural network algorithm.

4 Libraries Installed

In RStudio and RStudio cloud relevant libraries are installed to process machine learning algorithms. The Libraries included are CaTools, Libridate, forecast, Mlmetrics, dplyr, grid, stargazer, seasonal, fma, keras, tidyverse.

Using the above setup and running the data in ARIMA and LSTM the results are below-

1. **Importing library**

```
library(fpp2)
library(seasonal)
library(fma)
library(stargazer)
library(grid)
library(forecast)
library(dplyr)
library (MLmetrics)
require(caTools)
require(lubridate)
```

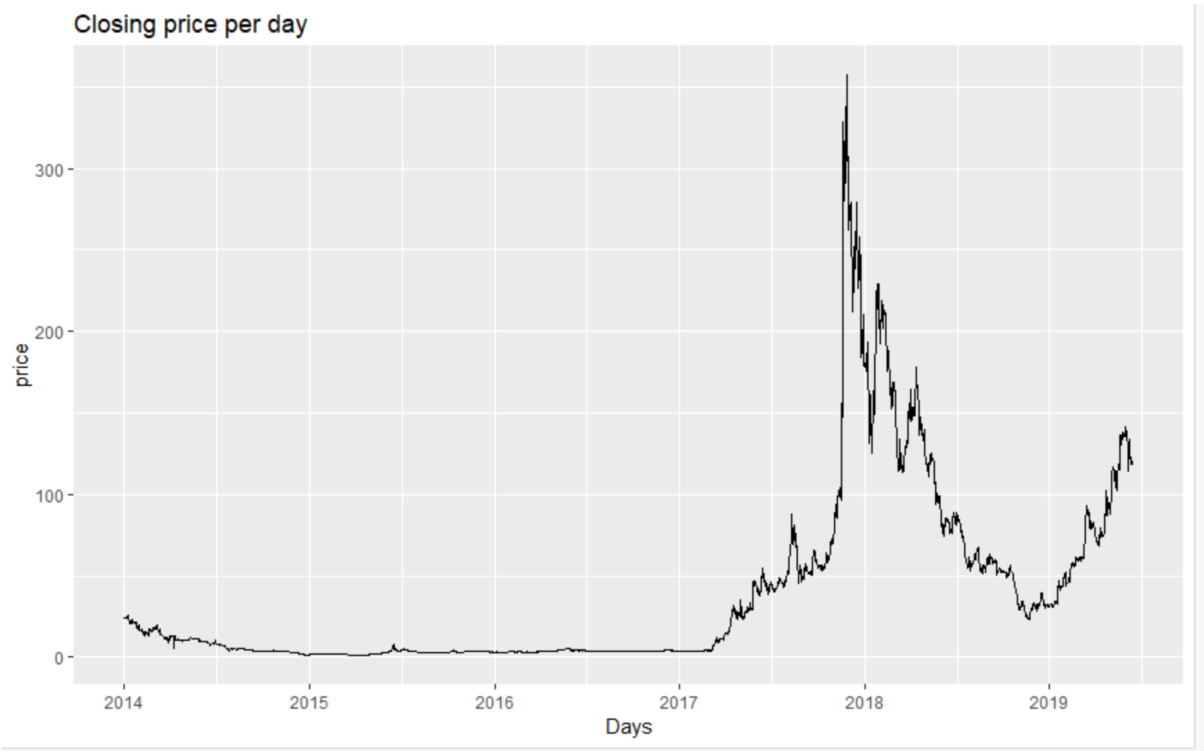
2. **Pre-processing, splitting the data into 0.6 train and 0.4 test data**

```
#Splitting the dataset into train and test data
set.seed(1)
bd$date=parse_date_time(bd$date, orders = c("ymd", "dmy", "mdy"))
bd = bd[,c(1,3)] # we retain the closing price.
#Summary
summary(bd)
```

3. **Converting the data to time series**

```
v1_d = ts(bd[,2], frequency = 365, start = c(2014,1))
```

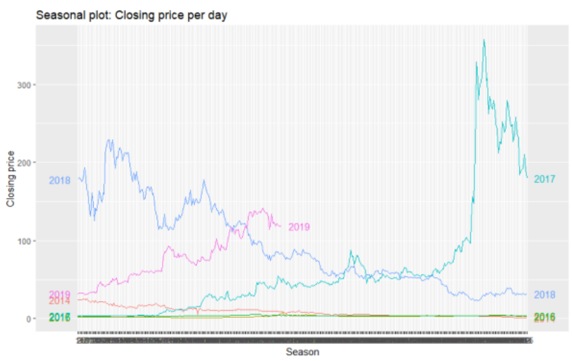
4. **Plotting the data set to examine stationary**

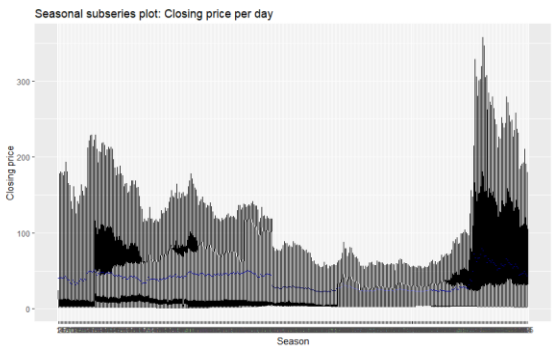


5. Taking a deeper look at the seasonality for which Seasonal plot and seasonal subseries plot has been plotted:

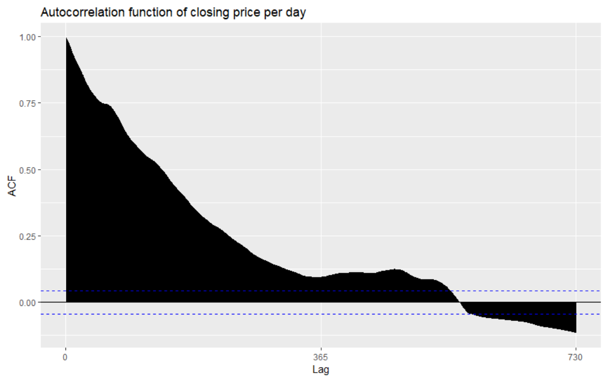
```
# taking a look abt seasonality:
ggseasonplot(v1_d, year.labels=TRUE, year.labels.left=TRUE) +
ylab("Closing price") +
ggtitle("Seasonal plot: Closing price per day")

## taking a deep look about seasonality:
ggsubseriesplot(v1_d) +
ylab("Closing price") +
ggtitle("Seasonal subseries plot: Closing price per day")
```

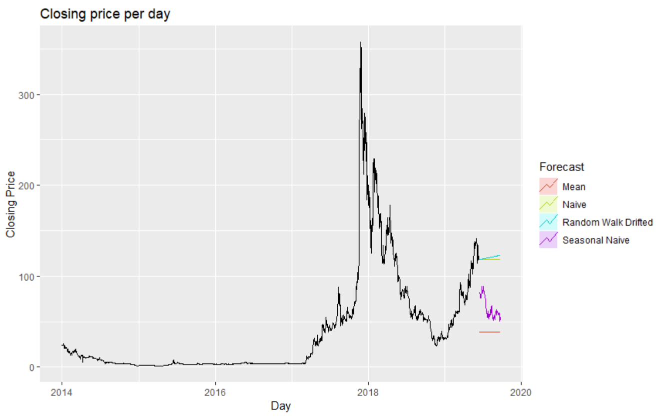




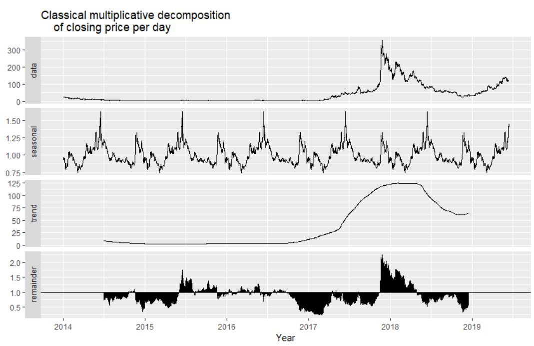
6. Auto correlation



7. Forecasting of mean, random walk



8. Multiplicative decomposition



9. Fit Auto ARIMA in training data set and getting results

fit_arima_auto	list [10] (S3: forecast)	List of length 10
method	character [1]	'ARIMA(3,1,3)'
model	list [18] (S3: ARIMA, forecast_ARI	List of length 18
level	double [2]	80 95
mean	double [30] (S3: ts)	118 117 118 118 118 119 ...
lower	double [30 x 2] (S3: mts, ts, matri:	111.0 107.2 105.9 103.9 101.5 100.9 107.5 101.9 99.5 96.4 92.9 91.5 ...
upper	double [30 x 2] (S3: mts, ts, matri:	124 127 130 132 134 136 128 132 137 140 142 146 ...
x	double [1961] (S3: ts)	24.5 24.5 24.0 24.0 24.9 24.9 ...
series	character [1]	'v1_d_train'
fitted	double [1961] (S3: ts)	24.5 24.5 24.5 24.0 24.0 24.9 ...
residuals	double [1961] (S3: ts)	2.45e-02 -1.05e-06 -4.98e-01 1.18e-02 9.02e-01 -3.72e-02 ...

10. Calculating the absolute value=true value- estimated value

result$V1 = days

colnames(result) = c("days" , "true_value","Estimated_value")

result$absolut_valu = abs(result$true_value-result$Estimated_value)

11. Results:

Filter

	days	true_value	Estimated_value	absolut_valu
1	2019-06-07	118.51	117.7545	0.75548298
2	2019-06-08	114.87	117.0456	2.17560261
3	2019-06-09	129.83	118.0432	11.78676716
4	2019-06-10	136.08	118.0582	18.02183751
5	2019-06-11	136.16	117.5360	18.62403386
6	2019-06-12	130.86	118.5376	12.32241163
7	2019-06-13	132.71	118.0579	14.65210656
8	2019-06-14	138.35	117.9139	20.43606179
9	2019-06-15	136.95	118.7278	18.22224744
10	2019-06-16	134.19	117.9732	16.21681399
11	2019-06-17	135.13	118.2424	16.88758908
12	2019-06-18	136.83	118.7114	18.11863089
13	2019-06-19	135.78	117.9233	17.85666314
14	2019-06-20	139.07	118.5131	20.55686745
15	2019-06-21	141.77	118.5680	23.20197767
16	2019-06-22	136.83	117.9597	18.87032515
17	2019-06-23	135.40	118.6967	16.70332207
18	2019-06-24	135.51	118.3727	17.13734984
19	2019-06-25	130.52	118.0843	12.43574627
20	2019-06-26	114.24	118.7713	4.53129420
21	2019-06-27	119.46	118.1921	1.26789024
22	2019-06-28	133.44	118.2661	15.17391150
23	2019-06-29	122.16	118.7367	3.42326893
24	2019-06-30	122.67	118.0762	4.59375677
25	2019-07-01	118.68	118.4579	0.22208608
26	2019-07-02	121.97	118.6164	3.35357409
27	2019-07-03	119.67	118.0504	1.61958690
28	2019-07-04	118.53	118.6124	0.08241682
29	2019-07-05	118.31	118.4512	0.14120942
30	2019-07-06	118.33	118.1132	0.21679104
31	2019-07-07	118.51	117.7545	0.75548298

LSTM

CONFIRGURATION MANUAL OF LSTM:

1. **Install and import libraries**

```
library(readr)
library(tseries)
library(tidyverse)
library(keras)
require(lubridate)
require(caTools)
```

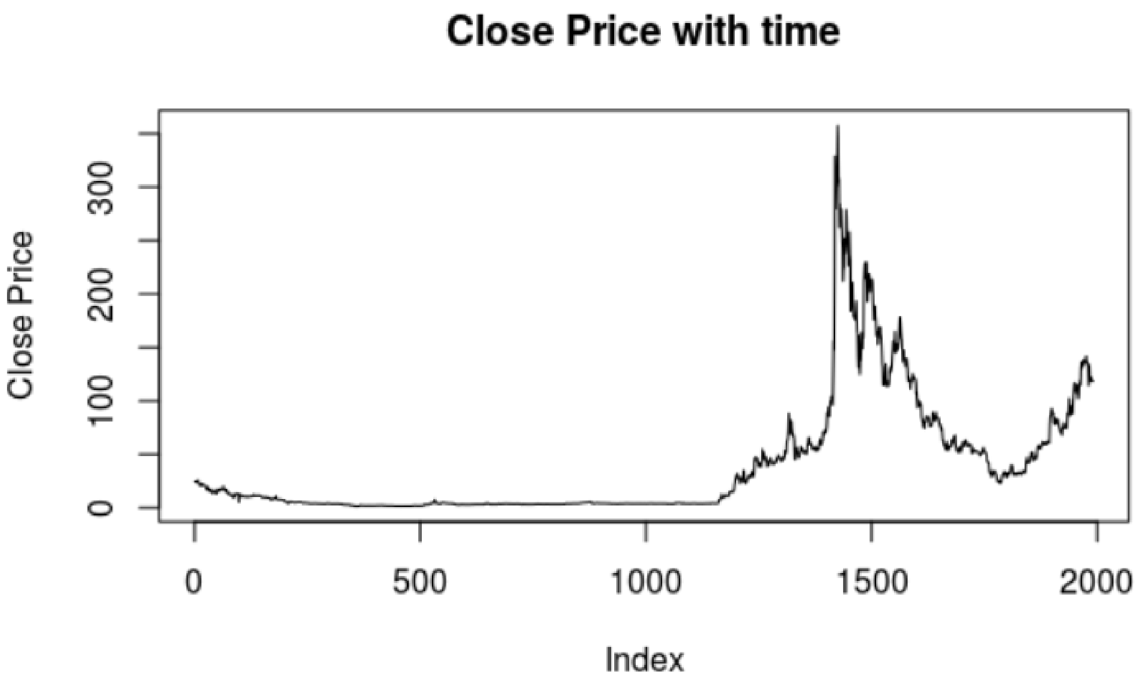
2. **Loading and reading the data in csv format**

```
data= read_csv("data set.csv")
```

3. **Data needs to be formatted**
4. **Data set is split into 60% train and 40% test.**

```
sample = sample.split(data$date, SplitRatio =0.6)
 train = subset(data, sample == TRUE)
 test  = subset(data, sample == FALSE)
```

5. **Plotting of closing price**



6. **Testing the stationarity of data set, so doing the kpss test of stationarity.**

```
#stationary
kpss.test(data$close)
diffed_close = diff(data$close, differences = 1)
kpss.test(diffed_close)
```

```
        KPSS Test for Level Stationarity

data:  data$close
KPSS Level = 9.5589, Truncation lag parameter = 8, p-value = 0.01
```

7. A lag variable has been created because LSTM requires data in supervised learning. This basically, differences in closing prices and look back =1.

	x-1	DF
1	0.00	0.00
2	0.00	-0.51
3	-0.51	-0.01
4	-0.01	0.94
5	0.94	-0.04
6	-0.04	-0.62
7	-0.62	1.21
8	1.21	-2.49
9	-2.49	-0.70
10	-0.70	-1.25
11	1.25	1.21

8. The order of observation is important for time series data, the supervised closed data is split into 0.6 test and 0.4 train.

```
N_close = nrow(supervised_close)
n_close = round(N_close *0.6, digits = 0)
train_close = supervised_close[1:n_close, ]
test_close  = supervised_close[(n_close+1):N_close,  ]
```

9. As with any neural network model we scale the X input data into activation function range. To normalize the data range, we used the feature range parameter, and selected the default value (0, 1) which is typical for data with low dispersion.
10. The default activation function for LSTM is the sigmoid function, the range of which is (-1, 1)

Name	Type	Value
Scaled_close	list [3]	List of length 3
scaled_train	list [1194 x 2] (S3: data.frame)	A data.frame with 1194 rows and 2 columns
scaled_test	list [796 x 2] (S3: data.frame)	A data.frame with 796 rows and 2 columns
scaler	double [2]	-7.73 7.18

11. Inverted scaling

```
 invert_scaling = function(Scaled, scaler, feature_range = c(0, 1)){
 min = scaler[1]
 max = scaler[2]
 t = length(Scaled)
 mins = feature_range[1]
 maxs = feature_range[2]
 inverted_dfs = numeric(t)

 for( i in 1:t){
  X = (Scaled[i]- mins)/(maxs - mins)
  rawValues = X *(max - min) + min
  inverted_dfs[i] <- rawValues
 }
 return(inverted_dfs)
}
```

12. LSTM Model:

```
#LSTM
class(x_train_close)
#x_train_close <- array(data = x_train_close, dim = c(nrow(x_train_close),1,
look_back))

dim(x_train_close) <- c(length(x_train_close), 1, 1)
head(x_train_close)

X_shape2_close = dim(x_train_close)[2]
X_shape3_close = dim(x_train_close)[3]
batch_size = 1
units = 1
```

```
#LSTM
> class(x_train_close)
[1] "numeric"
> dim(x_train_close) <- c(length(x_train_close), 1, 1)
> head(x_train_close)
[1] 0.03688799 0.03688799 -0.03152247 0.03554661 0.16297787 0.03152247
> X_shape2_close = dim(x_train_close)[2]
> X_shape3_close = dim(x_train_close)[3]
> batch_size = 1
> units = 1
> model_close <- keras_model_sequential()
> model_close%>%
+   layer_lstm(units, batch_input_shape = c(batch_size, X_shape2_close,
X_shape3_close), stateful= TRUE)%>%
+   layer_dense(units = 1)
```

13. Network loop: -

The network loop is created which iterates through every window in batch creating the batch states as all zeros. The model is structured to remember its learning at every iteration by defining the stateful as true.

```
model_close <- keras_model_sequential()

model_close%>%

 layer_lstm(units, batch_input_shape = c(batch_size, X_shape2_close,
X_shape3_close), stateful= TRUE)%>%

 layer_dense(units = 1)
```

14. Defining the loss: -

In this the mean square error function is used for the loss to minimize the errors.

```
model_close %>% compile(
 loss = 'mean_squared_error',
 optimizer = optimizer_adam( lr= 0.02, decay = 1e-6 ),
 metrics = c('accuracy')
 )
```

15. The network is trained with 25 number of epochs which we had initialized, and then observe the change in our loss through time. The current loss decreases with the increase in the epochs as observed, increasing our model accuracy in predicting the Litecoin prices.

16. Model Summary:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (1, 1)                    12
_________________________________________________________________
dense (Dense)                (1, 1)                    2
=================================================================
Total params: 14
Trainable params: 14
Non-trainable params: 0
_________________________________________________________________
```

17. 25 iterations is made on train data which is 1194 observations.

```
1194/1194 [==============================] - 4s 3ms/sample - loss: 0.0049 - acc: 8.3752e
-04
1194/1194 [==============================] - 4s 3ms/sample - loss: 0.0051 - acc: 8.3752e
-04
1194/1194 [==============================] - 4s 3ms/sample - loss: 0.0049 - acc: 8.3752e
-04
1194/1194 [==============================] - 4s 3ms/sample - loss: 0.0049 - acc: 8.3752e
-04
1194/1194 [==============================] - 4s 3ms/sample - loss: 0.0049 - acc: 8.3752e
-04
1194/1194 [==============================] - 4s 3ms/sample - loss: 0.0051 - acc: 8.3752e
-04
```

18. Modelling on 796 observations

A. Input =1

```
L_close = length(x_test_close)

scaler_close = Scaled_close$scaler

predictions_close1 = numeric(L_close)

for(i in 1:L_close){

  X_close = x_test_close[i]

  dim(X_close) = c(1,1,1)

  yhat = model_close %>% predict(X_close, batch_size=batch_size)

  # invert scaling

  yhat_close = invert_scaling(yhat, scaler_close,  c(-1, 1))

  # invert differencing

  yhat_close  = yhat_close + data$close[(n_close+i-1)]

  # store

  predictions_close1[i] <- yhat_close

}
```

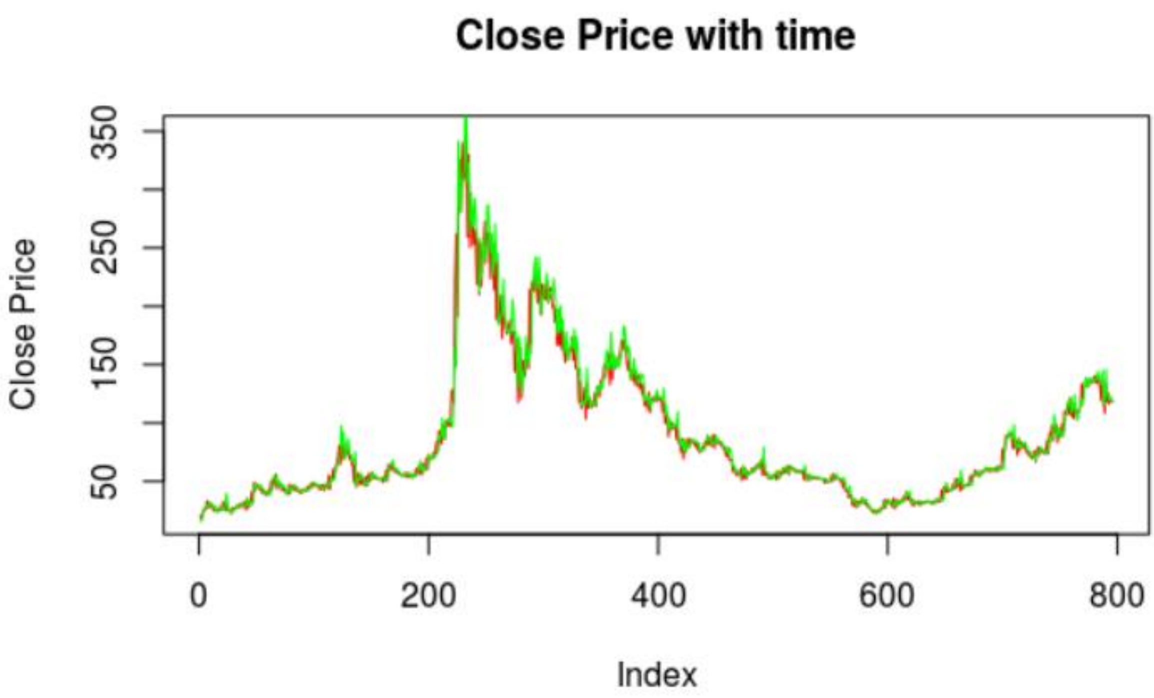
yhat	num [1, 1] 0.0748

B. Input =2

```
L_close = length(x_train_close)
scaler_close = Scaled_close$scaler
predictions_close2 = numeric(L_close)
for(i in 2:L_close){
  X_close = x_train_close[i]
  dim(X_close) = c(1,1,1)
  yhat = model_close %>% predict(X_close, batch_size=batch_size)
  # invert scaling
  yhat_close = invert_scaling(yhat, scaler_close,  c(-1, 1))
  # invert differencing
  yhat_close  = yhat_close + data$close[(i-1)]
  # store
  predictions_close2[i] <- yhat_close
}
```

yhat_close	15.9123097861931

19. Plotting the predictions for all the 1991 observations:



20. Creating the data.final for recording the absolute values which is the difference true value and estimated value as shown in the tabulated figure below.

```
datefinal=seq(from = as.Date("2019-06-07"),to = as.Date("2019-07-07"),by =
"day")
datafinal=data.frame(date=datefinal, true_value=data$close[(1991-30):1991] ,
estimate_value=predictions_close[(1991-30):1991])
datafinal$absol_est=abs(datafinal$true_value-datafinal$estimate_value)
write.table(datafinal, "LSTM.csv", row.names=FALSE, sep=";",dec=".", na=" ")
```

	date	true_value	estimate_value	absol_est
1	2019-06-07	117.08	104.1310	12.9490168
2	2019-06-08	118.51	111.6910	6.8190168
3	2019-06-09	114.87	117.3610	2.4909832
4	2019-06-10	129.83	119.7405	10.0895237
5	2019-06-11	136.08	115.1510	20.9290168
6	2019-06-12	136.16	130.1110	6.0490168
7	2019-06-13	130.86	136.3610	5.5009832
8	2019-06-14	132.71	139.4525	6.7424517
9	2019-06-15	138.35	131.1410	7.2090168
10	2019-06-16	136.95	132.9910	3.9590168
11	2019-06-17	134.19	138.6374	4.4474430
12	2019-06-18	135.13	137.5507	2.4207465
13	2019-06-19	136.83	134.4710	2.3590168
14	2019-06-20	135.78	135.4110	0.3690168
15	2019-06-21	139.07	137.1116	1.9584208
16	2019-06-22	141.77	136.0610	5.7090168
17	2019-06-23	136.83	139.3510	2.5209832
18	2019-06-24	135.40	144.5269	9.1268988
19	2019-06-25	135.51	137.4061	1.8960596
20	2019-06-26	130.52	135.6810	5.1609832