National College of Ireland

Configuration Manual

MSc Research Project
MSc in FinTech

Sagar Anil Tade
Student ID: x18109641

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Sagar Anil Tade
Student ID:	x18109641
Programme:	MSc in FinTech
Year:	2019
Module:	MSc Research Project
Supervisor:	Noel Cosgrave
Submission Due Date:	12/08/2019
Project Title:	Configuration Manual
Word Count:	1000
Page Count:	5

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

Sagar Anil Tade
x18109641

1 Introduction

Configuration manual is mainly used to for reproducing the data and and the facts and checking and achieving the reproducablity of the project. So the title of my thesis is "Can a hybrid of ANN-GARCH model provide a significant improvement in predicting the price volatility of the Ethereum?". All the details required to run this project are provided in this document. Machine configurations and software configurations are also included in this document. Code snippets of the interesting part of the code will also be attached in the following sections.

2 Data Collection

Dataset used for this research is obtained from coinbase [1]. Data is manually scraped from the website. Then this data is stored into excel file with extension .xlsx. This excel file is imported into R code for using. R data object was created of the same. Alternative of obtaining this data is from Quandl. Data chosen was from 07th August 2015 to 27th July 2019. It has 1451 observation and 5 attibutes namely open, close, high, low and date.

2.1 Dataset Metadata

Data collected is in the raw dataframe format. It needs to be converted into ts object for analysis. Date and time format is POSITX and others are numerical columns.

3 System Setup

3.1 Hardware

This research is done on a personal laptop. Its configurations are as follows:

- Laptop Machine:
  Processor: Intel(R) Core (TM) i5 3317U CPU @1.70 GHz Dual core
  8GB RAM
  500GB HDD

[1] Dataset: `ttps://coinmarketcap.com/`

3.2 Software

The research work is implemented on Windows 8.1 Pro edition.

64-bit RStudio version 1.2.1335 is used with R-version 3.5.1 on x86_64-w64-mingw32/x64 platform to complete the data preprocessing part.

Anaconda with Spyder editor and Python 3.7 were used for rest of the project.

Office 365 and 2019 Microsoft office suite is used for data recording and maintaining. Notepad++ is used for doing scratch work 7.6.6

4 Software/ Libraries

RStudio along with Python are used for this research project. Following are list of packages used in R and Python respectively. **R:**
Quandl - For data fecting (Optional Package)
readxl For data fetching.
TTR-0.23-4 - for volatility estimation.
Quantmod - for mathematical equations.
Zoo- 1.8 - for timeseries.
Smooth - for forecasting
MComp- for Statiscal analysis
tseries - for timeseries
ggplot and plotly- for plotting graphs and visualization.
xts 0.11 - for timeseries

Python:
panda 2.2.4 - for data fetching
numpy 2.2.4 - for data handeling
matplotlib.pyplot 2.2.4 - for visualization
warning 2.2.4 - ignoring the warnings
matplotlib.dates 2.2.4 - for converting dates.
Arch 2.2.4- for GARCH Modelling
Keras 2.2.4 Libraries and Packages: Sequential, Dense, LSTM, Droupout for implementation of LSTM, saving and loading model.

Overleaf for creating report.

5 R Code Files

Following is the snippet of the R code which is used for calculating the volatility with yang zang estimator.

```
library("caTools")
c <- xts(x=pct_change, order.by = eth$Date)
b = roll_sd(c, 30, center = F)
d <- data.frame(date=index(b), coredata(b))
stdev <- d$coredata.b
eth <- cbind(eth, stdev)
```

```
#Rolling Volatility With 30 Time Periods
Volatility <- volatility(eth[-1], n = 30, calc = "yang.zhang", N = 365)
plot(Volatility, type = 'l')

eth <- cbind(eth, Volatility) #Appending volatility to original dataframe
plot_ly(x = eth$index, y = Volatility, type = 'scatter', mode = 'lines') %>%
         layout(title = "Rolling Volatility With 30 Time Periods - Yang Zang
             Estimator",
                xaxis = list(title = "Index"),
                yaxis = list (title = "Volatility"))



# Check for stationarity using ADF and KPSS tests
adf.test(na.omit(Volatility))
kpss.test(na.omit(Volatility), null="Trend")

# Test for checking ARCH ###
library(fDMA)
archtest(ts=Volatility)

# Exporting data to excel
l <- list(df = eth)
openxlsx::write.xlsx(l, file =
    "D:\\Sagar\\Study\\Sem3\\Research_Thesis\\Research_Project\\Research\\Code\\Dataset\\Upd


### *Please Note: Run the Python.py code after this which is located in the
    zip.####
```

Following is the code from python which is used for creating rolling window forecast of timeseries.

```
from arch import arch_model
am = arch_model(vol, vol='Garch', p=1, o=1, q=1, dist='Normal')


res1 = am.fit()
res1.summary()


df1 = pd.DataFrame(columns=['test', 'I'])

df1['test'] = res1.resid
df1.loc[df1['test'] < 0, 'I'] = 1
df1["I"] = df1["I"].fillna(0)
```

```
df['forecast_vol'] = np.sqrt(res1.params['omega'] + res1.params['alpha[1]'] *
    res1.resid**2 + res1.params['gamma[1]'] * res1.resid**2 * df1['I'] +
    res1.conditional_volatility**2 * res1.params['beta[1]'] ) ## Scaled from
    0.1 to 0.01 when *3 then rmse minimum 0.49788 else 0.619080


def rmse_tr(predictions, targets): return np.sqrt(((predictions - targets) **
    2).mean())
skor = rmse_tr(df.loc[df.index[300:], 'forecast_vol'], df.loc[df.index[300:],
    'Volatility'])
```

In the same way, next snippet is of LSTM

```
# Initialising the RNN
regressor = Sequential()

# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 10, return_sequences = True, input_shape =
    (X_train.shape[1], 1)))
regressor.add(Dropout(0.1))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 10, return_sequences = True))
regressor.add(Dropout(0.1))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 10, return_sequences = True))
regressor.add(Dropout(0.1))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 10))
regressor.add(Dropout(0.1))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)

regressor.summary()


predicted_stock_price = regressor.predict(X_train)


# Visualising the results
plt.figure(figsize=(18,6))
```

```
plt.plot(df.iloc[300:, 10:11].values, color = 'red', label = 'Observed
    Volatility')
plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted Volatility
    By LSTM_GJR-GARCH(1,1)')
plt.title('Real Rolling Volatility vs Forecast of LSTM_GJR-GARCH(1,1)')
plt.xlabel('Time')
plt.ylabel('Volatility')
plt.legend()
plt.show()
```