ROBOTRADE

Robotrade Manual 1.0

Table of contents

1. Purpose 3

2. Audience 3

3. User Guide 4

3.1 Prerequisite 4

3.1.1 System Requirements 4

3.1.2 Software Requirement 5

3.1.3 WAMPserver 5

3.1.4 Python libraries required: 7

3.1.5 Install TA-Lib: 8

3.2 Prediction model 9

3.2.1 Sequential Neural Network 9

3.2.1.1 Training the model (Optional) 9

3.2.1.2 Backtesting the model on Strategy 1 10

3.2.1.3 Backtesting the model on Strategy 2 11

3.2.1.4 Output verification: 12

3.2.2 Random Forest 14

3.3 User Interface 16

3.3.1 Plotly 16

3.3.2 WAMP SERVER 17

3.4.3 PHP 19

3.5 Other model 25

3.5.2 CNN 25

3.5.3 LSTM 27

4.FAQ 29

5. APPENDIX 30

1. Purpose

RoboTrade is an application for helping the trader in making trade-related decisions by providing them with the best buy and sell options. This is done through the utilization of multiple indicators.

Overall the system will allow the user to do the following:-

- Visually represent the movements in the Forex market
- Make Buy/sell decision using market indicators

2. Audience

Robotrade has been developed for 2 users.

System Admin

Trader

Who is the system admin?
The person is solely responsible for the overall functioning and activities of the Robotrade.

Who is the Trader?
The person who is going to view and the backtesting results along with the live chart of currencies on the user interface.

Every end-user has been allocated with certain roles and responsibilities that will help them to understand the functioning segments of the system.

Features of a System Admin:

- Installing the recommended software's
- Installing the python libraries.
- Making the changes in the source code to fit the systems environment and requirements.
- Executing the python files on the anaconda command prompt, Jupiter notebook, or Google Collaboratory.
- Set the database for storing the back-testing results
- Installing and preparing the WampServer.

Features of a Trader:

- Select the currency and frequency on the User Interface to view the back-testing results
- View the backtesting on different currencies using different strategies in the tabular format.
- Select the currency to choose the live data graph.
- Go to plotly to view the 4-hourly data with open, high, low, and close on the 4 hourly data brought by FXCM.

3. User Guide

3.1 Prerequisite

3.1.1 System Requirements

Minimum System Requirement	
RAM	8 GB
HDD	50 GB
Processor	Intel i5 (7th Generation or above).
GPU	NVIDIA GTX 940M or above.

Recommended System Requirement	
RAM	16 GB
SSD	50 GB
Processor	Intel i7 (8th Generation or above).
GPU	NVIDIA GTX 1650 or above.

3.1.2 Software Requirement

- Python 3.6 or above.
- Jupyter Notebook,
- Spyder 4.0.1(python 3.7) or Anaconda command prompt
- Google Chrome for PHP
- (Brackets or Notepad++) Text editing software
- Google Colab (optional)

3.1.3 WAMPserver

Step 1: Open the link to download WAMP server 3.2.0 on your desktop. https://download.cnet.com/WampServer-64-Bit/3000-10248_4-75544590.html

Step 2: Click on the “Download Now” button.

Step 3: Open the .exe file after downloading and accept the system requests to proceed with the process.

Step 4: Select the required sources provided in the screenshot below and agree to further requests, then press “Install”.

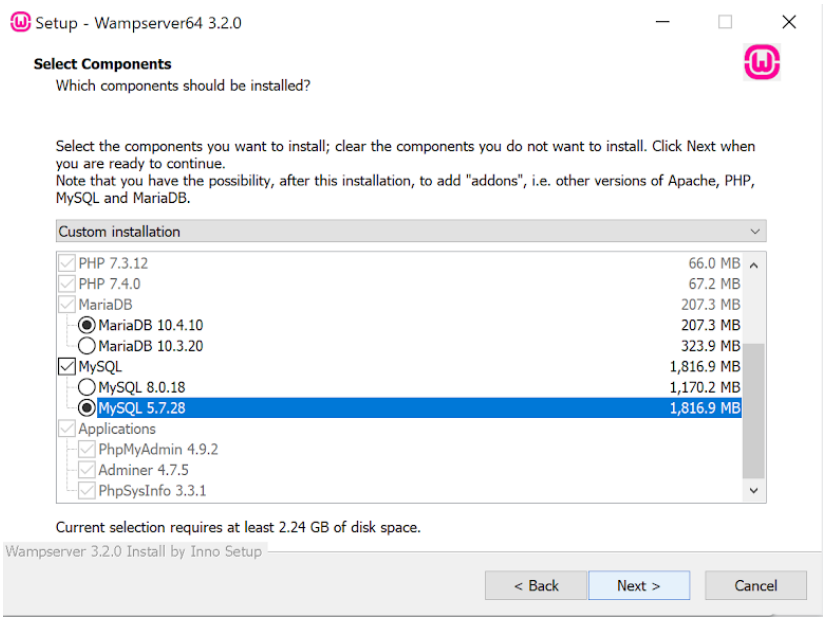


Figure 1: WAMPserver installation

Step 5: Search for “Wampserver64” on the search bar and run the application.

Step 6: Check whether all the services are running in the windows notification area, the way it’s shown in the screenshot below.

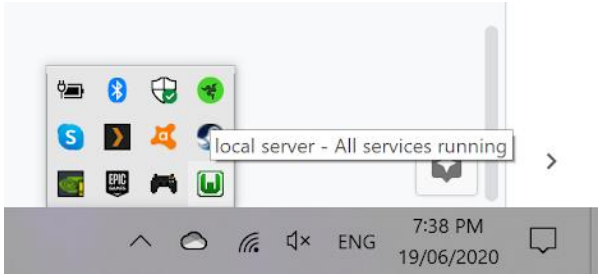


Figure 2: Check whether the server is turned on

Step 7: Search for “localhost” in the search bar on the new tab of your default browser to see whether the Wampserver is working on your computer. You will get the following result:

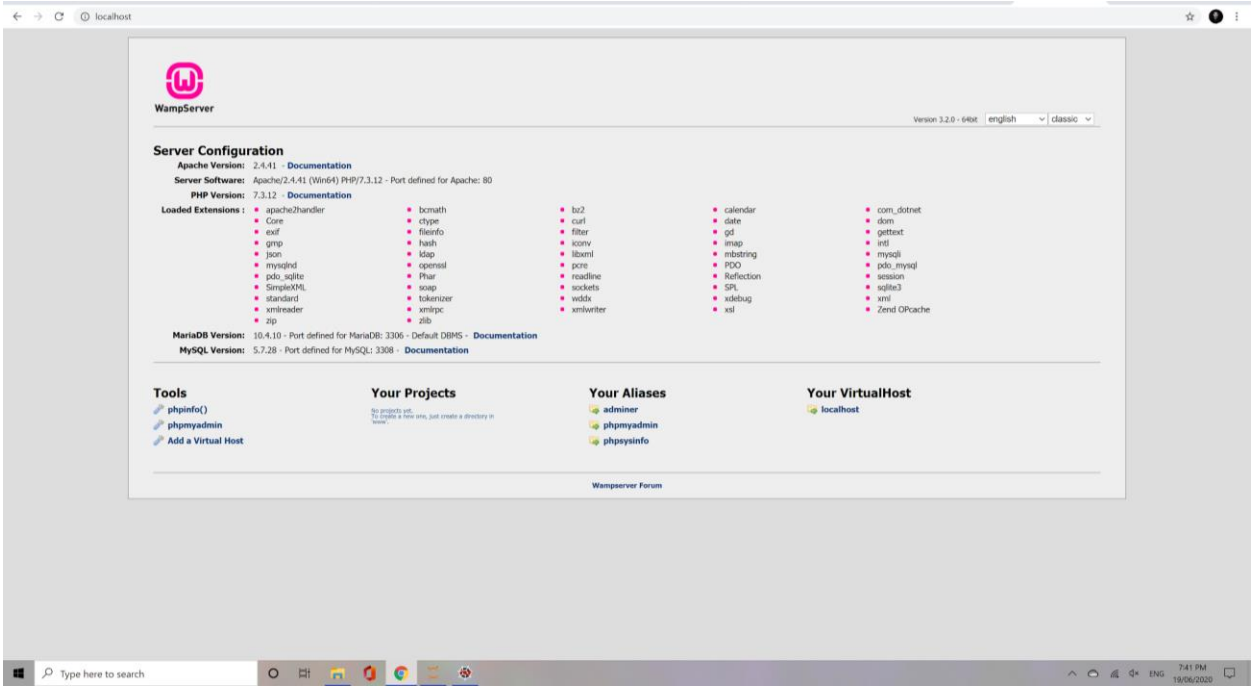


Figure 3: Webpage displaying WAMPserver on localhost

3.1.4 Python libraries required:

Step 1: To install the packages, open Administrator Command Prompt and run the following command-
cd "C:\<dir>\Robotrade"

Step 2: After redirecting CMD to the folder, the following code will install the prerequisite packages in the computer-
pip install -r requirements.txt

The packages below will be installed:

- spyder-kernels==0.5.2
- websocket-client==0.57.0
- socketIO-client==0.7.2
- python-socketio==4.6.0
- astor==0.8.1
- bokeh==1.4.0
- keras-applications==1.0.8
- chart-studio==1.1.0
- Click==7.0
- fxcmpy==1.2.6
- ipykernel==5.3.0
- ipython==5.5.0
- ipython-genutils==0.2.0
- ipywidgets==7.5.1
- Keras-Preprocessing==1.1.2
- Keras==2.4.2
- matplotlib==3.1.3
- mysql==0.0.2
- mysql-connector==2.2.9
- mysqlclient==1.4.6
- numpy==1.18.1
- pandas==0.24.2
- PyMySQL==0.9.3
- plotly==4.5.4
- scikit-image==0.15.0
- scikit-learn==0.21.3
- scikit-plot==0.3.7
- scipy==1.4.1

- tensorboard==2.2.2
- tensorboard-plugin-wit==1.6.0.post3
- tensorflow==2.2.0
- tensorflow-estimator==2.2.0
- tqdm==4.36.1
- collections-extended==1.0.3
- seaborn==0.9.0
- pickleshare==0.7.5
- DateTime==4.3
- jupyter==1.0.0
- jupyter-client==5.3.4
- jupyter-console==6.1.0
- jupyter-core==4.6.1
- jupyterlab==1.2.6
- jupyterlab-server==1.0.6

3.1.5 Install TA-Lib:

Step 1: open https://www.lfd.uci.edu/~gohlke/pythonlibs/#ta-lib and download the .whl file

Step 2: Based on your python version and system architecture download the .whl file In our case, the python version is 3.7.2. and 64-bit. Thus, we downloaded TatSu-4.4.0-cp37-cp37m-win_amd64.whl.

Step 3: Open your command prompt and direct it to the folder where the download file is placed.

Step 4: Run **"pip install TatSu-4.4.0-cp37-cp37m-win_amd64.whl "**

3.2 Prediction model

3.2.1 Sequential Neural Network

3.2.1.1 Training the model (Optional)

Step 1: Open the Anaconda command prompt.

Step 2: Move to the specific folder (\Keras_NN_with_softmax_relu) containing the file SNN_train.py

Step 3: Run the following command on the prompt window-
python SNN_train.py AUD_USD_H4_for_keras.csv train H4

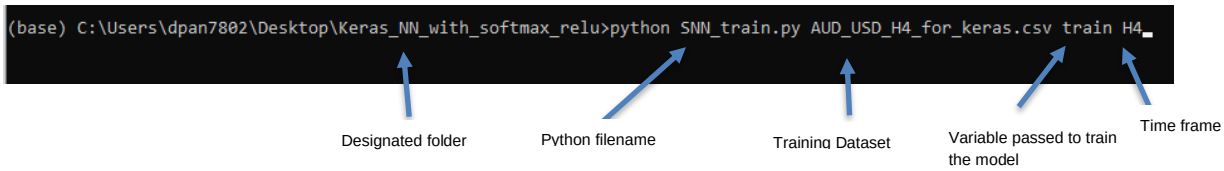


Figure 4: Displaying the command to be run on Anaconda prompt

This will produce the .h5 training model trained on Sequential Neural Network.
We recommend using AUD_USD_H4_for_keras.csv to train a model for 4 Hourly data, and EUR_USD_H1_for_keras.csv to train the model for 1 Hourly data.

Step 4: For reference, Keras_NN.ipynb notebook is attached.

3.2.1.2 Backtesting the model on Strategy 1

Strategy 1- This strategy is backtesting on the live from FXCM which was predicted using Sequential Neural Network **using row by row storage implementation.**

Step 1: Open the Anaconda command prompt.

Step 2: Move to the specific folder (\Keras_NN_with_softmax_relu) containing the file Keras_NN_live_backtest_row_by_row_input_php.py

Step 3: Run the following command on the prompt window-

python Keras_NN_live_backtest_row_by_row_input_php.py run_for all 200 dbname: final1 username: root password:

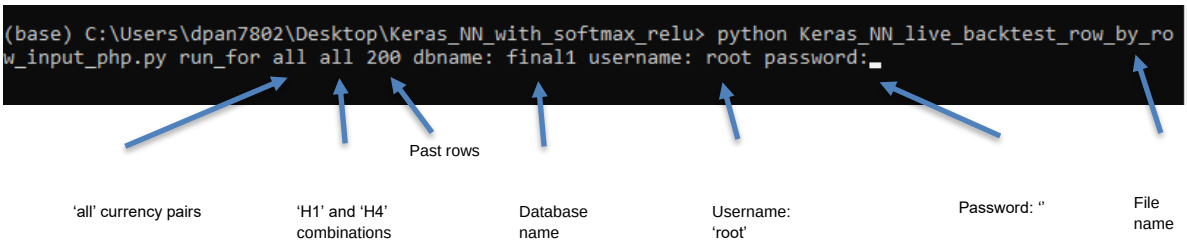


Figure 5: Displaying the command to be run on Anaconda prompt

This code will keep running and update the MySQL as well as PHP webpage every hour for 4 currency pairs included in the model. This will be called the data from currency pairs from the FXCM server for both 1 Hour and 4 Hour duration. The data will be passed through the model provided row by row and the prediction will be implemented for the next trade.

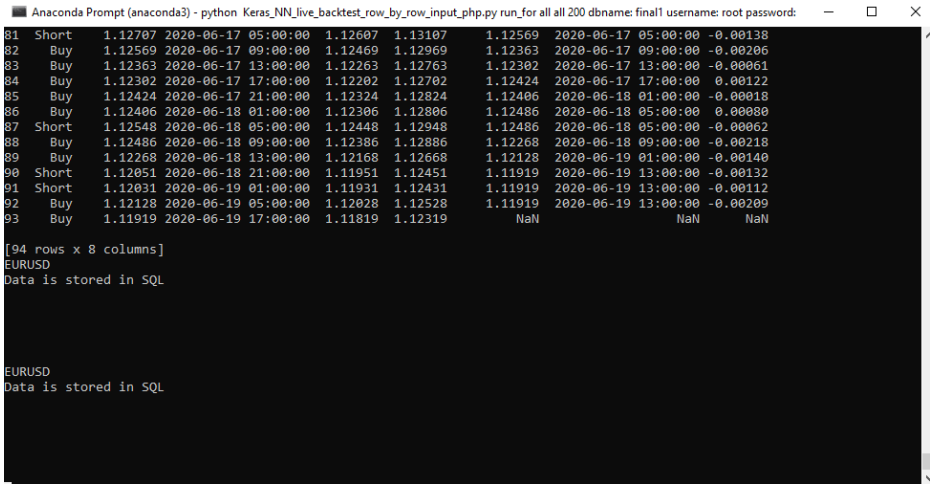


Figure 6: Displaying the output of the code above

Step 4: For reference purpose, the user can test the working of backtesting running notebook file Keras_NN_livedata_backtesting_row_by_row_1 (1).ipynb

3.2.1.3 Backtesting the model on Strategy 2

Strategy 2- This strategy is backtesting on the live from FXCM which was predicted using Sequential Neural Network **using data frame implementation.**

Step 1: Open the Anaconda command prompt.

Step 2: Move to the specific folder (\Keras_NN_with_softmax_relu) containing the file Keras_NN_live_backtest_strategy.py

Step 3: Run the following command on the prompt window-

python Keras_NN_live_backtest_strategy.py run_for all 200 dbname: final1 username: root password:

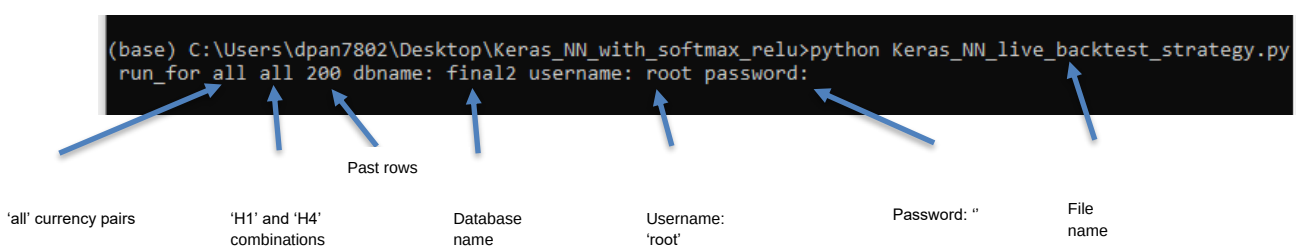


Figure 7: Displaying the command to be run on Anaconda prompt

This code will keep running and update the MySQL as well as PHP webpage every hour for 4 currency pairs included in the model. This will be called the data from currency pairs from the FXCM server for both 1 Hour and 4 Hours duration. The data will be passed through the model as a data frame assuming the stop loss and take profit to be implemented in the particular millisecond second as the data may get aggregated in the future instead of get_candle.

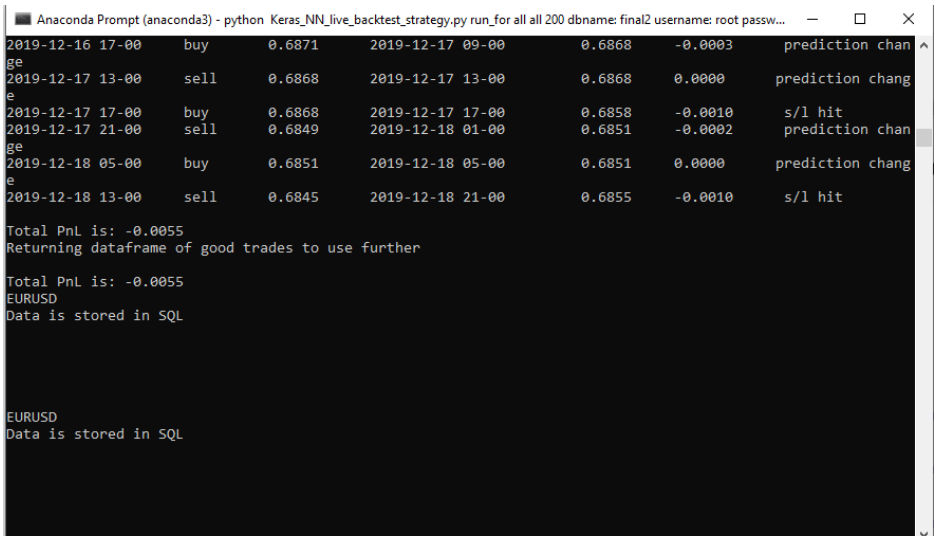


Figure 8: Displaying the output of the code above

Step 4: For reference purpose, user can test the working of backtesting running notebook file Keras_NN_live_backtest_strategy_notebook.ipynb

3.2.1.4 Output verification:

The notebook file **\Keras_NN_with_softmax_relu\Keras_NN_live_backtest_strategy_notebook.ipynb** as well as **\Keras_NN_with_softmax_relu\Keras_NN_livedata_backtesting_row_by_row_1 (1).ipynb** will contain the outputs mentioned in the Final report for testing purposes. The testing was run over 1000 rows in the strategy and the output may display the same. However, the current code used in backtesting on the strategies may include the live dataset which can vary with the output as the number of rows displayed in the PHP webpage and saved in the server is

set up to 200 rows. The user can modify the number of rows to be passed through the model and saved in the database by changing the “past rows” value in the command ran in Anaconda prompt mentioned above.

The screenshots mentioned in the appendix will support the statement mentioned above.

3.2.2 Random Forest

3.2.2.1 Random Forest implementation:

Step 1: Open Jupyter Notebook using Windows Search box or through anaconda navigator by typing “Jupyter notebook”.

Step 2: Direct to the specific folder (\Random_forest_strategy) containing the file Random_forest_train_test_strategy_notebook.ipynb

Work out best moving average crossover

In [3]:
```
import numpy as np
import pandas as pd
import os
#path = r"C:\Capstone\Data"
#os.chdir(path)

#load data from CSV file
aud_4h = pd.read_csv("AUD_USD_H4.csv", usecols = [1,2,3,4,5], parse_dates = ['datetime'],
                     dtype={'close':np.float32})
print(aud_4h.shape)
aud_4h.head()
```

```
(24477, 5)
```

Out[3]:

	datetime	open	high	low	close
0	2005-01-02 18:00:00+00:00	0.78230	0.78280	0.78060	0.78060
1	2005-01-02 22:00:00+00:00	0.78070	0.78390	0.78070	0.78155
2	2005-01-03 02:00:00+00:00	0.78145	0.78165	0.77265	0.77535
3	2005-01-03 06:00:00+00:00	0.77535	0.77945	0.77315	0.77835
4	2005-01-03 10:00:00+00:00	0.77835	0.78135	0.77615	0.77775

In [4]:
```
#crate usable date column

from datetime import datetime
aud_4h['date']= aud_4h['datetime'].apply(lambda x: datetime.strftime(x, '%Y-%m-%d %H:%M'))
aud_4h = aud_4h.drop(columns='datetime')
aud_4h = aud_4h[['date','open', 'high', 'low', 'close']]
aud_4h.head()
```

Out[4]:

	date	open	high	low	close
0	2005-01-02 18:00	0.78230	0.78280	0.78060	0.78060
1	2005-01-02 22:00	0.78070	0.78390	0.78070	0.78155
2	2005-01-03 02:00	0.78145	0.78165	0.77265	0.77535

Figure 9: Displaying notebook beginning

Step 3: The notebook is divided into several segments to run step by step accordingly. We recommend the user to follow the given sequence of steps:

- Work out the best moving average crossover
- Add indicators to the dataset
- Train and Test Split
- Training the model
- Load the testing data
- Run the backtest strategy

Step 4: Or you can directly click Cell → Run All.

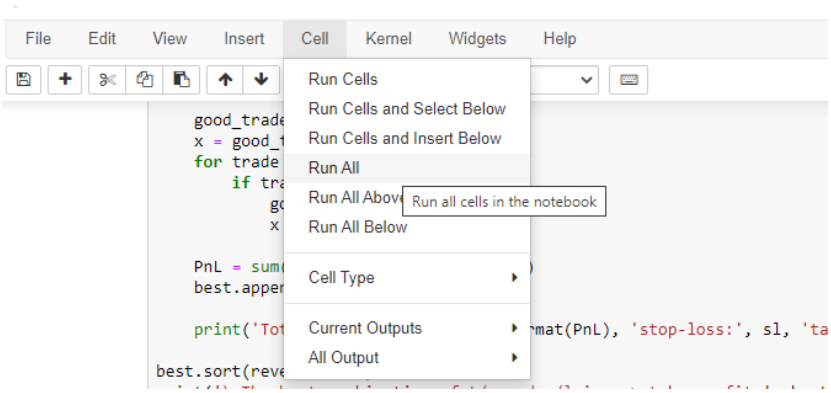


Figure 10: Run All function of jupyter notebook

The code will run all cells and finish in approximately 20-30 minutes, depending on your PC specifications.

3.3 User Interface

3.3.1 Plotly

A Final_graphs.py file has to be executed before viewing the PHP website. This will be an on-going script which will keep fetching data on a 4-hourly basis. The following command needs to be executed in the command prompt or terminal.

Step 1: Open the Anaconda command prompt.

Step 2: Move to the specific folder (\Keras_NN_with_softmax_relu) containing the file Keras_NN_live_backtest_strategy.py

Step 3: Run the following command on the prompt window-
python Final_graph.py

```
(base) C:\Users\dpan7802\Desktop\Keras_NN_with_softmax_relu>python Final_graphs.py
```

Figure 11: Run command on Anaconda prompt

After executing this line, it will take some time to run and the following message will show up.
"Fxcm API has been connected".

```
(base) C:\Users\dpan7802\Desktop\Keras_NN_with_softmax_relu>python Final_graphs.py
Fxcm api is connected!
AUD/USD graph has been updated in PHP!
```

Figure 12: Displaying output of the previous command (should keep running)

(Note that this will not work on Saturdays and Sundays as the market is closed during the weekend.)

As soon as the graphs are updated, it will show a message that the graph has been updated.

This would plot candlestick graphs for several currencies onto Chart-studio. These graphs would be automatically updated on our Php website, Robotrade.

3.3.2 WAMP SERVER

To run the User interface:-
Make sure that the plot code is running in the background.

Step 1:- Check if the database is created and the table is being updated with the backtesting data.
In order to do so, click on the Windows notification area ->click on the wampserver icon.

Click on phpMyAdmin.

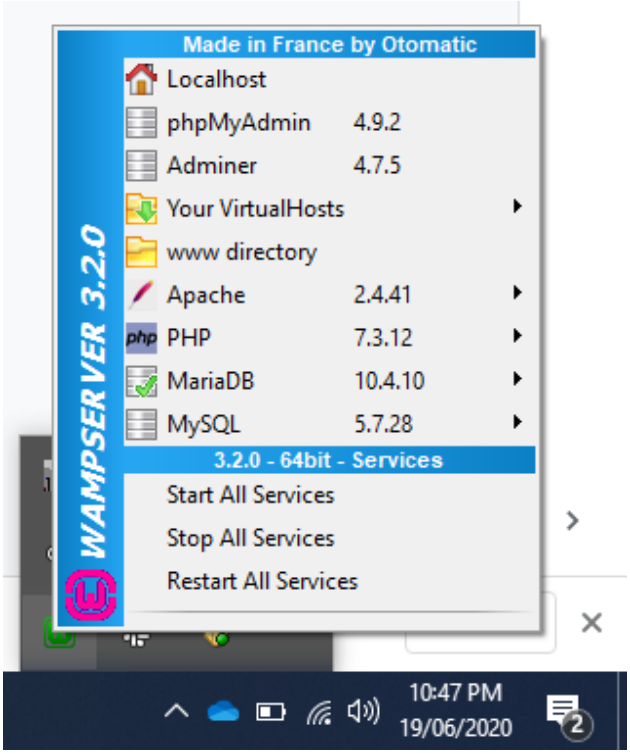


Figure 13: Click Wampserver in notification area

This will open a new tab on the default browser.

Figure 14: PHP MyAdmin login page

Change the username to yours (default: “root”)
Fill in the password(default:””)
Choose the Server Choice as “MySQL”

Step 2: Check if the database is created.

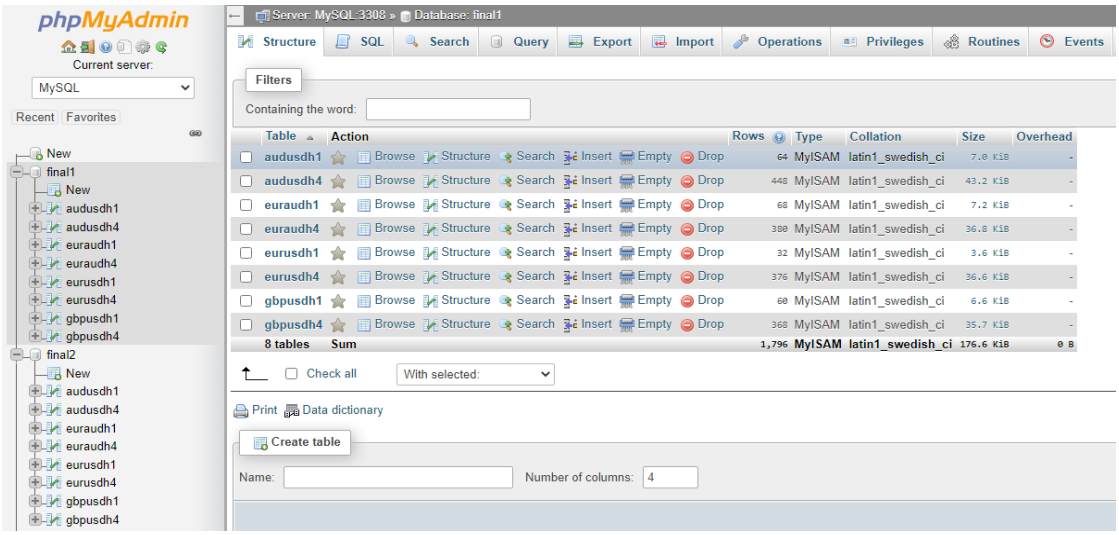


Figure 15: Database creation

Note: if you have followed all the above instructions it should give you two databases'. final1 and final2 along with 8 tables within each database.

3.4.3 PHP

Step 1: Move the PHP website in the designated folder.

Move the folder "tradebot_php" from the extracted folder to the wamp64/www folder (default path: C:\wamp64\www)

Note: "The user interface" has been designed with the following user setting:-

```
$servername = "localhost:3308";
$username = "robotrade";
$password = "robotrade";
```

Please change it as per your own device's port number in all the PHP files i.e. "index.php, strategy1.php and strategy2.php", which can be found here in the windows notification area:

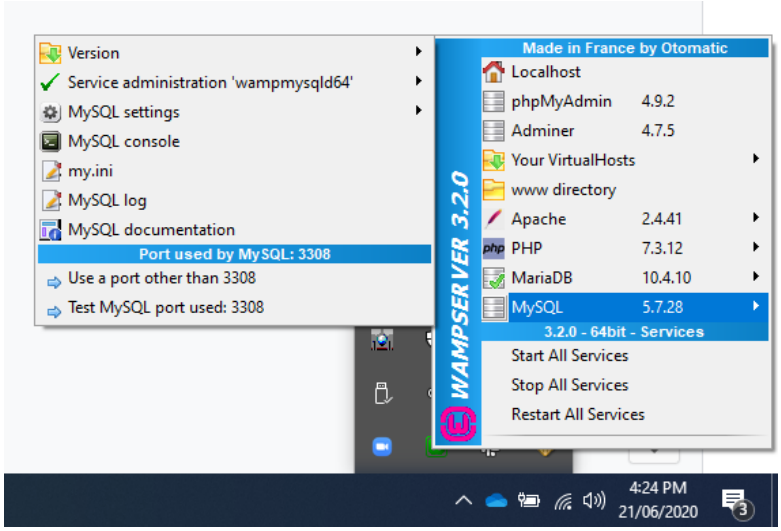


Figure 16: Checking the port number

In our case, the port number is 3308.

Step 2: Running PHP on the browser.

- Open the browser
- Type localhost/tradebot_php

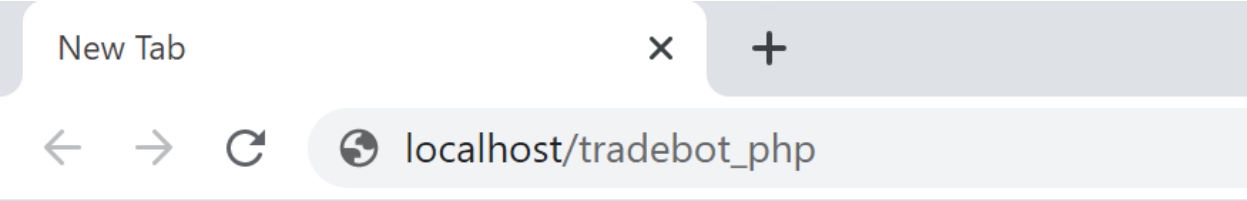


Figure 17: Link to open PHP

This will redirect you to the Homepage of the user Interface (index.php) which looks like this.

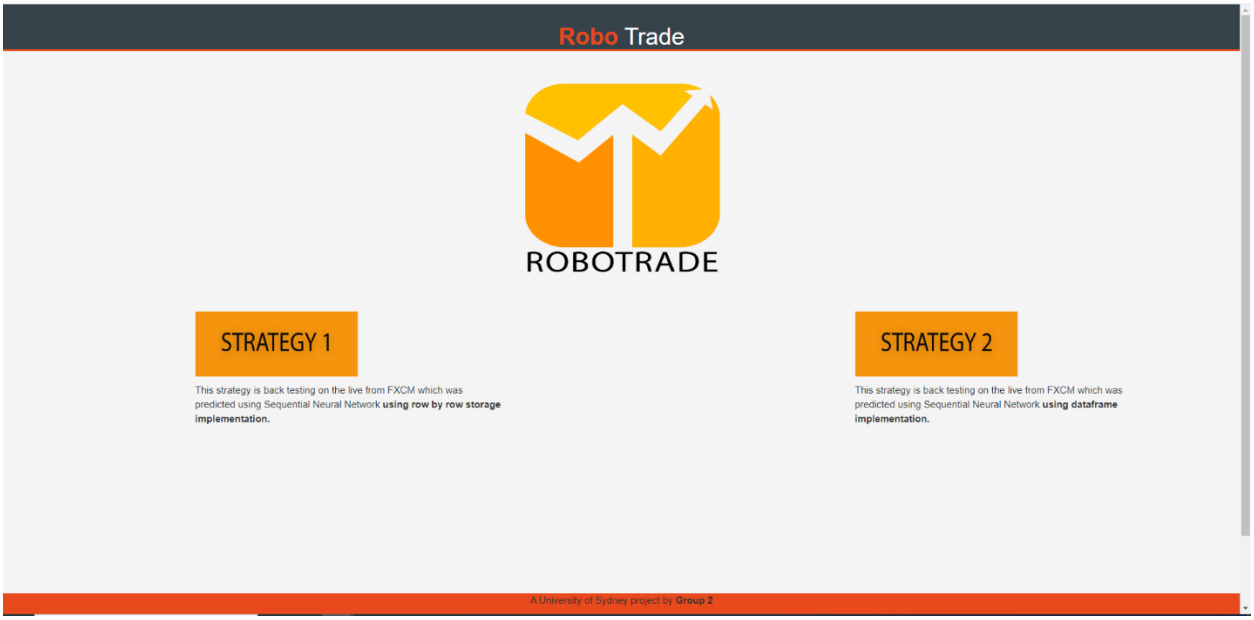
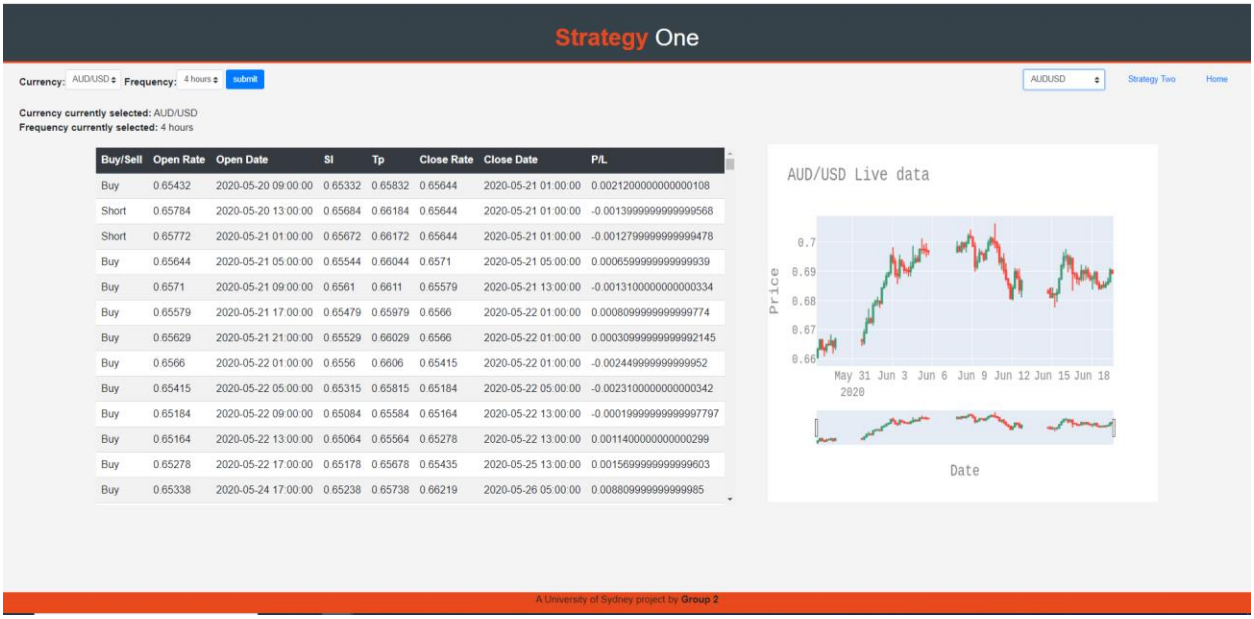


Figure 18: Home page of PHP

Step 3: You can choose to view the strategy of your choice by clicking on the user-item.

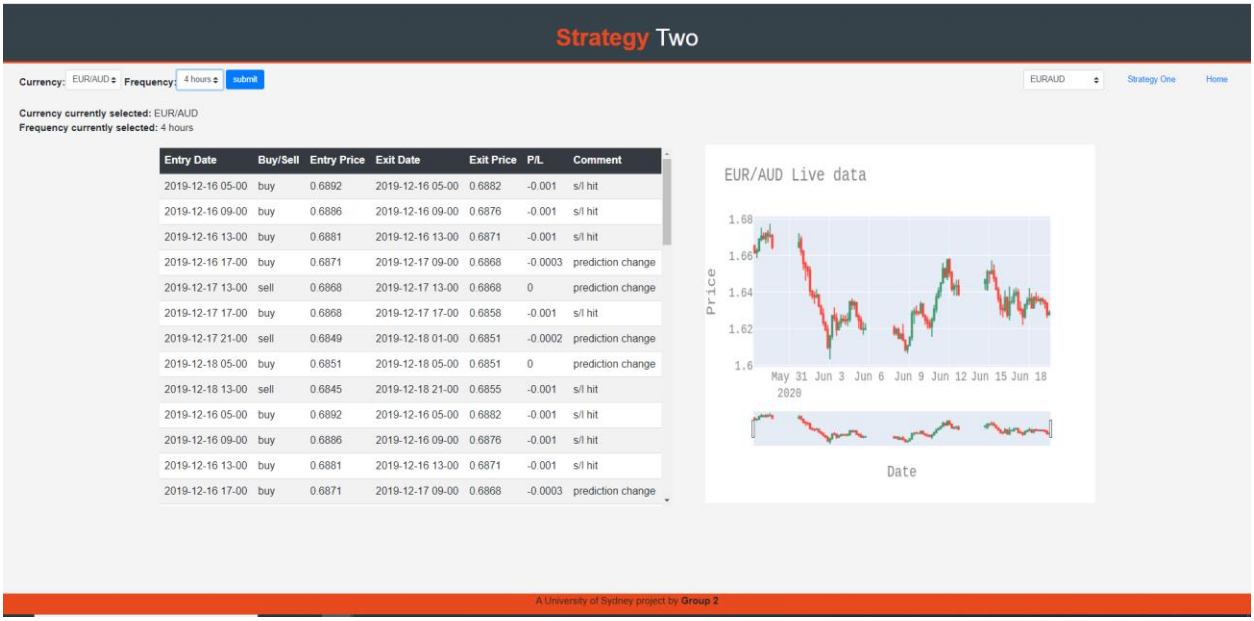
- **Strategy 1**- This strategy is backtesting on the live from FXCM which was predicted using Sequential Neural Network **using row by row storage implementation.**



Buy/Sell	Open Rate	Open Date	Sl	Tp	Close Rate	Close Date	P/L
Buy	0.65432	2020-05-20 09:00:00	0.65332	0.65832	0.65644	2020-05-21 01:00:00	0.0021200000000000108
Short	0.65784	2020-05-20 13:00:00	0.65684	0.66184	0.65644	2020-05-21 01:00:00	-0.0013999999999999568
Short	0.65772	2020-05-21 01:00:00	0.65672	0.66172	0.65644	2020-05-21 01:00:00	-0.0012799999999999478
Buy	0.65644	2020-05-21 05:00:00	0.65544	0.66044	0.6571	2020-05-21 05:00:00	0.0006599999999999939
Buy	0.6571	2020-05-21 09:00:00	0.6561	0.6611	0.65579	2020-05-21 13:00:00	-0.0013100000000000334
Buy	0.65579	2020-05-21 17:00:00	0.65479	0.65979	0.6566	2020-05-22 01:00:00	0.0008099999999999774
Buy	0.65629	2020-05-21 21:00:00	0.65529	0.66029	0.6566	2020-05-22 01:00:00	0.00030999999999992145
Buy	0.6566	2020-05-22 01:00:00	0.6556	0.6606	0.65415	2020-05-22 01:00:00	-0.0024499999999999952
Buy	0.65415	2020-05-22 05:00:00	0.65315	0.65815	0.65184	2020-05-22 05:00:00	-0.0023100000000000342
Buy	0.65184	2020-05-22 09:00:00	0.65084	0.65584	0.65164	2020-05-22 13:00:00	-0.0001999999999999797
Buy	0.65164	2020-05-22 13:00:00	0.65064	0.65564	0.65278	2020-05-22 13:00:00	0.0011400000000000299
Buy	0.65278	2020-05-22 17:00:00	0.65178	0.65678	0.65435	2020-05-25 13:00:00	0.0015699999999999603
Buy	0.65338	2020-05-24 17:00:00	0.65238	0.65738	0.66219	2020-05-26 05:00:00	0.008809999999999985

Figure 19: Page displaying strategy 1

- **Strategy 2**- This strategy is backtesting on the live from FXCM which was predicted using Sequential Neural Network **using data frame implementation.**



Entry Date	Buy/Sell	Entry Price	Exit Date	Exit Price	P/L	Comment
2019-12-16 05-00	buy	0.6892	2019-12-16 05-00	0.6882	-0.001	s/l hit
2019-12-16 09-00	buy	0.6886	2019-12-16 09-00	0.6876	-0.001	s/l hit
2019-12-16 13-00	buy	0.6881	2019-12-16 13-00	0.6871	-0.001	s/l hit
2019-12-16 17-00	buy	0.6871	2019-12-17 09-00	0.6868	-0.0003	prediction change
2019-12-17 13-00	sell	0.6868	2019-12-17 13-00	0.6868	0	prediction change
2019-12-17 17-00	buy	0.6868	2019-12-17 17-00	0.6858	-0.001	s/l hit
2019-12-17 21-00	sell	0.6849	2019-12-18 01-00	0.6851	-0.0002	prediction change
2019-12-18 05-00	buy	0.6851	2019-12-18 05-00	0.6851	0	prediction change
2019-12-18 13-00	sell	0.6845	2019-12-18 21-00	0.6855	-0.001	s/l hit
2019-12-16 05-00	buy	0.6892	2019-12-16 05-00	0.6882	-0.001	s/l hit
2019-12-16 09-00	buy	0.6886	2019-12-16 09-00	0.6876	-0.001	s/l hit
2019-12-16 13-00	buy	0.6881	2019-12-16 13-00	0.6871	-0.001	s/l hit
2019-12-16 17-00	buy	0.6871	2019-12-17 09-00	0.6868	-0.0003	prediction change

Figure 20: Page displaying strategy 2

Step 4: Moving through the User interface.

- On opening the strategy1/ strategy 2 page you'll receive an alert box

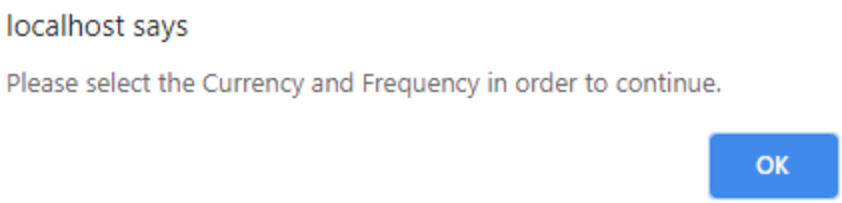


Figure 21: On load display function

- Press 'OK' to continue.
- Select the currency and frequency from the dropdown menu.

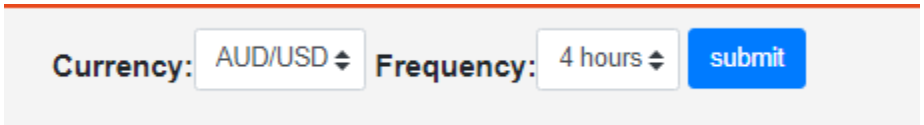


Figure 22: Selection of currency and frequency from the drop-down

- Click on "Submit" Button
- This will show the table generated from backtesting.

Short	0.68929	2020-06-16 17:00:00	0.68829	0.69329	0.69017	2020-06-17 05:00:00	0.0008799999999999919
Short	0.68877	2020-06-16 21:00:00	0.68777	0.69277	0.69017	2020-06-17 05:00:00	0.0013999999999999568
Short	0.68744	2020-06-17 01:00:00	0.68644	0.69144	0.69017	2020-06-17 05:00:00	0.002729999999999899
Short	0.68726	2020-06-17 05:00:00	0.68626	0.69126	0.69017	2020-06-17 05:00:00	0.002909999999999968
Buy	0.69017	2020-06-17 09:00:00	0.68917	0.69417	0.69037	2020-06-17 13:00:00	0.000200000000000089
Short	0.68926	2020-06-17 13:00:00	0.68826	0.69326	0.69037	2020-06-17 13:00:00	0.0011100000000000554
Buy	0.69037	2020-06-17 17:00:00	0.68937	0.69437	0.68772	2020-06-18 01:00:00	-0.002650000000000041
Short	0.68838	2020-06-17 21:00:00	0.68738	0.69238	0.68772	2020-06-18 01:00:00	-0.0006599999999999939
Buy	0.68772	2020-06-18 05:00:00	0.68672	0.69172	0.68949	2020-06-18 05:00:00	0.0017700000000000493
Buy	0.68949	2020-06-18 09:00:00	0.68849	0.69349	0.6847	2020-06-18 21:00:00	-0.004790000000000072
Short	0.68577	2020-06-18 13:00:00	0.68477	0.68977	0.6847	2020-06-18 21:00:00	-0.0010700000000000154
Short	0.68418	2020-06-18 17:00:00	0.68318	0.68818	0.68556	2020-06-19 01:00:00	0.0013799999999999368
Buy	0.6847	2020-06-19 01:00:00	0.6837	0.6887	0.68635	2020-06-19 05:00:00	0.0016500000000000403
Buy	0.68556	2020-06-19 05:00:00	0.68456	0.68956	0.68635	2020-06-19 09:00:00	0.0007900000000000684

Figure 23: Table generated

- To view the hourly live data click on the “Choose currency” dropdown menu.

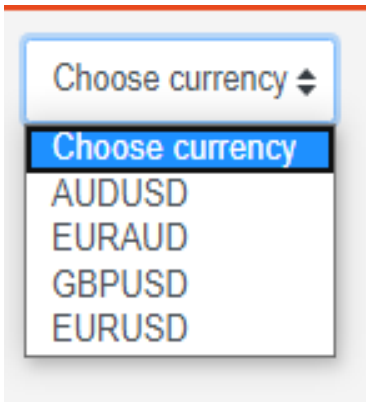


Figure 24: Dropdown for graph

- This will show you the graph similar to this.

Figure 25: Graph generated by plotly

- Click on the graph to go to the live data graph of that currency plotted using.

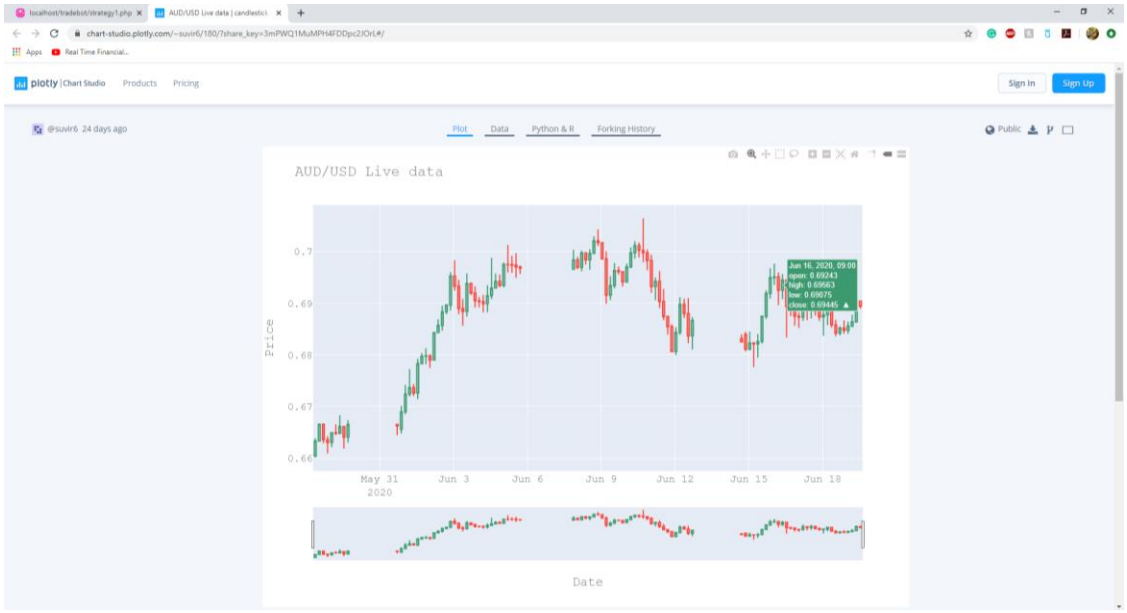


Figure 26: Plotly page link to the graph

- You can use the range selector/slider as per your need.

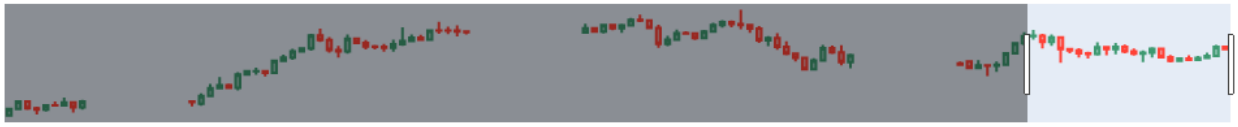


Figure 27: Slider for plotly graph

- In order to switch between different strategy or go to the homepage choose the following options:-

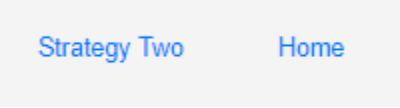


Figure 28: Page linking buttons

3.5 Other model

3.5.2 CNN

3.5.2.1 Training the model (Optional)

Step 1: Open the Anaconda command prompt.

Step 2: Move to the specific folder (\CNN_7x7_image_processing) containing the file SNN_train.py

Step 3: Run the following command on the prompt window-

python CNN_train_test.py AUD_USD_H4.csv train

```
(base) C:\Users\dpan7802\Desktop\CNN_7x7_image_processing>python CNN_train_test.py AUD_USD_H4.csv train
```

Figure 29: Code to run on Anaconda command prompt

This will produce the CNN_weighted_model.h5 training model trained on Convolutional Neural Network.
We recommend using AUD_USD_H4.csv to train the model for 4 Hourly data only.
This model may display the sample array charts created while training the model.

Step 4: For reference, CNN_trial_2.ipynb notebook is attached.

3.5.2.1 Backtesting the model on Strategy 1 <name strategy>

Step 1: Open the Anaconda command prompt.

Step 2: Move to the specific folder (\CNN_7x7_image_processing) containing the file Keras_NN_live_backtest_row_by_row_input_php.py

Step 3: Run the following command on the prompt window-

python CNN_live_backtest_row_by_row_input.py run_for AUD/USD H4 200

```
(base) C:\Users\dpan7802\Desktop\CNN_7x7_image_processing>python CNN_live_backtest_row_by_row_input.py run_for AUD/USD H4 200
```

Figure 30: Code to run on Anaconda command prompt

This code will be running the backtesting strategy of row by row data insertion implementation for fair prediction in backtesting using CNN Conv2D model. This will be calling the data from currency pairs from the FXCM server for 4 Hour duration. The data will be passed through the model provided row by row and the prediction will be implemented for the next trade.

The user can input the currency pair as well as the number of past rows he wants to backtest through the model.

```
tay 195
tay 196
tay 197
tay 198
tay 199
otal amount of trade made:  17 , and 1 trades not closed
here are 9 winning trades and 7 lossing trades. Based on current closed trades, the P/L is :  259.3999999999986
   size  open_rate            open_date       sl       tp  close_rate           close_date      P/L
     1    0.65453  2020-05-20 05:00:00  0.65253  0.65853     0.66354  2020-05-28 17:00:00  0.00901
     0    0.65660  2020-05-22 01:00:00  0.65460  0.66060     0.66354  2020-05-28 17:00:00  0.00694
     1    0.66354  2020-05-28 21:00:00  0.66154  0.66754     0.68893  2020-06-03 05:00:00  0.02539
     1    0.68893  2020-06-03 09:00:00  0.68693  0.69293     0.68836  2020-06-03 09:00:00 -0.00057
     1    0.68836  2020-06-03 13:00:00  0.68636  0.69236     0.69492  2020-06-15 21:00:00  0.00656
     0    0.69291  2020-06-04 17:00:00  0.69091  0.69691     0.69492  2020-06-15 21:00:00  0.00201
     0    0.69421  2020-06-04 21:00:00  0.69221  0.69821     0.69492  2020-06-15 21:00:00  0.00071
     0    0.69337  2020-06-05 01:00:00  0.69137  0.69737     0.69492  2020-06-15 21:00:00  0.00155
     0    0.69744  2020-06-05 05:00:00  0.69544  0.70144     0.69492  2020-06-15 21:00:00 -0.00252
     0    0.69743  2020-06-05 09:00:00  0.69543  0.70143     0.69492  2020-06-15 21:00:00 -0.00251
     0    0.69833  2020-06-08 01:00:00  0.69633  0.70233     0.69492  2020-06-15 21:00:00 -0.00341
     0    0.69971  2020-06-08 09:00:00  0.69771  0.70371     0.69492  2020-06-15 21:00:00 -0.00479
     0    0.70203  2020-06-08 21:00:00  0.70003  0.70603     0.69492  2020-06-15 21:00:00 -0.00711
     0    0.70110  2020-06-10 13:00:00  0.69910  0.70510     0.69492  2020-06-15 21:00:00 -0.00618
     0    0.69419  2020-06-11 09:00:00  0.69219  0.69819     0.69492  2020-06-15 21:00:00  0.00073
     1    0.69492  2020-06-16 01:00:00  0.69292  0.69892     0.69505  2020-06-16 05:00:00  0.00013
     1    0.69505  2020-06-16 05:00:00  0.69305  0.69905         NaN                  NaN      NaN
ress Ctrl+C to terminate!
```

Figure 31: Output for backtesting code mentioned above

3.5.3 LSTM

3.5.2.1 Training the model (Disclaimer: Based on Tensorflow v1 existing LSTM RNN model)

Note: Prerequisite to run this model on your device you need to have version 1 of tensorflow installed on your device.

Step 1: Open Anaconda command prompt.

Step 2: Reinstall TensorFlow to the previous version by running the following commands-

- **pip uninstall tensorflow**
  This will uninstall the previous version.
- **pip show tensorflow**
  This should not display any version, as the tensorflow is removed.
- **pip install tensorflow==1.5**

Step 3: Move to the specific folder (\"LSTM _RNN") containing the file lstmtrader.py

Step 4: Run the following command on prompt window-

python lstmtrader.py GBP_USD_D.csv train

```
(base) C:\Users\dpan7802\Desktop\LSTM _RNN>python lstmtrader.py 'GBP_USD_D.csv' train
```

Figure 32: Code ran on Anaconda prompt for training

This will produce the weights in graphs and checkpoints folder including the training model and well as TF events graphs trained on Long-Short Term Neural Network model including Recurrent Neural Network technique.
We recommend using GBP_USD_H4.csv to train the model for 4 Hourly data only.

3.5.2.2 Preparing the test data and testing the model

Step 1: Run the following command on prompt window-
python lstmtrader.py GBP_USD_D_test.csv test

```
(base) C:\Users\dpan7802\Desktop\LSTM _RNN>python lstmtrader.py GBP_USD_D_test.csv test
```

Figure 33: Code ran on Anaconda prompt for testing

This will produce the predicted suggestions in a .csv file including Change rate and Position advice, which was then tested in a notebook file backtesting_d.ipynb

Step 2: Refer to the existing backtesting_d.ipynb file.

Step 3: Direct the notebook directory to the folder including backtesting_d.ipynb

Step 4: Run the notebook and the flow will explain the usage of the previously created .csv file from our lstmtrader.py prediction

```
In [51]: Buy_PnL = sum(x[6] for x in good_buy_trades)
         Sell_PnL = sum(x[6] for x in good_sell_trades)
         trades = sorted((good_buy_trades + good_sell_trades), key=lambda x: x[0])

         print('Total PnL is:', "{0:.4f}".format(Buy_PnL + Sell_PnL))

         Total PnL is: 0.1018

In [66]: print('entry date\t', '\t\tb/s\t', 'entry price\t', '\texit date\t', '\texit price', '  PnL    ', 'comment','\n')
         for trade in trades:
             print(trade[1],' ',trade[2],'   ","{:.4f}".format(trade[3]),'     ',trade[4],'  ","{:.4f}".format(trade[5]),'  ","{:.4f}".fo
```

```
entry date                    b/s      entry price          exit date                exit price   PnL      comment

2017-02-14T22:00:00.000000000Z   buy      1.2470       2017-02-14T22:00:00.000000000Z   1.2384    -0.0050    s/l hit
2017-02-16T22:00:00.000000000Z   buy      1.2488       2017-02-16T22:00:00.000000000Z   1.2388    -0.0050    s/l hit
2017-02-19T22:00:00.000000000Z   buy      1.2408       2017-02-19T22:00:00.000000000Z   1.2483    0.0050    t/p hit
2017-02-20T22:00:00.000000000Z   sell     1.2463       2017-02-20T22:00:00.000000000Z   1.2401    0.0050    t/p hit
2017-02-21T22:00:00.000000000Z   buy      1.2472       2017-02-21T22:00:00.000000000Z   1.2420    -0.0050    s/l hit
2017-02-26T22:00:00.000000000Z   buy      1.2473       2017-02-26T22:00:00.000000000Z   1.2383    -0.0050    s/l hit
2017-02-27T22:00:00.000000000Z   buy      1.2440       2017-02-27T22:00:00.000000000Z   1.2374    -0.0050    s/l hit
2017-02-28T22:00:00.000000000Z   sell     1.2378       2017-02-28T22:00:00.000000000Z   1.2280    0.0050    t/p hit
2017-03-01T22:00:00.000000000Z   sell     1.2294       2017-03-01T22:00:00.000000000Z   1.2242    0.0050    t/p hit
2017-03-06T22:00:00.000000000Z   buy      1.2238       2017-03-06T22:00:00.000000000Z   1.2169    -0.0050    s/l hit
2017-03-07T22:00:00.000000000Z   buy      1.2199       2017-03-07T22:00:00.000000000Z   1.2139    -0.0050    s/l hit
2017-03-13T21:00:00.000000000Z   buy      1.2220       2017-03-13T21:00:00.000000000Z   1.2109    -0.0050    s/l hit
2017-03-14T21:00:00.000000000Z   buy      1.2150       2017-03-14T21:00:00.000000000Z   1.2310    0.0050    t/p hit
2017-03-15T21:00:00.000000000Z   buy      1.2292       2017-03-15T21:00:00.000000000Z   1.2377    0.0050    t/p hit
2017-03-16T21:00:00.000000000Z   buy      1.2359       2017-03-16T21:00:00.000000000Z   1.2393    0.0035    prediction chang
e
2017-03-19T21:00:00.000000000Z   sell     1.2386       2017-03-19T21:00:00.000000000Z   1.2335    0.0050    t/p hit
```

Figure 34: Notebook output reference for the model

4.FAQ

- IF you get the following issue:-

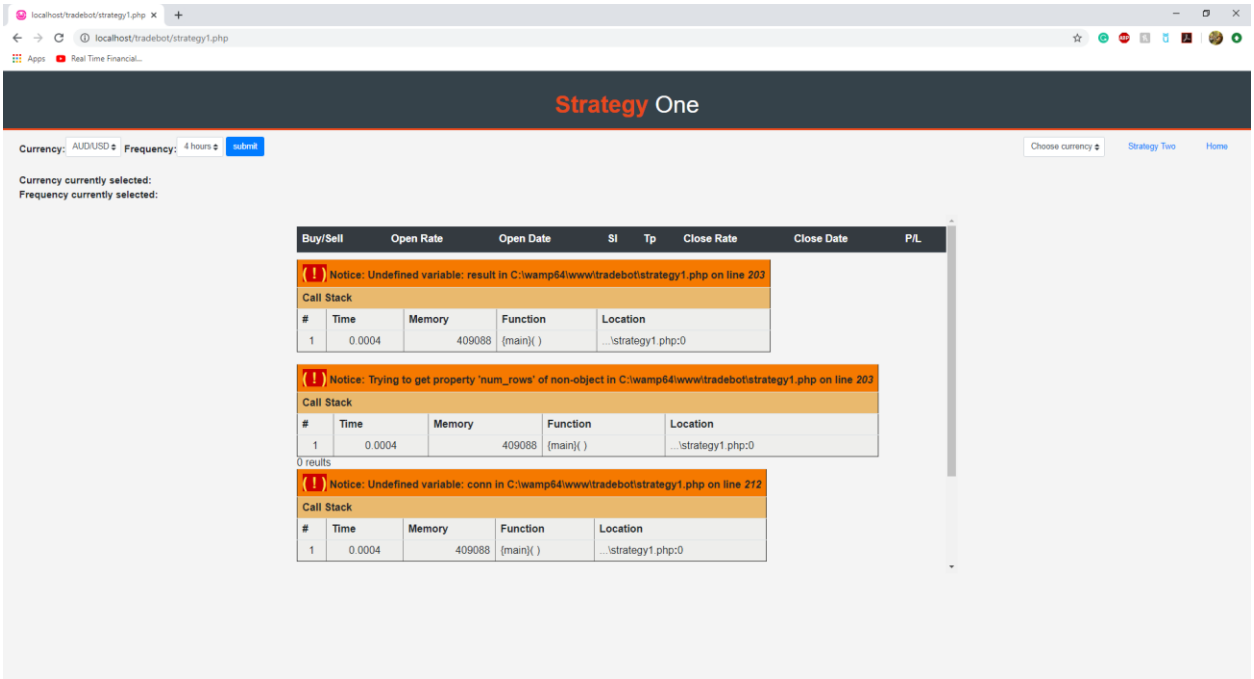


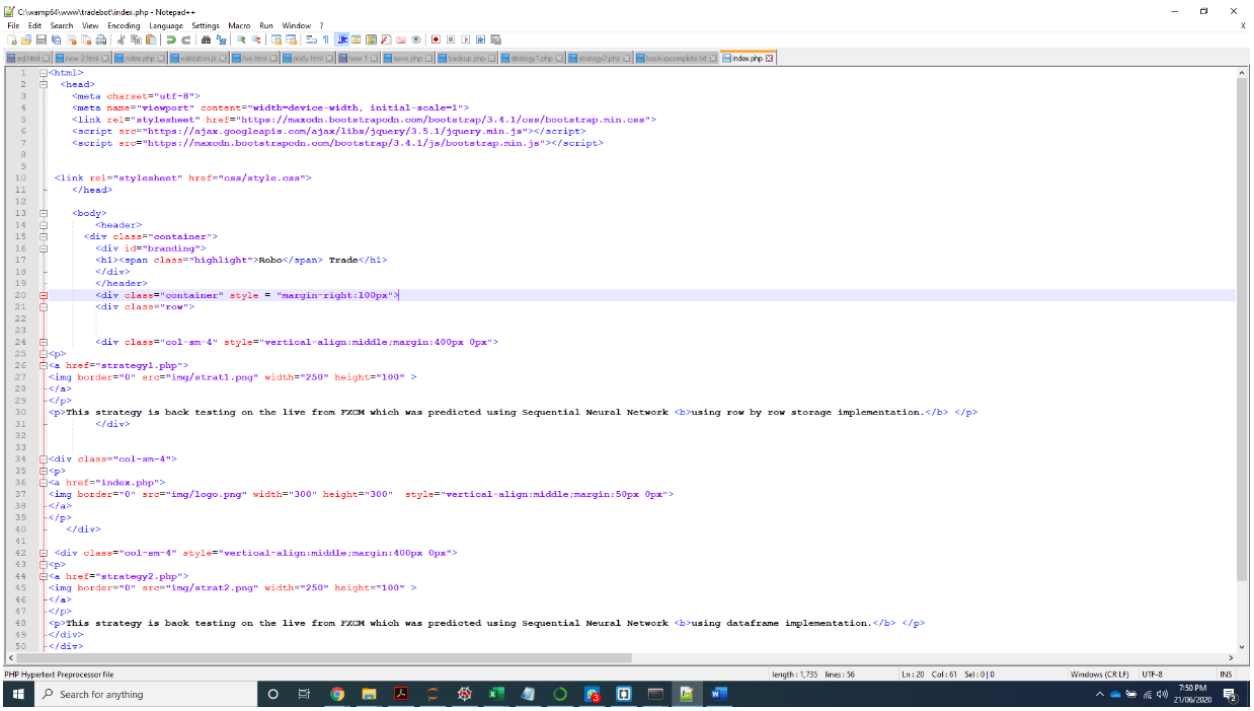
Figure 35: Error message generated on newly loaded page

Please select the currency and frequency and submit it. It should resolve the issue.
If the error still persists then there is an issue with the connectivity with the table

- The requirement to run TensorFlow must be satisfied and if the user has the GPU, he must contain the suitable drivers to support the installation and working of the TensorFlow library in python.
- Running the requirements.txt may including some errors which can be solved by installing the corresponding packages separately as the versions of packages and libraries may vary in the user's personal computer.

5. APPENDIX

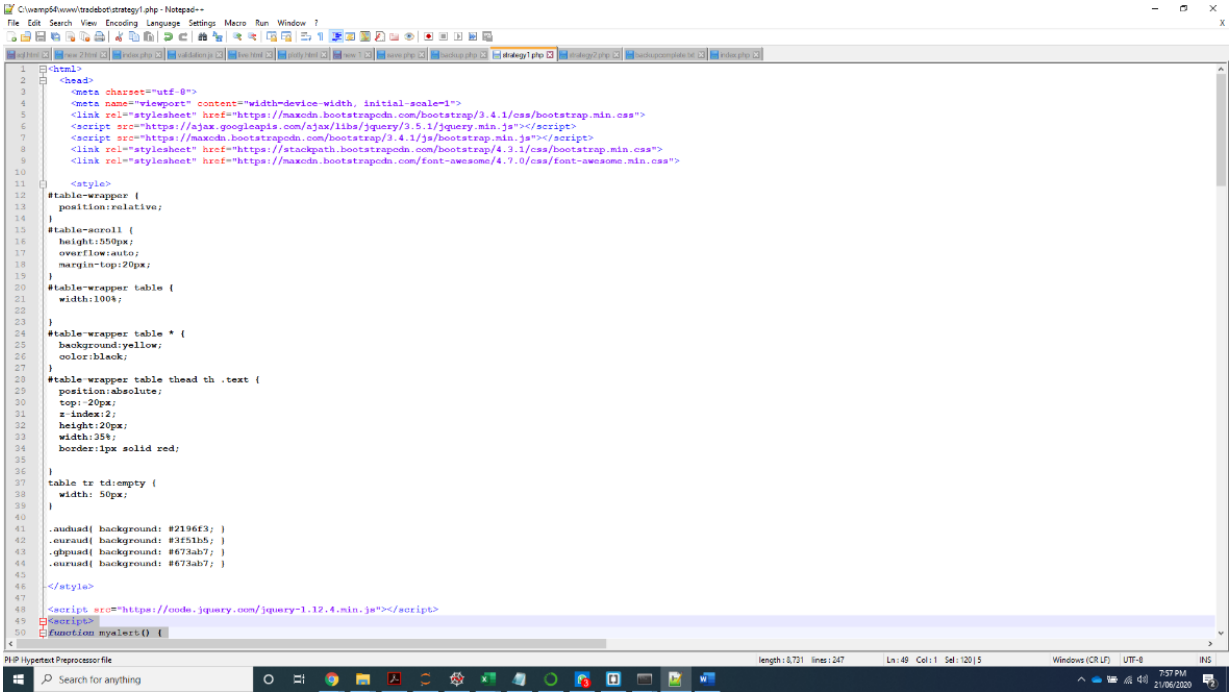
Index.php



```
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>


  <link rel="stylesheet" href="css/style.css">
    </head>

    <body>
        <header>
      <div class="container">
        <div id="branding">
        <h1><span class="highlight">Robo</span> Trade</h1>
        </div>
        </header>
        <div class="container" style = "margin-right:100px">
        <div class="row">


        <div class="col-sm-4" style="vertical-align:middle;margin:400px 0px">
<p>
<a href="strategy1.php">
 <img border="0" src="img/strat1.png" width="250" height="100" >
</a>
</p>
 <p>This strategy is back testing on the live from FXCM which was predicted using Sequential Neural Network <b>using row by row storage implementation.</b> </p>
        </div>


<div class="col-sm-4">
<p>
<a href="index.php">
 <img border="0" src="img/logo.png" width="300" height="300"  style="vertical-align:middle;margin:50px 0px">
</a>
</p>
    </div>

  <div class="col-sm-4" style="vertical-align:middle;margin:400px 0px">
<p>
<a href="strategy2.php">
 <img border="0" src="img/strat2.png" width="250" height="100" >
</a>
</p>
 <p>This strategy is back testing on the live from FXCM which was predicted using Sequential Neural Network <b>using dataframe implementation.</b> </p>
 </div>
</div>
```

Description- The file index.php is used for creating the homepage for the PHP page and this provides the users with two options to select the strategy for which we could view the backtesting results.

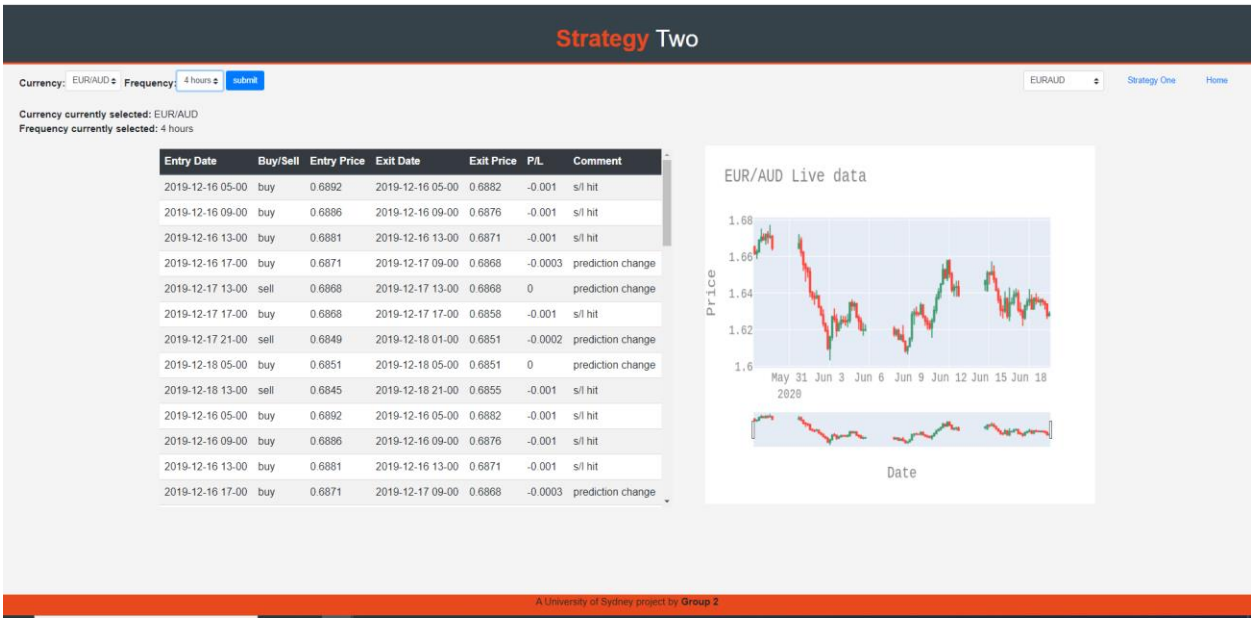
strategy1.php



```
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

    <style>
#table-wrapper {
  position:relative;
}
#table-scroll {
  height:550px;
  overflow:auto;
  margin-top:20px;
}
#table-wrapper table {
  width:100%;

}
#table-wrapper table * {
  background:yellow;
  color:black;
}
#table-wrapper table thead th .text {
  position:absolute;
  top:-20px;
  z-index:2;
  height:20px;
  width:35%;
  border:1px solid red;

}
table tr td:empty {
  width: 50px;
}

.audusd{ background: #2196f3; }
.eurusd{ background: #3f51b5; }
.gbpusd{ background: #673ab7; }
.eurusd{ background: #673ab7; }

</style>

<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
<script>
function myalert() {
```

Description-This file strategy1.php is used to for creating the PHP page for strategy one. This takes the input from the user as the frequency and currency and produces the results in the form of the table along with the input from the user which lets them select the currency to views its 4 hourly live data which has been plotted using plotly. This page also allows the user to easily move from one page to another with ease i.e. homepage and strategy 1

strategy2.php



Description-This file strategy2.php is used to for creating the php page for strategy two. This takes the input from the user as the frequency and currency and produces the results in the form of the table along with the input from the user which lets them select the currency to views its 4 hourly live data which has been plotted using plotly. This page also allows the user to easily move from one page to another with ease i.e. homepage and strategy 1

SNN_train.py

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\dpan7802

Editor - C:\Users\dpan7802\Desktop\Robotrade\Keras_NN_with_softmax_relu\SNN_train.py

temp.py | Keras_NN_live_backtest_row_by_row_input_php.py | Keras_NN_live_backtest_strategy.py | SNN_train.py

```
    classifier.add(Dense(62,activation='relu',kernel_initializer='uniform'))
    classifier.add(Dropout(0.1))

    classifier.add(Dense(62,activation='relu',kernel_initializer='uniform'))
    classifier.add(Dropout(0.1))

    classifier.add(Dense(3,activation="softmax", kernel_initializer="uniform"))
    classifier.compile(optimizer='adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

    return classifier


def enc (row_test):
   if (row_test['Short'] > row_test['Stay'] and
       row_test['Short'] > row_test['Buy']  ):
      return '0'
   else:
       if (row_test['Stay'] > row_test['Short'] and
           row_test['Stay'] > row_test['Buy']    ):
          return '1'
       else:
           if (row_test['Buy'] > row_test['Short'] and
               row_test['Buy'] > row_test['Stay'] ):
               return '2'
           else:
               return 'nan'


def main(dataset = 'AUD_USD_H4_for_keras.csv', operation = 'train', duration = 'H4'):

    try:
        check_for_gpu()
    except:
        pass
    # clean and setup the dataset, return: dataframe
    df = pd.read_csv(dataset)
    df = data_preprocessing(df)

    # save features
    df.to_csv("Dataset_features_extracted.csv")

    # add next day prediction used for training
    df = next_day_predictions(df)

    if operation == 'train':

        #setup to train
        df_X, df_y, X, y, X_train, X_test, y_train, y_test = train_test_split_func(df)
        df_y = to_categorical(df_y)
        y_train = to_categorical(y_train)

        classifier = seq_model(X_train, y_train)
        classifier.fit(X_train, y_train, batch_size=30, epochs=100)
        if duration == 'H4':
            classifier.save('keras_nn_aud_usd_train_with_fit_transform_h4.h5')
        elif duration == 'H1':
            classifier.save('keras_nn_eur_usd_train_with_fit_transform_h1.h5')
```

This code provides the insight of how the training model is showing the output mentioned in the steps above. This code is based on Seq Neural network approach and the training will create a .h5 weighted model file for further prediction.

Keras_NN_live_backtest_strategy.py

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\dpan7802

Editor - C:\Users\dpan7802\Desktop\Robotrade\Keras_NN_with_softmax_relu\Keras_NN_live_backtest_strategy.py

temp.py | Keras_NN_live_backtest_row_by_row_input_php.py | Keras_NN_live_backtest_strategy.py | SNN_train.py

```python
def main(operation = 'run_for', currency_pair = 'all', duration = 'all', past_rows = 200, dbname = 'final2', username = 'root', password = ''):

    ''' Description and suggestions:
            make_csv('pair_name',x); where pair_name can be any currency pairs with 4 decimal points in format, eg: "AUD/USD"
                                      x is the number of rows you'd like to include as historical data
            new_data.csv is the extracted dataset from the FXCM API
            run_all() includes variables, where:
                default predict_pnl = True:
                    this variable is True as default and it displays the Profit/Loss pips according to different Stop-Loss and Take_Profit combinations
                default print_explained_trades = True:
                    this variable is True as default and it displays the explained good trades the model is making
                default return_dataframe = False:
                    this variable is False as default and it returns the explained good trades as a dataframe for for studies and usage

    '''

    if duration == 'all':
        classifier1 = keras.models.load_model('keras_nn_eur_usd_h1_train_with_fit_transform_.h5')
        classifier2 = keras.models.load_model('keras_nn_aud_usd_h4_train_with_fit_transform_.h5')
    else:
        print("Choose duration 'all'")
        classifier1 = False
        classifier2 = False

    make_new_database(dbname= dbname, username=username, password= password)
    duration1 = 'H1'
    duration2 = 'H4'
    past_rows = int(past_rows)

    while True:
        #now = datetime.datetime.now()
        #if now.hour in [10, 14 , 18, 22 ]:
         #   past_rows = int(past_rows)
         #   new_data = make_csv(currency_pair, duration, past_rows)
        #    new_data.to_csv('new_data.csv', index=False)
            for currency_pair in ['AUD/USD', 'EUR/AUD', 'GBP/USD', 'EUR/USD']:
                create_table(dbname = dbname, username=username, password= password, currency_pair=currency_pair, duration=duration1)
                create_table(dbname = dbname, username=username, password= password, currency_pair=currency_pair, duration=duration2)


                new_data1 = make_csv(currency_pair, duration1, past_rows)
                new_data1.to_csv('new_data1.csv', index=False)
                new_data2 = make_csv(currency_pair, duration2, past_rows)
                new_data2.to_csv('new_data2.csv', index=False)

                #new_data1 = pd.read_csv('new_data_h1.csv')
                #new_data2 = pd.read_csv('new_data.csv')

                df1 = new_data1
                df2 = new_data2

                results_h1 = run_all(df1, predict_pnl = True, print_explained_trades = True, return_dataframe = True, threshold = 0.0020, classifier = classifier1)
                results_h2 = run_all(df2, predict_pnl = True, print_explained_trades = True, return_dataframe = True, threshold = 0.0020, classifier = classifier2)

                store_in_sql(results=results_h1, username=username, password=password, dbname= dbname, currency_pair = currency_pair, duration = duration1)
                store_in_sql(results=results_h2, username=username, password=password, dbname= dbname, currency_pair = currency_pair, duration = duration2)

            time.sleep(3600)
```

This snippet shows the function responsible for the output of the anaconda command we represented above. This displays the dataframes extracted from the strategy and pushes to the database created in MySQL which will further be connected to the Webpage for ouput.

Keras_NN_live_backtest_row_by_row_input_php.py

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\dpan7802

Editor - C:\Users\dpan7802\Desktop\Robotrade\Keras_NN_with_softmax_relu\Keras_NN_live_backtest_row_by_row_input_php.py

temp.py | Keras_NN_live_backtest_row_by_row_input_php.py | Keras_NN_live_backtest_strategy.py | SNN_train.py

```
def make_new_database(dbname, username, password): #DON'T PUT PASSWORD,ETC HERE; this is a function
    # Create a connection object
    databaseServerIP            = "127.0.0.1"  # IP address of the MySQL database server
    databaseUserName            = username     # User name of the database server
    databaseUserPassword        = password          # Password for the database user

    newDatabaseName             = dbname    # Name of the database that is to be created
    charSet                     = "utf8mb4"     # Character set
    cusrorType                  = pymysql.cursors.DictCursor

    connectionInstance   = pymysql.connect(host=databaseServerIP, user=databaseUserName, password=databaseUserPassword,
                                   charset=charSet,cursorclass=cusrorType, port =3308) #ENTER PORT NUMBER HERE
    # Create a cursor object
    cursorInsatnce        = connectionInstance.cursor()
    # SQL Statement to create a database
    sqlStatement         = "CREATE DATABASE "+newDatabaseName

    try:
        # Execute the create database SQL statment through the cursor instance
        cursorInsatnce.execute(sqlStatement)
    except:
        pass

    connectionInstance.close()


def create_table(username, password, dbname, currency_pair, duration): #DON'T PUT PASSWORD,ETC HERE; this is a function

    dbServerName     = "127.0.0.1"
    dbUser           = username
    dbPassword       = password          ##Change This
    dbName           = dbname   ##Change this
    charSet          = "utf8mb4"
    cusrorType       = pymysql.cursors.DictCursor
    connectionObject   = pymysql.connect(host=dbServerName, user=dbUser, password=dbPassword,
                                         db=dbName, charset=charSet,cursorclass=cusrorType, port = 3308) #enter port number here
    cursorObject        = connectionObject.cursor()

    currency_pair = currency_pair[:3] + currency_pair[4:]
    print(currency_pair)
    sqlQuery            = "CREATE TABLE " + currency_pair + duration + " (size varchar(30), open_rate varchar(50), open_date varchar(50), sl float, tp float, close_rate varchar(50), close_date varchar(50), P_L varchar(50))"

    try:
        cursorObject.execute(sqlQuery)
    except:
        pass
    connectionObject.close()
    print("The Table " + currency_pair + " has been created!")


def store_in_sql(results,username, password, dbname, currency_pair, duration):#DON'T PUT PASSWORD,ETC HERE; this is a function

    dbServerName     = "127.0.0.1"
    dbUser           = username
    dbPassword       = password       # CHANGE THIS
    dbName           = dbname    # CHANGE THIS
    charSet          = "utf8mb4"
    cusrorType       = pymysql.cursors.DictCursor
    connectionObject   = pymysql.connect(host=dbServerName, user=dbUser, password=dbPassword,
                                         db=dbName, charset=charSet,cursorclass=cusrorType, port=3308) #enter port number here
    cursorObject        = connectionObject.cursor()

    currency_pair = currency_pair[:3] + currency_pair[4:]
    print(currency_pair)

    for ind,row in results.iterrows():
        try:
            sqlQuery            = "INSERT INTO "  + currency_pair + duration + " (size, open_rate , open_date , sl , tp, close_rate, close_date, P_L) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
            inserting_vals = (str(row['size']), str(row['open_rate']), str(row['open_date']), float(row['sl']), float(row['tp']), str(row['close_rate']), str(row['close_date']), str(row['P_L']))
            ## Inserting values...
            cursorObject.execute(sqlQuery, inserting_vals)
            connectionObject.commit()
        except:
            pass

    connectionObject.close()
    print("Data is stored in SQL\n\n\n\n\n")


def main(operation = 'run_for', currency_pair = 'all', duration = 'all', past_rows = 200, dbname = 'final1', username = 'root', password = ''):

    ''' Description and suggestions:
            make_csv('pair_name',x); where pair_name can be any currency pairs with 4 decimal points in format, eg: "AUD/USD"
                                     x is the number of rows you'd like to include as historical data
            new_data.csv is the extracted dataset from the FXCM API
            run_all() includes variables, where:
                default predict_pnl = True:
```

This snippet shows the function responsible for pushing the dataframes created with different currency pairs and durations to the MySQL and then the integration of php webpage is responsible to display the results on an interactive environment.

Output of Keras_NN_livedata_backtesting_row_by_row_1 (1).ipynb

	index	size	open_rate	open_date	sl	tp	close_rate	close_date	P/L
0	0	2	0.67901	2019-11-15 06:00:00	0.67701	0.68201	0.68417	2019-11-19 10:00:00	0.003
1	1	0	0.67870	2019-11-15 10:00:00	0.67670	0.68170	0.68417	2019-11-19 10:00:00	0.003
2	2	0	0.68049	2019-11-15 14:00:00	0.67849	0.68349	0.67728	2019-11-22 02:00:00	-0.002
3	3	0	0.68136	2019-11-15 18:00:00	0.67936	0.68436	0.67824	2019-11-20 18:00:00	-0.002
4	4	0	0.68171	2019-11-17 18:00:00	0.67971	0.68471	0.67917	2019-11-19 06:00:00	-0.002
...	...	...	...	...	...	...	...	...	...
705	705	2	0.68926	2020-06-17 13:00:00	0.68726	0.69226	0.68638	2020-06-17 21:00:00	-0.002
706	706	0	0.69037	2020-06-17 17:00:00	0.68837	0.69337	0.68638	2020-06-17 17:00:00	-0.002
707	707	2	0.68838	2020-06-17 21:00:00	0.68638	0.69138	NaN	NaN	NaN
708	708	0	0.68772	2020-06-18 05:00:00	0.68572	0.69072	NaN	NaN	NaN
709	709	0	0.68577	2020-06-18 13:00:00	0.68377	0.68877	NaN	NaN	NaN

710 rows × 9 columns

```
total = 0
count = 0
profit = 0
positive = 0
negative = 0
for index, row in sy1.iterrows():
    total += 1
    if np.isnan(row['P/L']):
        count += 1
    else:
        #it's divided by 10000 when calculating the sl/tp level, so when calculating the profit, we need to *10000
        profit += row['P/L'] * 10000
        if row['P/L'] > 0:
            positive += 1
        else:
            negative += 1
print("Total amount of trade made: ", total, ", and", count, "trades not closed")
print("There are", positive, "winning trades and", negative, "lossing trades.", "Based on current closed trades, the P/L is : ",
```

```
Total amount of trade made:  708 , and 4 trades not closed
There are 316 winning trades and 388 lossing trades. Based on current closed trades, the P/L is :  97.7999999999759
```

Output of Keras_NN_livedata_backtesting_strategy.ipynb

```
Prediction of top PnL predictions
Total PnL is: 0.1141 stop-loss: 0.001 take-profit: 0.004 , 331 trades
Total PnL is: 0.0471 stop-loss: 0.002 take-profit: 0.003 , 301 trades
Total PnL is: 0.0324 stop-loss: 0.003 take-profit: 0.004 , 253 trades
Total PnL is: 0.0798 stop-loss: 0.003 take-profit: 0.005 , 241 trades
Total PnL is: 0.0575 stop-loss: 0.004 take-profit: 0.006 , 208 trades
Total PnL is: 0.0588 stop-loss: 0.005 take-profit: 0.0075 , 194 trades

The best combination of t/p and s/l is --> take-profit: 0.004 stop-loss: 0.001
Explained good trades performed
```

Entry date	B/S	Entry price	Exit date	Exit price	PnL	Comment
2019-11-15 14:00:00	buy	0.6805	2019-11-18 22:00:00	0.6795	-0.0010	s/l hit
2019-11-20 18:00:00	buy	0.6795	2019-11-21 18:00:00	0.6785	-0.0010	s/l hit
2019-11-22 06:00:00	buy	0.6785	2019-11-25 10:00:00	0.6775	-0.0010	s/l hit
2019-11-26 02:00:00	buy	0.6777	2019-11-27 18:00:00	0.6775	-0.0002	prediction change
2019-11-27 22:00:00	sell	0.6775	2019-11-28 14:00:00	0.6765	0.0010	prediction change
2019-11-28 18:00:00	buy	0.6765	2019-11-29 14:00:00	0.6755	-0.0010	s/l hit
2019-12-15 22:00:00	buy	0.6883	2019-12-15 22:00:00	0.6873	-0.0010	s/l hit
2019-12-16 02:00:00	buy	0.6881	2019-12-16 02:00:00	0.6871	-0.0010	s/l hit
2019-12-16 06:00:00	buy	0.6870	2019-12-17 06:00:00	0.6860	-0.0010	s/l hit
2019-12-18 02:00:00	buy	0.6844	2019-12-19 14:00:00	0.6884	0.0040	t/p hit
2020-01-03 18:00:00	buy	0.6963	2020-01-03 18:00:00	0.6953	-0.0010	s/l hit
2020-01-08 18:00:00	sell	0.6876	2020-01-10 10:00:00	0.6886	-0.0010	s/l hit
2020-01-13 22:00:00	buy	0.6903	2020-01-13 22:00:00	0.6893	-0.0010	s/l hit
2020-01-14 02:00:00	buy	0.6891	2020-01-15 10:00:00	0.6881	-0.0010	s/l hit
2020-01-15 18:00:00	buy	0.6909	2020-01-15 18:00:00	0.6899	-0.0010	s/l hit
2020-01-16 18:00:00	buy	0.6892	2020-01-17 14:00:00	0.6882	-0.0010	s/l hit
2020-01-20 10:00:00	buy	0.6871	2020-01-20 10:00:00	0.6861	-0.0010	s/l hit
2020-01-21 02:00:00	buy	0.6861	2020-01-21 06:00:00	0.6851	-0.0010	s/l hit
2020-01-24 10:00:00	buy	0.6838	2020-01-24 14:00:00	0.6828	-0.0010	s/l hit
2020-01-29 06:00:00	sell	0.6770	2020-01-30 06:00:00	0.6730	0.0040	t/p hit
2020-02-02 18:00:00	buy	0.6686	2020-02-04 06:00:00	0.6726	0.0040	t/p hit
2020-02-04 14:00:00	sell	0.6721	2020-02-04 14:00:00	0.6731	-0.0010	s/l hit
2020-02-05 22:00:00	buy	0.6745	2020-02-06 14:00:00	0.6735	-0.0010	s/l hit