The University of Hong Kong

Faculty of Engineering

Department of Computer Science

COMP7704
Computing Projects in Algo Finance
(Trading Strategy of Strategies)

Submitted in partial fulfillment of the requirements for the admission to the degree of Master of Science in Computer Science

By
CHAN Yiu Keung, Joseph
2011565354

Supervisor: Dr. Hilton K.H. Chan
Date of submission: 01/07/2019

Abstract

This study tries to make use of arbitrary features (aka: signals or columns) to predict future price change in Foreign Exchange Market (Forex). In particular, technical indicators (eg: MACD, RSI etc) and pattern analysis are introduced as the features of each timestep. To make use of multiple features, a composite neural network (**Bi-LSTM + BERT**) is trained and evaluated on 4 different currency pairs (**USDCAD**, **USDJPY**, **AUDUSD**, **EURUSD**) separately.

In this thesis, **Bi**directional **L**ong **S**hort-**T**erm **M**emory (Bi-LSTM) is used as the encoder in the composite neural network (NN). **L**ong **S**hort-**T**erm **M**emory (LSTM) networks have been used successfully in many sequence-to-sequence analyses, eg: translations. Unlike unidirectional LSTM which only preserves information from left-to-right, Bi-LSTM will evaluate a sequence in both left-to-right and right-to-left, helping the NN model to understand context better.

Apart from Bi-LSTM, the **self-attention** technique used in the latest (2018/19) bleeding edge machine learning model, BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers), published by GOOGLE, would be used and integrated with the Bi-LSTM encoder. Self-attention technique is a way to understand the underlying relationship between each element(vector) in a sequence. Thanks to self-attention technique, BERT has been proven to outperform previous NLP (Natural Language Processing) models in understanding contextual relationship in large text corpus (like Wikipedia).

In addition to the composite neural network, some data pre-processing techniques are used for data cleaning and features engineering. In particular, an **outlier detection** algorithm from previous academic paper is used for cleaning up high frequency tick data. An OHLC (Candle) **resampling technique** based on Bid volume of tick data is also introduced to achieve stable statistical properties in the input data. **Random forest** algorithm is also applied to filter unimportant features to reduce the input dimension (number of features) due to machine resource consideration.

With the help of self-attention layers from BERT and the bidirectional LSTM encoder, the model outperforms the underlying asset, and achieves 8.36% average best return for four currency pairs (USDJPY, USDCAD, AUDUSD, EURUSD) in the period **01-Jul-2018** to **18-May-2019** (~11 Months). After applying the iterative training approach, the average best return even increases to 9.12%.

Declaration

I declare that this dissertation “Computing Project in Algo Finance (Trading Strategy of Strategies)” represents my own work, except where the acknowledgement is made. this dissertation does not include previous theses, materials or reports that have been submitted to The University of Hong Kong or to other institutions for a degree, diploma or other qualifications.

Acknowledgements

I would like to give my sincere thanks to my supervisor Dr. Hilton Chan for his great advice to my dissertation. The comments he gave me are really helpful for adjusting my mentality, and guiding me to a right direction in the dissertation.

A special thanks would be given to my family and my girlfriend for their encouragement and understanding throughout the dissertation.

Table of Content

Chapter 1: Introduction

1.1 Research Objective & Scope

Using multiple trading signals in a trading strategy is very popular among traders. Traders often make use of combinations of technical indicators, eg: RSI with the MACD to determine the time of entry and exit in the market. The trading signals are not only limited to technical indicators, but also other kinds of signals, eg: candle patterns analysis, sentiment analysis etc.

However, finding a profitable combination of signals manually is a tedious process which involves fine turning hyper-parameters based on back-testing results. The problem becomes even more complicated when considering the dynamic natures of markets. For example, a strategy works well in a market doesn't mean it could also work well in another market. Such dynamic market environment often costs traders lots of time and effort to find out a suitable strategy. With the advancement of machine learning techniques and ready-to-use libraries, there are opportunities for us to explore different signal combinations automatically.

The beauty of machine learning is the ability to find out the function that could map X → Y, where X is the vector(s) of feature(s). Supervised learning approach would be adopted in this thesis, i.e. Given input X and expected output Y, the trained model would be the mapping function. During the data pre-processing / training stage, the algorithm is expected to determine which

features are important for prediction, and it should get rid of useless features by discarding them during data pre-processing, or setting the weight of those useless features to nearly 0 during backpropagation. For example, if a feature is generated by random generator, the feature should be ignored after preliminary analysis or training.

In this dissertation, different machine learning models would be explored for predicting Forex market using multiple trading signals. In particular, some popular classification models would be tested, and a composite neural network (Bi-LSTM + BERT) would be used and evaluated as our final deliverable.

1.2 Foreign Exchange Market (Forex)

1.2.1 Why Forex?

Training machine learning models often requires lots of data. Most of the market data are very expensive, while free data often does not provide high resolution. Thus, we need to choose a market that has large amount of training data available at a reasonable cost.

Forex is a good candidate for machine learning because of the following properties:

1. Large amount of data available for training
    a. Long Trading Hour

        Forex can be traded 24 hours a day, 5.5 days a week, 12 months a year [1]

Figure 1 – FX Market Trading Hour (GMT)

b. High Turnover

The **daily turnover** of global FX market was over 5 trillion US dollars in 2013. The trend keeps increasing over the years.

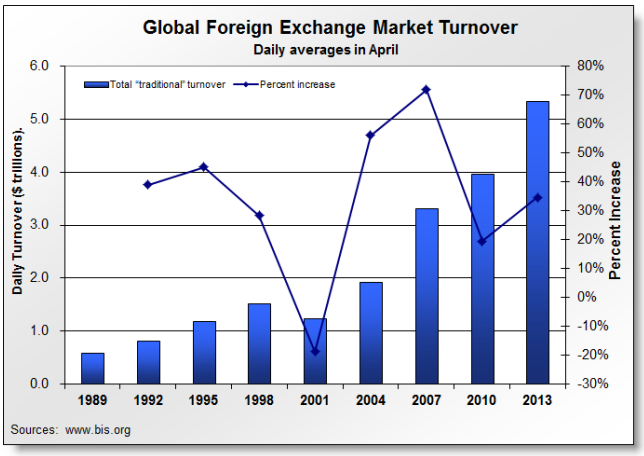


Figure 2 - Main Foreign Exchange Market Turnover [2]

2. High resolution data (tick / minute) is available for FREE

1.2.2 PIP and Leveraging

The spread of Forex market is extremely small as compared with stock market. The change of the price is measured in PIP. One PIP is equal to 0.0001. Due to the low spread nature, trading in Forex market often requires leveraging in order to obtain reasonable return.

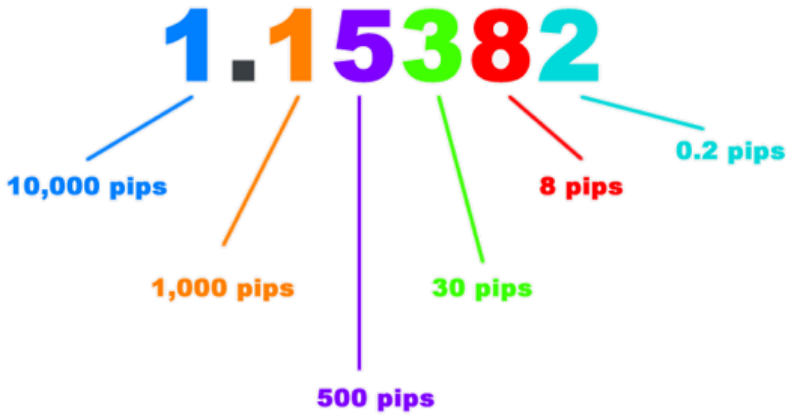


Figure 3 – PIP Example

Chapter 2: Market Data Pre-processing

2.1 Data Source

Tick data of four currency pairs (**USDCAD**, **USDJPY**, **EURUSD**, **AUDUSD**) is retrieved from Dukascopy Swiss Banking Group [3]. The raw data of each pair is stored as *.csv file format.

Data Period: **01-Jan-2005** to **18-May-2019**

Gmt time	Ask	Bid	AskVolume	BidVolume
20.05.2019 00:00:00.013	110.175	110.173	1	2.44
20.05.2019 00:00:00.115	110.178	110.174	4.6	1.25
20.05.2019 00:00:00.249	110.178	110.174	1.25	2.12

Figure 4 – Sample Data Of USDJPY

(Notes: The unit of AskVolume and BidVolume are in “Millions”)

2.2 Data Cleaning

High quality tick data from market is not guarantee. Raw data may contain invalid entries, for example invalid timestamp format, duplicated entries with the same timestamp, or unexpected negative price value etc. Apart from invalid entries, spikes are also very common in the series of tick data. In order to ensure the quality of our data, data cleaning is a must before feeding the data into our models.

2.2.1 Invalid Format Detection

Tick data in each row should be in the same format. The following cases are classified as invalid format:

1. Negative value
2. Date which is not in the format of “dd.MM.YYYY HH:mm:ss.SSS”
3. Missing data, eg: 0 or empty

Rows with invalid format would be discarded

2.2.2 Cleaning Duplicated Rows

The **timestamp** of each row must be unique, such that timestamps of any two rows must be different. Duplicated rows with the same timestamp would be discarded.

2.2.3 Cleaning Outlier Rows

The following outlier detection algorithm from previous academic paper (A method to "clean up" ultra high-frequency data) is adopted in this thesis [4].

Let $\{p_i\}_{i=1}^{N}$ be an ordered tick-by-tick price series

$$(|p_i - \bar{p}_{-i}(k)| < 3s_{-i}(k) + \varphi) = \begin{cases} TRUE & observation\ i\ is\ kept \\ FALSE & observation\ i\ is\ removed \end{cases} \quad (2.1)$$

where $\bar{p}_{-i}(k)$ and $s_{-i}(k)$ denote respectively the mean and the standard deviation of a neighborhood of k observations around i without the i-th observation and φ is the granularity parameter, i.e.

$$\bar{p}_{-i}(k) = mean(p_{i-k}, p_{i-k+1}, \dots, p_{i-1}, p_{i+1}, \dots, p_{i+k-1}, p_{i+k})$$

$$s_{-i}(k) = std(p_{i-k}, p_{i-k+1}, \dots, p_{i-1}, p_{i+1}, \dots, p_{i+k-1}, p_{i+k})$$

Take USDJPY as an example, let k = 60, i = 60, φ = 0.002

$$\bar{p}_{-60}(60) = \text{mean}(p_0, p_1, p_2, \dots, p_{59}, \quad p_{61}, p_{62}, \dots, p_{120})$$

$$s_{-60}(60) = \text{std}(p_0, p_1, p_2, \dots, p_{59}, \quad p_{61}, p_{62}, \dots, p_{120})$$

Where φ = Minimum price unit * 2 = 0.001 * 2 = 0.002

This approach takes the series before and after a timestamp into consideration. It helps removing invalid sudden spikes while keeping valid large price change. The algorithm is applied to the series of “Bid”, “BidVolume”, “Ask”, “AskVolume”. Whenever a tick data contains outlier value in either one of the series, the tick would be discarded.

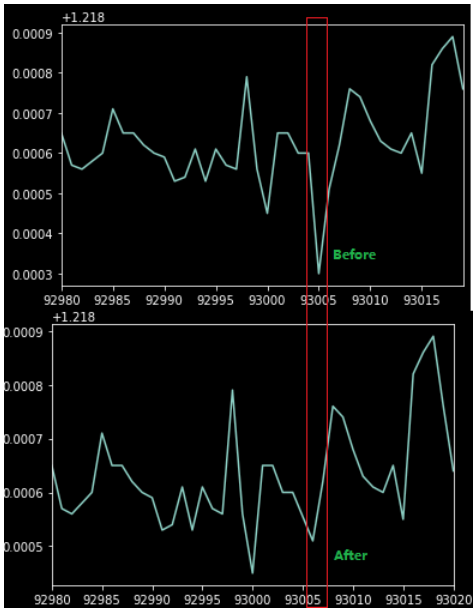


Figure 5 – Example of Outlier Cleaning

2.2.4 Data Gap Handling

Data gap is defined as those long period (>15mins) which has missing data during trading hours. One of the possible root causes of data gap is system failure, in which tick data during the failure period could not be recorded. If the value after the data gap has large difference with the value before the gap, the large difference might introduce instability (noise) to our models.

The following algorithm is used to detect data gap on **Bid price** only:

Let $\{\Delta p_i\}_{i=1}^{N}$ be an ordered tick-by-tick series of absolute price changes, i.e.

$$\Delta p_i = |p_i - p_{i-1}|$$

Let $\{\Delta T_i\}_{i=1}^{N}$ be an ordered tick-by-tick series of time difference (in seconds), i.e.

$$\Delta T_i = T_i - T_{i-1}$$

$$(\Delta T_i > 15mins)\ \&\ (|\Delta p_i - \overline{\Delta p}_{-i}(k)| \geq 2 * \Delta s_{-i}(k)) = \begin{cases} TRUE & observation\ i\ is\ a\ Gap \\ FALSE & observation\ i\ is\ not\ a\ Gap \end{cases} \quad (2.2)$$

where $\overline{\Delta p}_{-i}(k)$ and $\Delta s_{-i}(k)$ denote respectively the mean and the standard deviation of the past k observations right before i, i.e.

$$\overline{\Delta p}_{-i}(k) = mean(\Delta p_{i-k}, \Delta p_{i-k+1}, \Delta p_{i-k+2}, \dots, \Delta p_{i-1})$$

$$\overline{\Delta s}_{-i}(k) = std(\Delta p_{i-k}, \Delta p_{i-k+1}, \Delta p_{i-k+2}, \dots, \Delta p_{i-1})$$

After gaps are identified, the ticks are then split into multiple dataset by gaps

|< ------D1------ >|< ------D2----- > | ……..… | < -------Dn----- > |

start gap gap … gap end

i.e. no gaps within each dataset D1…Dn

	Timestamp	Bid	Ask	BidVolume	AskVolume	Time_Diff_In_Seconds	Is_Data_Gap
64589	2005-01-07 21:59:42.356	1.231	1.232	1.600	23.900	7.739	False
64590	2005-01-07 21:59:55.434	1.231	1.232	22.600	4.000	13.078	False
64591	2005-01-09 22:00:01.200	1.233	1.233	6.800	17.500	172,805.766	True
64592	2005-01-09 22:00:25.701	1.233	1.233	13.500	22.500	24.501	False
64593	2005-01-09 22:00:37.704	1.233	1.233	27.800	23.300	12.003	False
64594	2005-01-09 22:00:52.166	1.233	1.233	0.800	3.200	14.462	False

Figure 6 – Data Gap Example

The algorithm does not take non-trading hours into account. There might have **false-positive** cases such that gap due to non-trading hours might be classified as data gap if the price change is large enough. Nonetheless, it is okay to construct training samples from multiple dataset D1…Dn as long as the dataset Di has enough timesteps for particular model. This dataset splitting method only aims at ensuring each training sample does not contain data gap.

2.2.5 Settings and Statistical Summary

Data Period: **01-Jan-2005** to **18-May-2019**

	USDJPY	USDCAD	AUDUSD	EURUSD
Outlier Detector	Bid/Ask: k = 60; φ = 0.002 Bid/Ask Volume: k = 60; φ = 0.02	Bid/Ask: k = 60; φ = 0.00002 Bid/Ask Volume: k = 60; φ = 0.02	Bid/Ask: k = 60; φ = 0.00002 Bid/Ask Volume: k = 60; φ = 0.02	Bid/Ask: k = 60; φ = 0.00002 Bid/Ask Volume: k = 60; φ = 0.02
Gap Detector	k = 10000	k = 10000	k = 10000	k = 10000
Total Ticks	273970299	227961868	233175583	309752541
Ticks with Invalid Formats	0	0	0	0
Ticks with duplicated Timestamp	0	0	0	0
Outlier Ticks (bid/ask/bid volume/ask volume)	9135335 (3.33%)	8413212 (3.69%)	9165824 (3.93%)	13058275 (4.22%)
# of Gaps	729	710	774	709

2.3 Data Re-sampling

Ticks data is resampled into “Candlesticks” which is used to represent OHLC (Open, High, Low, Close) prices for some specific period of time (eg: minute, hour, day).

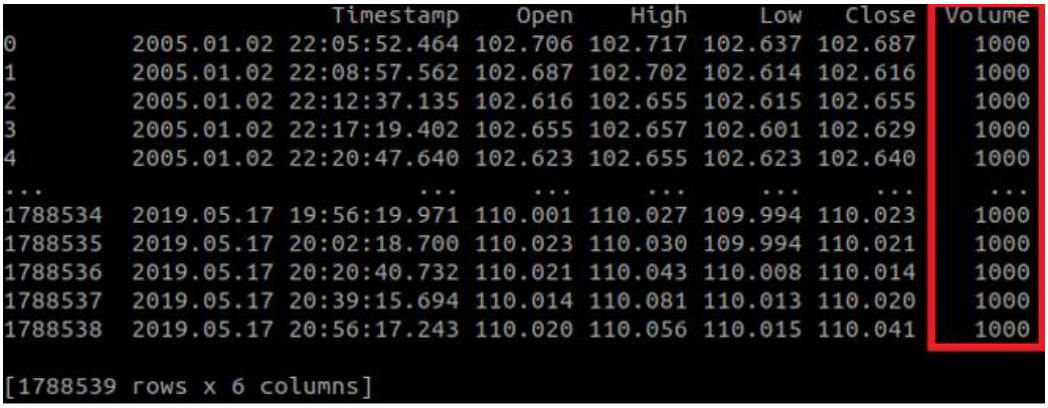
A straight forward approach of resampling would be using minute to minute OHLC dataset. However, the market **DOES NOT** follow this timing rule. The trading volume in some periods are extremely low as compared with the market opening period.

According to the book “Advances in Financial Machine Learning by Marcos Lopez De Prado” [5], it indicates that instead of separating the dataset using time bar, we could use different sampling rules, eg:

1. Ticks: OHLC data for every N ticks traded
2. Volume: OHLC data for every N volume traded
3. Dollar: OHLC data for every N dollars traded

The objective is to under-sample those periods with low activities to achieve **stable statistical properties.** [5]

In the dissertation, ticks are resampled per **1000-BidVolume**, and Bid prices are used to construct the OHLC bar in each timestep. After adding up the "BidVolume", the total value may be larger than 1000. In such case, the remaining volumes and Bid price would be considered in the next bar. The last row would be discarded if its volume is less than 1000.



	Timestamp	Open	High	Low	Close	Volume
0	2005.01.02 22:05:52.464	102.706	102.717	102.637	102.687	1000
1	2005.01.02 22:08:57.562	102.687	102.702	102.614	102.616	1000
2	2005.01.02 22:12:37.135	102.616	102.655	102.615	102.655	1000
3	2005.01.02 22:17:19.402	102.655	102.657	102.601	102.629	1000
4	2005.01.02 22:20:47.640	102.623	102.655	102.623	102.640	1000
...	...	...	...	...	...	...
1788534	2019.05.17 19:56:19.971	110.001	110.027	109.994	110.023	1000
1788535	2019.05.17 20:02:18.700	110.023	110.030	109.994	110.021	1000
1788536	2019.05.17 20:20:40.732	110.021	110.043	110.008	110.014	1000
1788537	2019.05.17 20:39:15.694	110.014	110.081	110.013	110.020	1000
1788538	2019.05.17 20:56:17.243	110.020	110.056	110.015	110.041	1000

[1788539 rows x 6 columns]

Figure 7 – Ticks Resampling Into OHLC For USDJPY

2.4 Features Generation

2.4.1 Basic Features

For each OHLC row, the following features (columns) would be generated:

- $\text{Row}_i[\text{'Open} - \text{Close'}] = \text{Row}_i[\text{'Open'}] - \text{Row}_i[\text{'Close'}]$ *(2.3)*
- $\text{Row}_i[\text{'High} - \text{Low'}] = \text{Row}_i[\text{'High}] - \text{Row}_i[\text{'Low}]$ *(2.4)*
- $\text{Row}_i[\text{'Log\_Return'}] = Log(\text{Row}_i[\text{'Close}]) - \text{Log}(\text{Row}_i[\text{'Open'}])$ *(2.5)*
- $\text{Row}_i[\text{'PIP\_Return'}] = (\text{Row}_i[\text{'Close'}] - \text{Row}_i[\text{'Open'}]) * 10000$ *(2.6)*
- $\text{Row}_i[\text{'Time\_Diff\_Freq'}] = \frac{1}{(Row_i['Timestamp'] - Row_{i-1}['Timestamp'])\ seconds}$ *(2.7)*

Where Row_i is an 1000-volume-resampled OHLC row with index i

Notes:

1. "Open-Close" and "High-Low" are constructed to reflect the OHLC bar width.
2. Time_Diff_Freq is used to reflect the frequency of trading activities.

   eg: Some inactive trading period would take longer time to accumulate 1000 Volume, i.e. smaller Time_Diff_Freq. The red area in the picture below shows relatively higher Time_Diff_Freq because the time difference with the previous bar is shorter.

	Timestamp	Open	High	Low	Close	Volume	Time_Diff_Freq
308180	2007-05-01 09:49:02.026	119.575	119.585	119.575	119.580	1,000.000	0.014
308181	2007-05-01 09:49:23.003	119.580	119.580	119.575	119.580	1,000.000	0.048
308182	2007-05-01 09:49:44.301	119.580	119.580	119.575	119.575	1,000.000	0.047
308183	2007-05-01 09:50:10.132	119.575	119.575	119.565	119.570	1,000.000	0.039
308184	2007-05-01 09:50:14.682	119.570	119.580	119.570	119.580	1,000.000	0.220
308185	2007-05-01 09:50:34.810	119.580	119.580	119.570	119.570	1,000.000	0.050
308186	2007-05-01 09:50:43.107	119.570	119.580	119.570	119.580	1,000.000	0.121
308187	2007-05-01 09:50:44.833	119.580	119.595	119.580	119.585	1,000.000	0.579
308188	2007-05-01 09:51:19.523	119.585	119.585	119.580	119.580	1,000.000	0.029
308189	2007-05-01 09:51:34.990	119.580	119.580	119.580	119.580	1,000.000	0.065
308190	2007-05-01 09:52:20.012	119.580	119.585	119.580	119.585	1,000.000	0.022
308191	2007-05-01 09:53:14.718	119.585	119.585	119.585	119.585	1,000.000	0.018
308192	2007-05-01 09:53:34.996	119.585	119.600	119.585	119.600	1,000.000	0.049
308193	2007-05-01 09:53:58.182	119.600	119.600	119.595	119.600	1,000.000	0.043
308194	2007-05-01 09:54:50.538	119.600	119.600	119.590	119.600	1,000.000	0.019

Figure 8 – Example of Time_Diff_Freq

3. Features are normalized using [MinMaxScaler](#) into range [0, 1] with **600-sliding window** [6]

   i.e. Use the corresponding values of previous 600 timesteps to normalize current value

$$MinMaxScaled_x = \frac{X - X.\min()}{\mathrm{X.max()} - X.\min()} \qquad (2.8)$$

The following normalized values are used for training machine learning models instead of original values:

	Normalized Feature Name Mapping
Time_Diff_Freq	MinMaxScaled_Time_Diff_Freq
Open	MinMaxScaled_Open
High	MinMaxScaled_High
Low	MinMaxScaled_Low
Close	MinMaxScaled_Close
Open-Close	MinMaxScaled_Open-Close
High-Low	MinMaxScaled_High-Low
Log_Return	MinMaxScaled_Log_Return
PIP_Return	MinMaxScaled_PIP_Return

	Timestamp	MinMaxScaled_Time_Diff_Freq	MinMaxScaled_Open	MinMaxScaled_High	MinMaxScaled_Low	MinMaxScaled_Close	Mi
0	2005-01-25 11:37:26.555	0.099	0.268	0.232	0.264	0.251	
1	2005-01-25 11:37:35.865	0.110	0.251	0.208	0.226	0.229	
2	2005-01-25 11:38:33.038	0.016	0.229	0.245	0.226	0.227	
3	2005-01-25 11:39:43.593	0.013	0.242	0.189	0.202	0.200	
4	2005-01-25 11:41:09.013	0.010	0.200	0.172	0.202	0.159	
...	...	...	...	...	...	...	
449637	2019-05-17 19:23:40.034	0.200	0.007	0.004	0.005	0.009	
449638	2019-05-17 19:34:42.593	0.156	0.009	0.004	0.005	0.003	
449639	2019-05-17 19:42:31.486	0.231	0.003	0.003	0.001	0.000	
449640	2019-05-17 19:57:31.123	0.109	0.000	0.004	0.001	0.007	
449641	2019-05-17 20:40:01.040	0.024	0.007	0.007	0.001	0.013	

Figure 9 – Example of MinMaxScaled Features

2.4.2 Technical Indicators

Some commonly used momentum indicators are generated by Talib as our features based on 1000-volume-resampled OHLC [7].

Type	Features	Description
Momentum Indicators	MACD	Moving Average Convergence/Divergence
	RSI	Relative Strength Index
	ADX	Average Directional Movement Index
	STOCH	Stochastic Oscillator Slow
	STOCHF	Stochastic Oscillator Fast
	DX	Directional Movement Index

The inputs are the series of values of a particular column in the 1000-volume-resampled OHLC

eg: Series['Close'] is a [m x 1] matrix where m is the number of timesteps(rows).

Each output is also in the form of [m x 1] matrix.

1. Moving Average Convergence/Divergence (MACD)

   Inputs:

   - MACD(Series['Close'], fastperiod=12, slowperiod=26, signalperiod=9)

   Outputs:

   - "technical_MACD_short_hold_long" =

     1 if MACD crosses above its signal line (Long)

     -1 if MACD crosses below the signal line (Short)

     0 Otherwise

   - "technical_MACD_histogram" = "MACD" – "Signal"

2. Relative Strength Index (RSI)

   Inputs:

   - RSI(Series[‘Close’], timeperiod=14)

   Outputs

   - “technical_RSI”, $\{x \in R: 0 \leq x \leq 1\}$

3. Average Directional Movement Index (ADX)

   Inputs:

   - ADX(Series[‘High’], Series[‘Low], Series[‘Close’], timeperiod=14)

   Outputs:

   - “technical_ADX”, $\{x \in R: 0 \leq x \leq 1\}$

4. Stochastic Oscillator Slow (STOCH)

   Inputs:

   - STOCH(Series[‘High’], Series[‘Low’], Series[‘Close’], fastk_period=5, slowk_period=3, slowd_period=3)

   Outputs:

   - “technical_STOCH_slowk”, $\{x \in R: 0 \leq x \leq 1\}$
   - “technical_STOCH_slowd”, $\{x \in R: 0 \leq x \leq 1\}$

5. Stochastic Oscillator Fast (STOCHF)

Inputs:

- STOCHF(Series['High'], Series['Low'], Series['Close'], fastk_period=5, fastd_period=3)

Outputs:

- "technical_STOCHF_fastk", $\{x \in R: 0 \leq x \leq 1\}$
- "technical_STOCHF_fastd", $\{x \in R: 0 \leq x \leq 1\}$

6. Directional Movement Index (DX)

Inputs:

- DX(Series['High'], Series['Low'], Series['Close'], timeperiod=14)

Outputs:

- "technical_DX", $\{x \in R: 0 \leq x \leq 1\}$

The beginning of the output (m x 1) series may contain NA (not available) value because of sliding windows of calculation, eg: [NA, NA, NA, 0.3, …, 0.6].

Rows with NA value in its attributes would be **discarded**.

	Timestamp	technical_MACD_short_hold_long	technical_MACD_histogram	...	technical_STOCHF_fastk	te
0	2005-01-21 10:35:38.924	0.000	0.000	...	0.770	
1	2005-01-21 10:40:52.247	0.000	0.000	...	0.121	
2	2005-01-21 10:48:22.955	0.000	0.000	...	0.226	
3	2005-01-21 10:52:03.085	0.000	0.000	...	0.271	
4	2005-01-21 10:55:58.024	0.000	0.000	...	0.346	
...	...	...	...	...	...	
449678	2019-05-17 19:23:40.034	0.000	-0.000	...	0.463	
449679	2019-05-17 19:34:42.593	0.000	-0.000	...	0.345	
449680	2019-05-17 19:42:31.486	0.000	-0.000	...	0.103	
449681	2019-05-17 19:57:31.123	0.000	-0.000	...	0.619	
449682	2019-05-17 20:40:01.040	0.000	-0.000	...	0.875	

Figure 10 – Example of Technical Indicator Features

9 columns are added to each timesteps of the 1000-volume-resampled OHLC:

Feature Name
technical_MACD_short_hold_long
technical_MACD_histogram
technical_RSI
technical_ADX
technical_STOCH_slowk
technical_STOCH_slowd
technical_STOCHF_fastk
technical_STOCHF_fastd
technical_DX

2.4.3 Candlestick Patterns Indicators

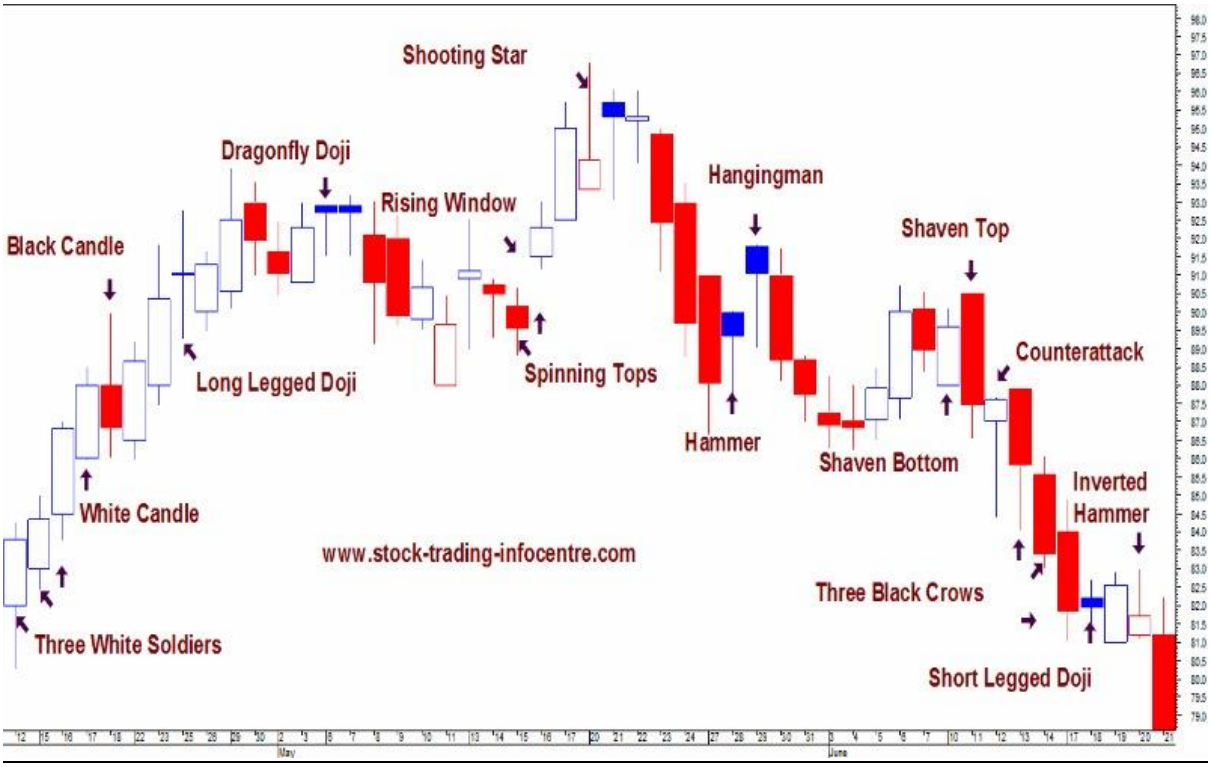


Figure 11 – Example of Pattern Analysis

Talib is used to detect the following patterns from 1000-volume-resampled OHLC [8].

Features	**Description**
CDL2CROWS	Two Crows
CDL3BLACKCROWS	Three Black Crows
CDL3INSIDE	Three Inside Up/Down
CDL3LINESTRIKE	Three-Line Strike
CDL3OUTSIDE	Three Outside Up/Down
CDL3STARSINSOUTH	Three Stars In The South
CDL3WHITESOLDIERS	Three Advancing White Soldiers
CDLABANDONEDBABY	Abandoned Baby
CDLADVANCEBLOCK	Advance Block
CDLBELTHOLD	Belt-hold
CDLBREAKAWAY	Breakaway
CDLCLOSINGMARUBOZU	Closing Marubozu
CDLCONCEALBABYSWALL	Concealing Baby Swallow

CDLCOUNTERATTACK	Counterattack
CDLDARKCLOUDCOVER	Dark Cloud Cover
CDLDOJI	Doji
CDLDOJISTAR	Doji Star
CDLDRAGONFLYDOJI	Dragonfly Doji
CDLENGULFING	Engulfing Pattern
CDLEVENINGDOJISTAR	Evening Doji Star
CDLEVENINGSTAR	Evening Star
CDLGAPSIDESIDEWHITE	Up/Down-gap side-by-side white lines
CDLGRAVESTONEDOJI	Gravestone Doji
CDLHAMMER	Hammer
CDLHANGINGMAN	Hanging Man
CDLHARAMI	Harami Pattern
CDLHARAMICROSS	Harami Cross Pattern
CDLHIGHWAVE	High-Wave Candle
CDLHIKKAKE	Hikkake Pattern
CDLHIKKAKEMOD	Modified Hikkake Pattern
CDLHOMINGPIGEON	Homing Pigeon
CDLIDENTICAL3CROWS	Identical Three Crows
CDLINNECK	In-Neck Pattern
CDLINVERTEDHAMMER	Inverted Hammer
CDLKICKING	Kicking
CDLKICKINGBYLENGTH	Kicking - bull/bear determined by the longer marubozu
CDLLADDERBOTTOM	Ladder Bottom
CDLLONGLEGGEDDOJI	Long Legged Doji
CDLLONGLINE	Long Line Candle
CDLMARUBOZU	Marubozu
CDLMATCHINGLOW	Matching Low
CDLMATHOLD	Mat Hold
CDLMORNINGDOJISTAR	Morning Doji Star
CDLMORNINGSTAR	Morning Star
CDLONNECK	On-Neck Pattern
CDLPIERCING	Piercing Pattern
CDLRICKSHAWMAN	Rickshaw Man
CDLRISEFALL3METHODS	Rising/Falling Three Methods
CDLSEPARATINGLINES	Separating Lines

CDLSHOOTINGSTAR	Shooting Star
CDLSHORTLINE	Short Line Candle
CDLSPINNINGTOP	Spinning Top
CDLSTALLEDPATTERN	Stalled Pattern
CDLSTICKSANDWICH	Stick Sandwich
CDLTAKURI	Takuri (Dragonfly Doji with very long lower shadow)
CDLTASUKIGAP	Tasuki Gap
CDLTHRUSTING	Thrusting Pattern
CDLTRISTAR	Tristar Pattern
CDLUNIQUE3RIVER	Unique 3 River
CDLUPSIDEGAP2CROWS	Upside Gap Two Crows
CDLXSIDEGAP3METHODS	Upside/Downside Gap Three Methods

Notes:

1. All pattern detector takes OHLC series as inputs:

   Series[‘Open’], Series[‘High’], Series[‘Low’], Series[‘Close’]

2. The output of each detector is a [m x 1] matrix where m is the length of input series.

3. Each value in the output series is one of the three possible values: [-1, 0, 1] where -1 means Bear, 0 means Nothing Detected, 1 means Bull
   - All the pattern detector functions of Talib only output -100, 0, or 100
   - The output is divided by 100 for down scaling

4. Inputs:
   - Tick data is time-resampled using Bid with following resolutions: [1min, 5min, 15min, 30min, 1H, 3H, 6H, D]
   - Detect patterns from the **time -resampled OHLC** datasets
   - Since the timestamp of 1000-volume-resampled OHLC could not be mapped directly into time -resampled OHLC datasets, last value would be used,

     Example:

     Assume a 15mins_CDL2CROWS pattern at 10:30:00.000 is 1

     The 15mins_CDL2CROWS of the row 10:35:38.924 in the 1000-volume-resampled OHLC would use the closest available value in the past, i.e. 10:30:00.000

5. Outputs

Total Number of Pattern Detectors Available = 61

Total Number of time-resampled OHLC ([1min, 5min, 15min, 30min, 1H, 3H, 6H, D]) = 8

Therefore, the total number of patterns features generated = 61 x 8 = **488**

i.e. 488 new columns are added to each rows of 1000-volume-resampled OHLC

['patterns_1min_CDL2CROWS', …., 'patterns_1min_CDLXSIDEGAP3METHODS', 'patterns_5min_CDL2CROWS', …., 'patterns_5min_CDLXSIDEGAP3METHODS', 'patterns_15min_CDL2CROWS', …., 'patterns_15min_CDLXSIDEGAP3METHODS', 'patterns_30min_CDL2CROWS', …., 'patterns_30min_CDLXSIDEGAP3METHODS', 'patterns_1H_CDL2CROWS', …., 'patterns_1H_CDLXSIDEGAP3METHODS', 'patterns_3H_CDL2CROWS', …., 'patterns_3H_CDLXSIDEGAP3METHODS', 'patterns_6H_CDL2CROWS', …., 'patterns_6H_CDLXSIDEGAP3METHODS', 'patterns_D_CDL2CROWS', …., 'patterns_D_CDLXSIDEGAP3METHODS']

2.5 Input Dimension Reduction

Dimension reduction / Features selection are important for FOUR reasons [9]:

- Simplify the model to make them understandable
- Reduce training times
- Avoid the curse of dimensionality
- Enable generalization by reducing overfitting

For each OHLC row, the number of features generated from the previous steps are as follows:

Type	Total Features
Basic Features (MinMaxScaled)	9
Technical Indicators	9
Candlestick Pattern Indicators	488
Total	**506**

It is impractical to train machine models with 506 features for each OHLC row due to machine resource restriction. In order to overcome this difficulty, we must apply some feature reduction techniques to select a subset of relatively important features.

In the following sections, only **"Technical Indicators"** and **"Candlestick Pattern Indicators"** are involved in the feature reduction process. Features are first filtered by **preliminary cleaning**. Finally, **random forest** is used to

evaluate the importance of each feature. Features which are not used by random forest in prediction would be discarded.

2.5.1 Preliminary Cleaning

Feature is discarded if it meets one of the following conditions:

1. All values in a column are constant

   For example, all zeros means no signal detected for all timesteps.

   Constant feature could be ignored because no relationship with the target prediction could be draw.

2. All values in a column duplicates to that of previous column with same index.

   i.e. Two columns having the same series of values (correlation 1)

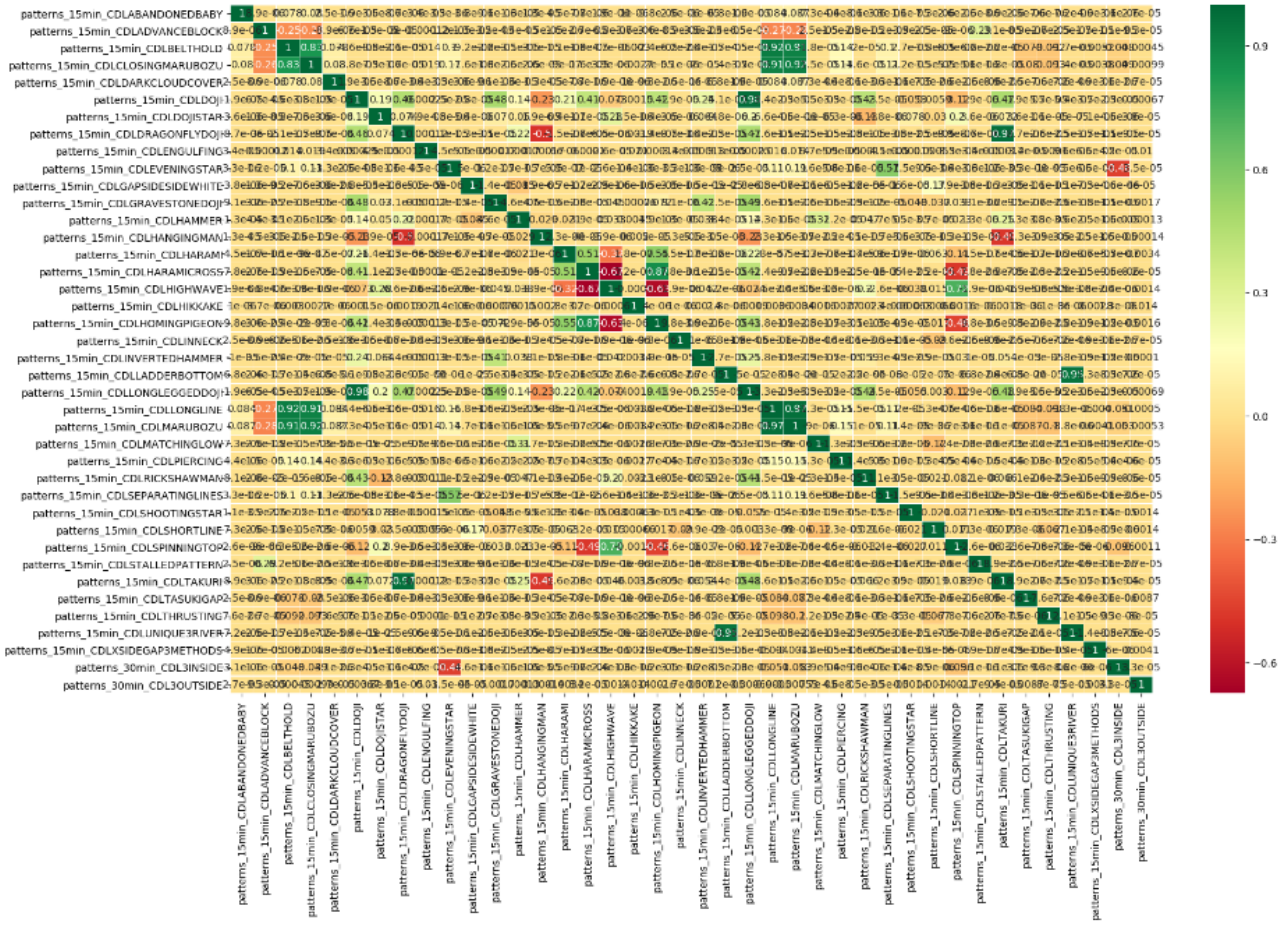


Figure 12 – Correlation Matrix of Subset of Features After Removing Duplicated Features (USDJPY)

2.5.2 Random Forest

The features after preliminary cleaning are then evaluated by random forest algorithm for selecting a subset of relatively important features.

The main idea of random forest is to construct a multitude of decision trees at training time. The more times a feature is being referenced in the decision trees, the more important the feature is [10].

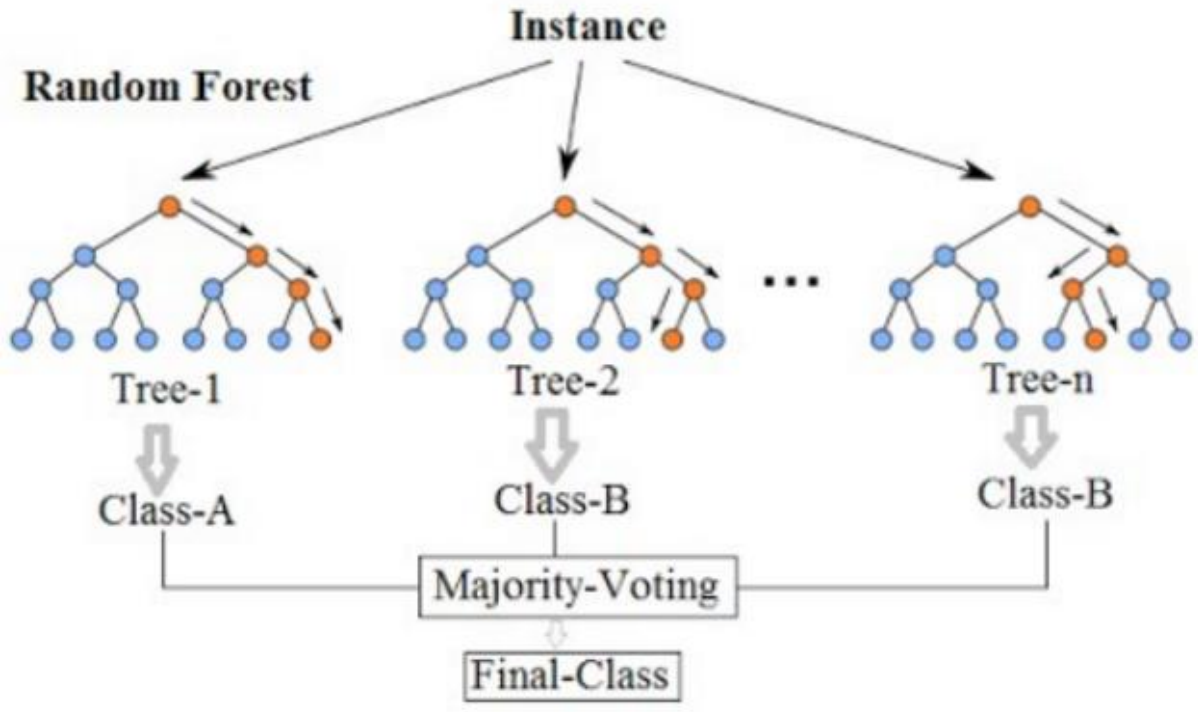


Figure 13 – Simplified Random Forest

2.5.2.1 Prediction Target

Training the random forest classification model requires pre-defined class label for each training sample.

The target class label for each timesteps are one of the followings:

Short(-1), Hold(0), Long(1)

The rule of defining the label is by forward-looking the actual result of a strategy.

In this dissertation, the following strategy would be used:

1. Forward-looking in the future 60 OHLC bars
2. Stop loss if a position loss more than x pips

   Label: **Short**
3. Take profit if earn more than x pips

   Label: **Long**
4. Otherwise, the final result at the last bar would be used to determine the class

$$Class = \begin{cases} 1 \; if \; the \; actual \; return \; after \; 60 \; bars > 0 \\ 0 \; if \; the \; actual \; return \; after \; 60 \; bars = 0 \\ -1 \; if \; the \; actual \; return \; after \; 60 \; bars < 0 \end{cases}$$

Example Label Output:

Timestamp	Short_Hold_Long
1	1
2	1
3	-1
…	…
…	…
t-1	0
t	1

Settings of each currency pair

	Take Profit PIP	Stop Loss PIP
USDJPY	400	100
USDCAD	20	5
AUDUSD	20	5
EURUSD	20	5

Note that the Profit & Loss Ratio is 4:1

2.5.2.2 Training Samples

The features (technical & pattern indicators) after filtering by preliminary cleaning in **section 2.5.1** would be used to construct the training samples

Timestep	**Input (X)**	**Short_Hold_Long (Y)** **[-1, 0, 1]**
T+0	[Filtered Technical Indicators, Filtered Patterns Indicators]	Class Label (T+0)
T+1	…	Class Label (T+1)
T+2	…	Class Label (T+2)
…	…	…
T+n	…	Class Label (T+n)

If there are **"n"** OHLC rows in the dataset, there will be **"n-1"** samples

The samples constructed are first **shuffled** in random order.

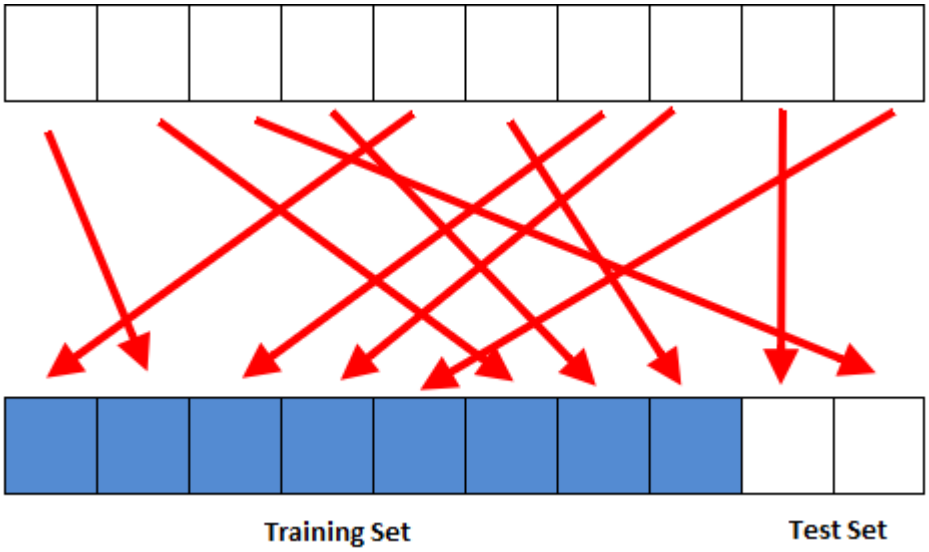


Figure – Data Shuffling and Splitting

Then, they are split into two sets by the ratio of **4:1**. **80%** of the samples are used for training the random forest, while **20%** of the samples are used for testing the model accuracy.

2.5.2.3 Hyper-parameters

The random forest algorithm from Scikit-learn is used [11].

RandomForestClassifier(*n_estimators=100, criterion='gini', max_depth= None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, m ax_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impur ity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=No ne, verbose=0, warm_start=False, class_weight=None*)

2.5.2.4 Statistic Result

	USDJPY	**USDCAD**	**AUDUSD**	**EURUSD**
Total Samples	1690662	602846	793537	1787246
Number of Long	520478 (30.78%)	138781 (23.02%)	109039 (13.74%)	231554 (12.96%)
Number of Short	513962 (30.40%)	137785 (22.86%)	107481 (13.54%)	226604 (12.68%)
Number of Hold	656222 (38.81%)	326280 (54.12%)	577017 (72.71%)	1329088 (74.37%)
Number of Samples in Training Set	1352529	482276	634829	1429796
Number of Samples in Test Set	338133	120570	158708	357450
10-KFold Validation Score	44.21%	55.59%	72.99%	74.81%

k-Fold Cross-Validation is a statistical method used to estimate the skill of machine learning models. The algorithm splits dataset into k consecutive folds. Each fold is then used once as a validation while the k - 1 remaining folds form the training set [12]. The higher the KFold score, the better the machine learning model.

The k-fold score of AUDUSD and EURUSD are pretty high due to high proportion of "Hold" which introduces bias. This statement is confirmed with the confusion matrix which compares the number of matches between the actual class label with the predicted class label in the test set.

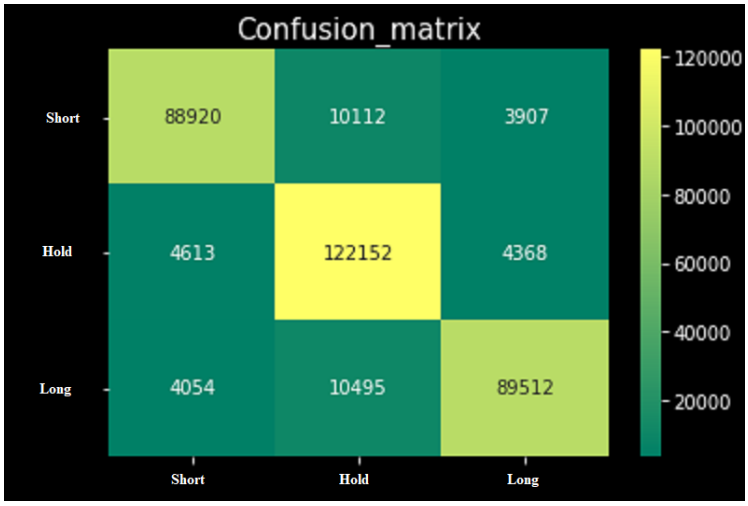


Figure 14 – Confusion Matrix
of Random Forest (USDJPY)

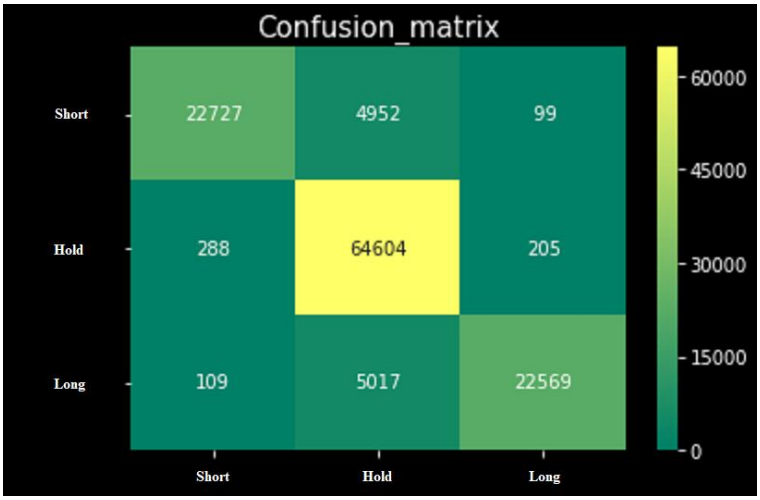


Figure 15 – Confusion Matrix
of Random Forest (USDCAD)

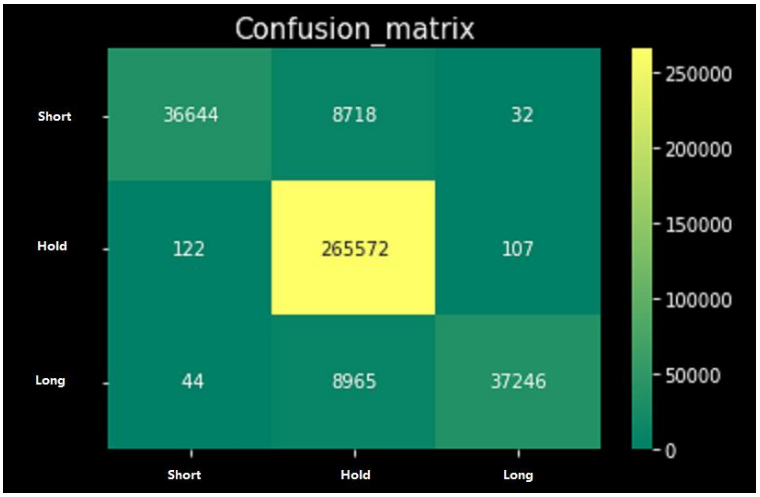


Figure 16 – Confusion Matrix
of Random Forest (EURUSD)

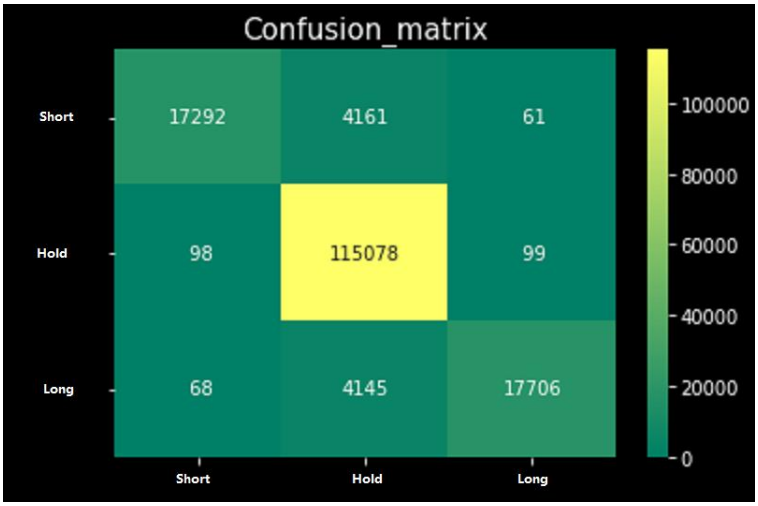


Figure 17 – Confusion Matrix
of Random Forest (AUDUSD)

The “Hold” class in the confusion matrix of EURUSD and AUDUSD are relatively higher than “Short” and “Long”. Although there is bias to “Hold”, the diagonal can still be observed from the matrix, such that there is prediction power of random forest.

2.5.2.5 Importance Matrix

After training the random forest on each currency pair, the importance matrix from each currency pair is extracted and used to determine which features are important in prediction.

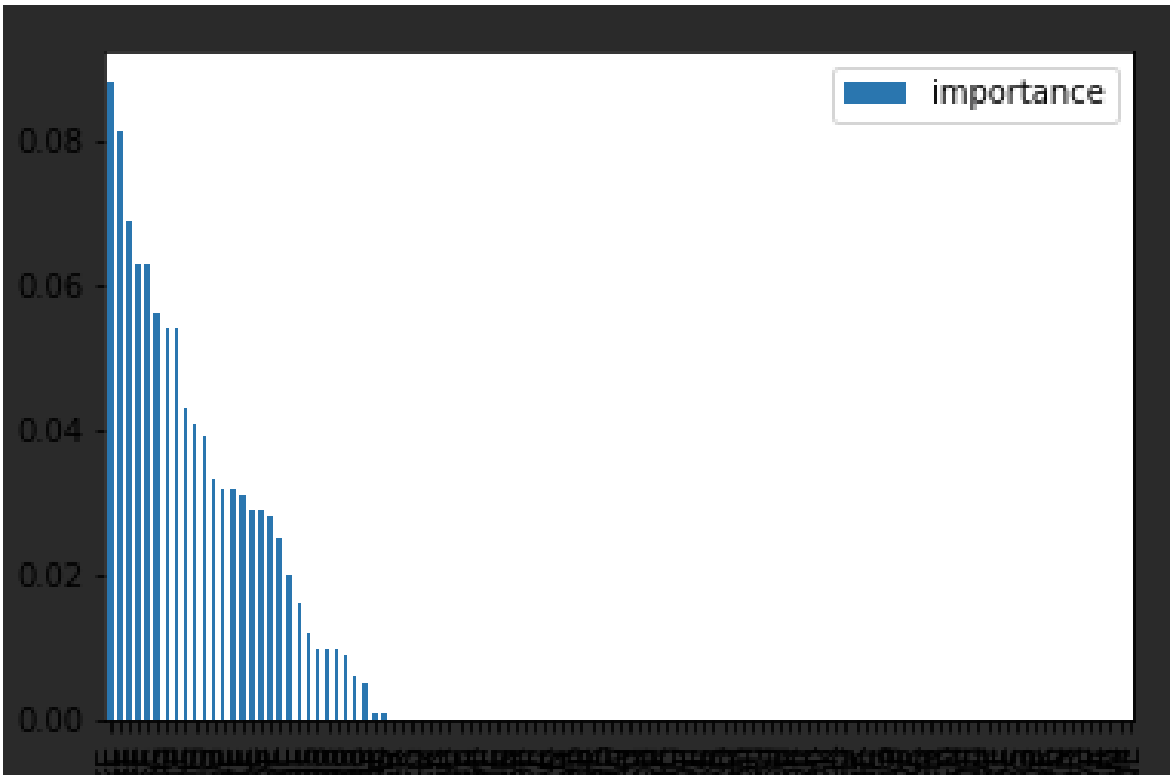


Figure 18 – Importance Matrix of USDJPY

Take the importance matrix of USDJPY as an example, x-axis is the features while y-axis is the importance. We could observe that only subset of the features is being used by the random forest. By using the importance matrix, we eliminate those unused features for subsequence machine learning models.

2.5.2.6 Summary

Note that only technical indicators and patterns indicators are involved in the feature reduction process. The followings are the final features after feature reduction process. Notes that the features are the columns for all OHLC rows in a dataset.

USDJPY:

Category	Features
Price	'MinMaxScaled_Time_Diff_Freq', 'MinMaxScaled_Open', 'MinMaxScaled_High', 'MinMaxScaled_Low', 'MinMaxScaled_Close', 'MinMaxScaled_Open-Close', 'MinMaxScaled_High-Low'
Price Action	'MinMaxScaled_Log_Return', 'MinMaxScaled_PIP_Return'
Technical Indicators	'technical_MACD_histogram', 'technical_RSI', 'technical_ADX', 'technical_DX', 'technical_STOCH_slowd', 'technical_STOCH_slowk', 'technical_STOCHF_fastk'
Patterns Indicators	'patterns_1H_CDLHIKKAKE', 'patterns_3H_CDLHIKKAKE', 'patterns_30min_CDLHIKKAKE', 'patterns_15min_CDLHIKKAKE', 'patterns_5min_CDLHIKKAKE', 'patterns_6H_CDLHIKKAKE', 'patterns_1min_CDLHIKKAKE'

USDCAD:

Category	Features
Price	'MinMaxScaled_Time_Diff_Freq', 'MinMaxScaled_Open', 'MinMaxScaled_High', 'MinMaxScaled_Low', 'MinMaxScaled_Close', 'MinMaxScaled_Open-Close', 'MinMaxScaled_High-Low'
Price Action	'MinMaxScaled_Log_Return', 'MinMaxScaled_PIP_Return'
Technical Indicators	'technical_MACD_histogram', 'technical_ADX', 'technical_RSI', 'technical_DX', 'technical_STOCH_slowd', 'technical_STOCH_slowk', 'technical_STOCHF_fastk'
Patterns Indicators	'patterns_30min_CDLHIKKAKE', 'patterns_1H_CDLHIKKAKE', 'patterns_15min_CDLHIKKAKE', 'patterns_5min_CDLHIKKAKE', 'patterns_3H_CDLHIKKAKE', 'patterns_6H_CDLHIKKAKE', 'patterns_1min_CDLHIKKAKE'

AUDUSD:

Category	Features
Price	'MinMaxScaled_Time_Diff_Freq', 'MinMaxScaled_Open', 'MinMaxScaled_High', 'MinMaxScaled_Low', 'MinMaxScaled_Close', 'MinMaxScaled_Open-Close', 'MinMaxScaled_High-Low'
Price Action	'MinMaxScaled_Log_Return', 'MinMaxScaled_PIP_Return'
Technical Indicators	'technical_MACD_histogram', 'technical_STOCH_slowd', 'technical_RSI', 'technical_STOCH_slowk', 'technical_ADX', 'technical_STOCHF_fastk', 'technical_DX'
Patterns Indicators	'patterns_3H_CDLHIKKAKE', 'patterns_6H_CDLHIKKAKE', 'patterns_15min_CDLHIKKAKE', 'patterns_1H_CDLHIKKAKE', 'patterns_30min_CDLHIKKAKE', 'patterns_5min_CDLHIKKAKE', 'patterns_1min_CDLHIKKAKE', 'patterns_15min_CDLENGULFING', 'patterns_D_CDLENGULFING', 'patterns_3H_CDLENGULFING'

EURUSD:

Category	Features
Price	'MinMaxScaled_Time_Diff_Freq', 'MinMaxScaled_Open', 'MinMaxScaled_High', 'MinMaxScaled_Low', 'MinMaxScaled_Close', 'MinMaxScaled_Open-Close', 'MinMaxScaled_High-Low'
Price Action	'MinMaxScaled_Log_Return', 'MinMaxScaled_PIP_Return'
Technical Indicators	'technical_MACD_histogram', 'technical_RSI', 'technical_ADX', 'technical_STOCH_slowd', 'technical_STOCH_slowk', 'technical_DX', 'technical_STOCHF_fastk'
Patterns Indicators	'patterns_30min_CDLHIKKAKE', 'patterns_15min_CDLHIKKAKE', 'patterns_3H_CDLHIKKAKE', 'patterns_1H_CDLHIKKAKE', 'patterns_6H_CDLHIKKAKE', 'patterns_5min_CDLHIKKAKE', 'patterns_1min_CDLHIKKAKE', 'patterns_1H_CDLENGULFING'

For technical indicators, ‘technical_MACD_short_hold_long’ and ‘technical_STOCHF_fastd’ are removed from technical indicators for all currency pairs.

For pattern indicators, only ‘CDLHIKKAKE’ and ‘CDLENGULFING’ remain.

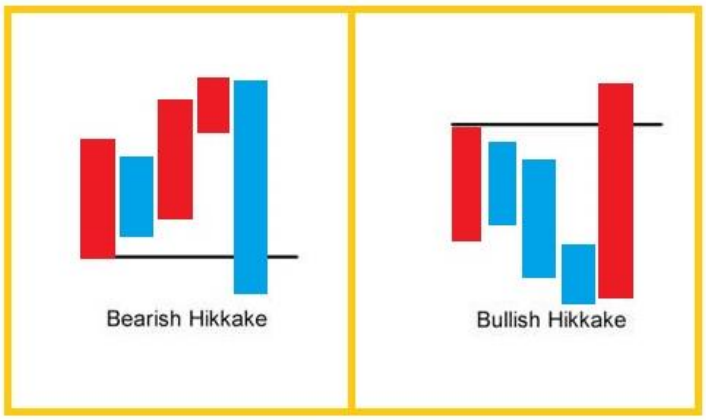


Figure 19 – Hikkake Pattern (CDLHIKKAKE)

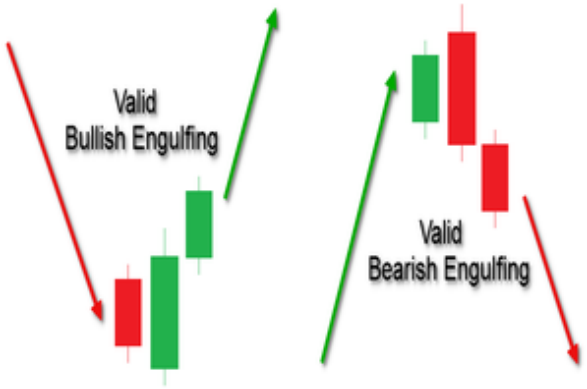


Figure 20 – Engulfing Pattern (CDLENGULFING)

The reasons of why these two patterns are important requires future task for study. In this thesis, the reduced set of features would be used for training our machine learning models.

Chapter 3: Model Evaluation

In the first section, the following commonly used classification models would be tried and evaluated on **USDJPY**:

Classification Models
K-nearest Neighbours (KNN)
Support Vector Machines (SVM)
Logistic Regression
Gaussian Naïve Bayes
Perceptron

From the analysis of the classification models on **USDJPY**, we could understand that the traditional classification models do not work well in time-series data in our setup.

In the second section of this chapter, a composite neural network (**Bi-LSTM + BERT**) would be introduced and evaluated. LSTM is a very popular neural network used in sequence-to-sequence analysis, while BERT is the latest (2018/19) network published by GOOGLE for Natural Language Processing (NLP) tasks. Natural Language Processing (NLP) tasks also involve sequence analysis because a paragraph is basically a sequence of words.

The Forex data is essentially a sequence of vectors. For each timestep, multiple features (columns) form a vector. Vectors from multiple timesteps form a sequence. Therefore, we could apply similar sequence-to-sequence techniques for future price movement prediction.

3.1 Classification Models

3.1.1 Models

For each classification model, the high-level summary and parameters would be included. As the focus of this dissertation would be put in the neural network of time series prediction, the detailed description of each classification model would be skipped.

The classification model is based on the implementation of scikit-learn (https://scikit-learn.org). The corresponding description for each parameter could be found on the API documentations as indicated on the List of References.

3.1.1.1 K-nearest Neighbours (KNN)

KNN separates data into specific number of clusters by iteratively calculating the distance between data points [13].

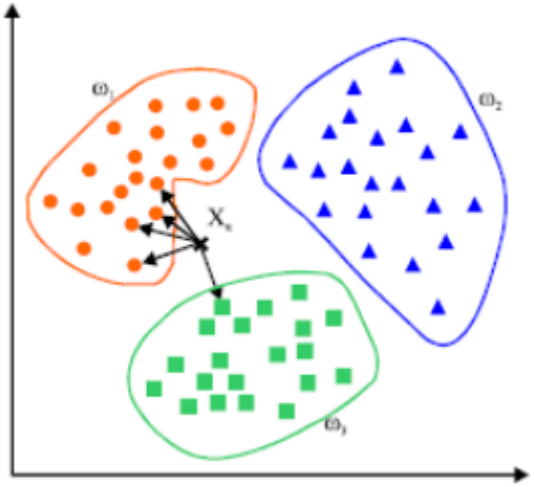


Figure 21 - K-nearest Neighbours (KNN)

KNeighborsClassifier(*n_neighbors=3, weights='uniform', algorithm='auto' , leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*)

3.1.1.2 Support Vector Machines (SVM)

SVM helps to classify data into groups by applying a linear plane onto the vectors space. It can also perform non-linear classification using kernel trick [14].

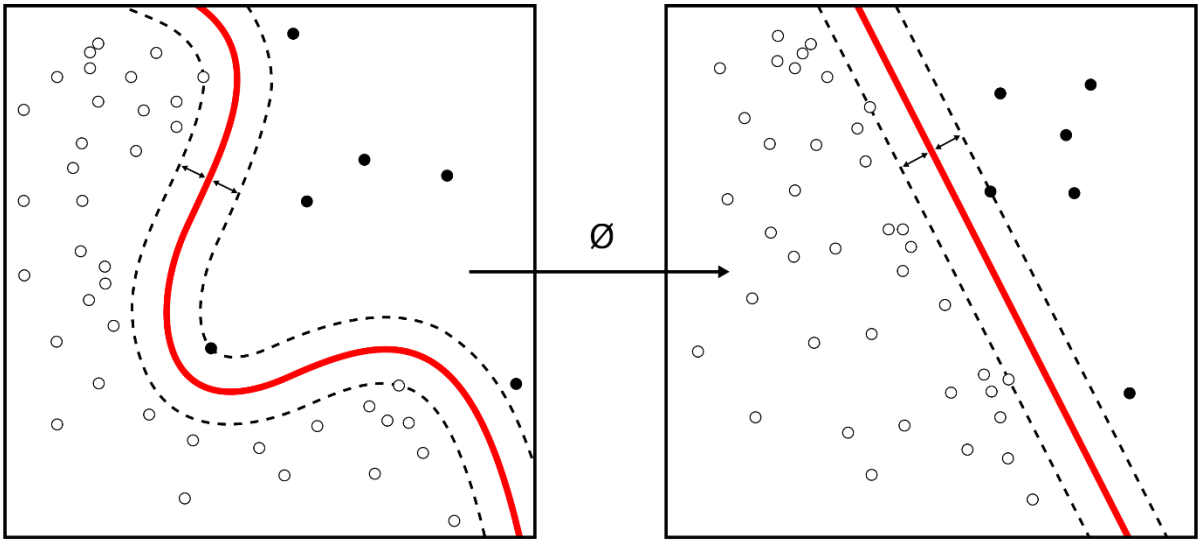


Figure 22 - Support Vector Machines (SVM)

LinearSVC(*penalty='l2', loss='squared_hinge', dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000*)

3.1.1.3 Logistic Regression

It helps classifying data into different categories. Its output is a finite set of discrete variables [15].

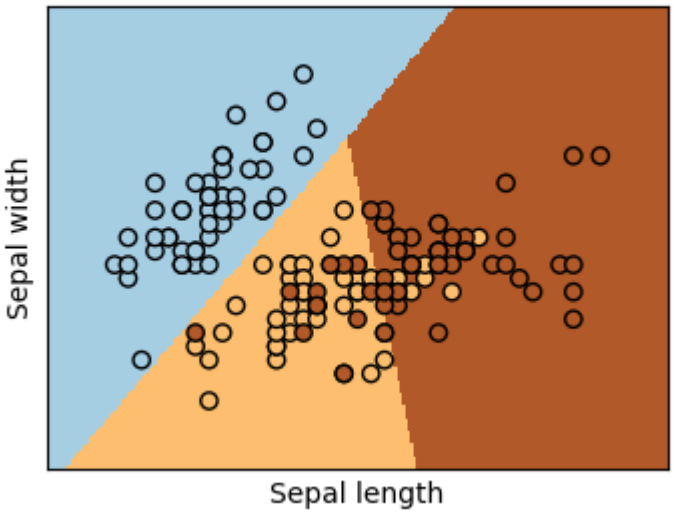


Figure 23 - Logistic Regression

LogisticRegression(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='warn', max_iter=100, multi_class='warn', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)

3.1.1.4 Gaussian Naïve Bayes

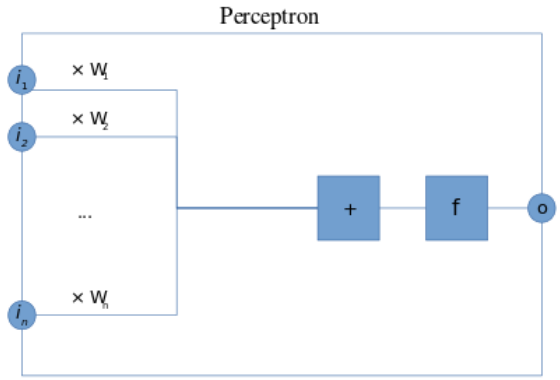
Naïve Bayes is one of the probabilistic classifiers by using the observed probability in the input data [16].

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \tag{3.1}$$

GaussianNB(*priors=None, var_smoothing=1e-09*)

3.1.1.5 Perceptron

Perceptron is an algorithm for supervised learning of binary classification. Appropriate weights are assigned to inputs, and another function is used to generate output from the weighted sum of inputs [17].



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right) \tag{3.2}$$

Figure 24 - Perceptron

Perceptron(*penalty=None, alpha=0.0001, fit_intercept=True, max_iter=5, tol=0.001, shuffle=True, verbose=0, eta0=1.0, n_jobs=None, random_state=0, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False*)

3.1.2 Input and Prediction Target

The subset of features (X) after feature reduction step (**section 2.5.2.5**) would be used, while class label generated in **section 2.5.2.1** would be used as the target prediction (Y).

Timestep	**Vectors (X)**	**Short_Hold_Long (Y) [-1, 0, 1]**
T+0	[Price, Price Action, Technical Indicators, Patterns Indicators]	Class Label (T+0)
T+1	…	Class Label (T+1)
T+2	…	Class Label (T+2)
…	…	…
T+n	…	Class Label (T+n)

According to section 2.5.2.6, the numbers of features per timestep for each currency pair are as follows:

	Number of Features Per Timestep
USDJPY	23
USDCAD	23
AUDUSD	26
EURUSD	24

Multiple features in a timestep form a vector, such that the number of features is the dimension of each vector.

$$Vector_i = [Feature_1(i), Feature_2(i), \dots, Feature_n(i)]$$

where n is the vector dimension (aka: number of features)

$$f(Vector_i) = ClassLabel_{i+1}$$

where i is the OHLC row index, and f is the prediction function of machine learning model

Similar to training the random forest in feature reduction steps, **"n-1"** samples are constructed from **"n"** 1000-BidVolume-Resampled OHLC rows.

3.1.3 Models Evaluation

Model	Short (-1) Hold (0) Long (1)	Precision	Recall	F1-score
K-nearest Neighbours (KNN)	-1	0.33	0.48	0.39
	0	0.42	0.35	0.38
	1	0.36	0.28	0.31
Support Vector Machines (SVM)	-1	0.35	0.03	0.05
	0	0.38	**0.93**	**0.54**
	1	0.38	0.07	0.11
Logistic Regression	-1	0.36	0.03	0.06
	0	0.38	**0.93**	**0.54**
	1	0.38	0.07	0.11
Gaussian Naïve Bayes	-1	0.36	0.20	0.26
	0	0.41	**0.70**	**0.52**
	1	0.37	0.22	0.27
Perceptron	-1	0.30	0.17	0.22
	0	0.37	0.33	0.35
	1	0.31	0.48	0.38

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \tag{3.3}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \tag{3.4}$$

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{3.5}$$

From the above result evaluated on **USDJPY**, it only shows better performance in predicting “Hold” (0). However, “Hold” class occupies the

most portion of the entire dataset, i.e. 38.81% according to section 2.5.2.3. Thus, the dataset would have bias towards “Hold”.

For “Short” and “Long”, none of the above models show convincing accuracy(F1-score) based on the features we have constructed.

3.2 Regression Neural Network

3.2.1 Models

In following section, Bi-LSTM (**Bi**directional **L**ong **S**hort-**T**erm **M**emory) and the self-attention mechanism from BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) would be described. In our final model, Bi-LSTM would be acted as encoder and integrated with the self-attention network from BERT

3.2.1.1 Bidirectional Long Short-Term Memory (Bi-LSTM)

Bi-LSTM is an enhancement to the LSTM model which is a commonly used machine learning model for time series prediction. Unlike feedforward neural networks, LSTM is a type of recurrent neural network (RNN) architecture which makes use of feedback connections for processing sequence of data. It has been widely used to classify, process and make predictions based on time series data [18][19].

Time series data may have lags of unknown duration between events which introduce difficulties in traditional RNN, hidden Markov models and other sequence learning models which are sensitive to gap length [18]. Compared with traditional RNN, LSTM consists of extra gates to allow the network to learn the long-term dependency between elements in a series.

A common LSTM block is composed of a cell and three gates (i.e. cell, input gate, output gate, forget gate). Basically, a cell is responsible for learning the dependencies between the elements in the input series. The **input gate** controls the extent to which a new value flows into the cell, the **forget gate** controls the extent to which a value remains in the cell and the **output gate** controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM gates is often the logistic function. [18][19]

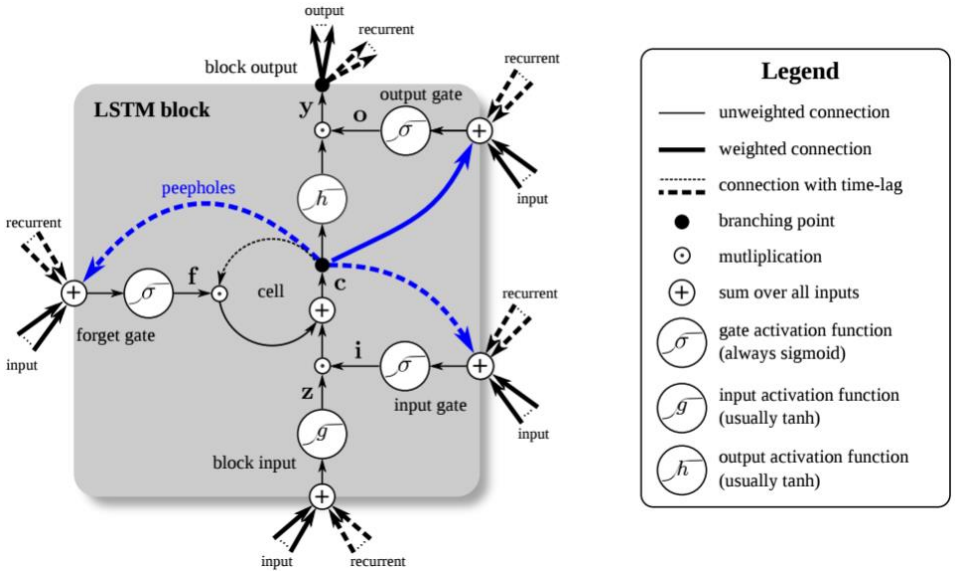


Figure 25 – LSTM Block

A LSTM chain consists of multiple LSTM blocks to evaluate and learn a series of data from left-to-right.

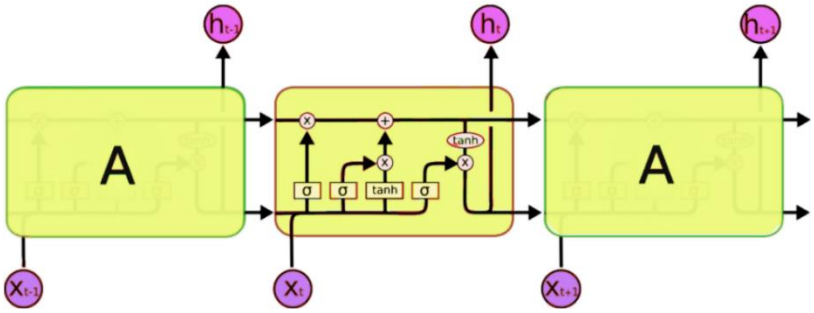


Figure 26 – LSTM Chain

However, it restricts the network from accessing information on the right, such that the state of the LSTM block only depends on the input and previous block. Very often, the context of an element in a series depends on both left and right.

To address the limitation of LSTM, Bi-LSTM is introduced. The series of data is encoded from **Left-to-Right** and also **Right-to-Left** in order to preserve more sequential relationships. It connects two hidden layers of opposite directions to the same output. With this form of generative deep learning, the output layer can get information from past (backwards) and future (forward) states simultaneously. [20][21]

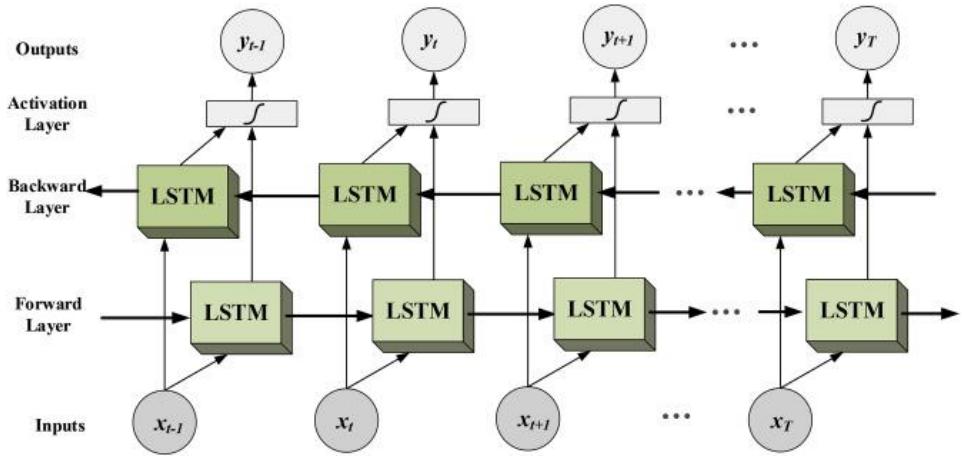


Figure 27 – Bi-LSTM

3.2.1.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT, published by google in Oct, 2018, is a model designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context in all layers. It has obtained state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks such as question answering and language inference, without substantial task-specific architecture modifications [22].

BERT makes use of self-attention which has been adopted successfully in a variety of tasks such as reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [23, 24, 25, 26, 27]. Self-attention is an intra-attention mechanism for relating different positions of a single sequence in order to compute a representation of the sequence [22].

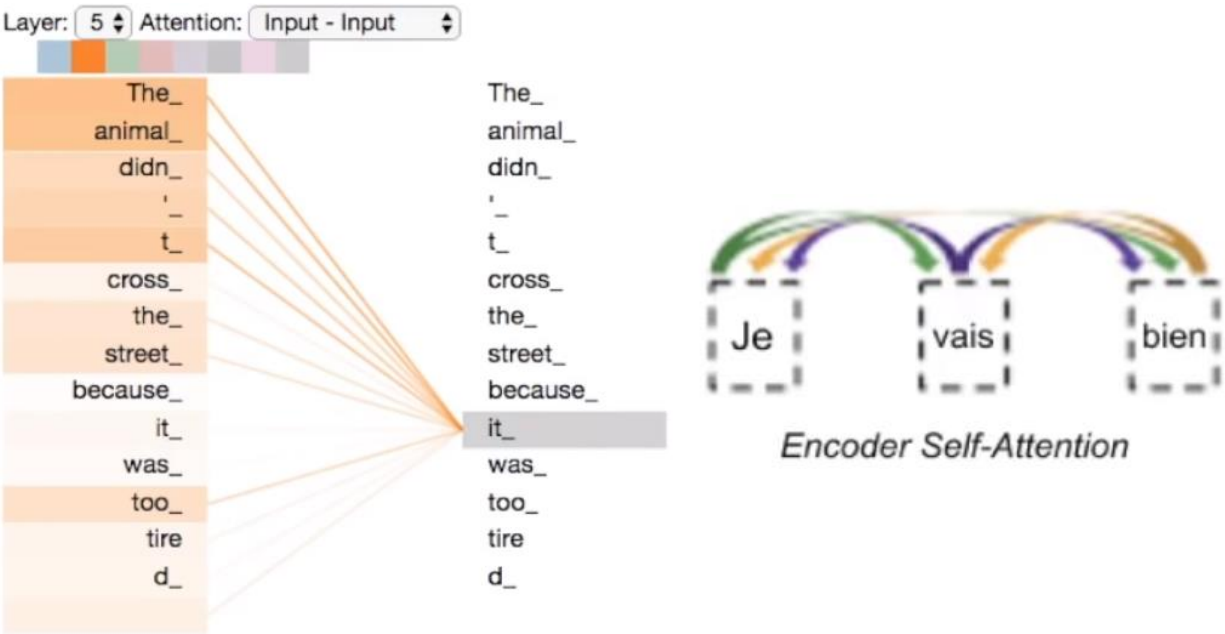


Figure 28 – Examples of Self-attention

3.2.1.3 Composite Model (Bi-LSTM + BERT)

In the dissertation, we make use of two layers of bidirectional LSTM as the input sequence encoder, and pass the encoded result to the self-attention layer extracted from BERT. Finally, the output from attention layer would be condensed into our prediction target, i.e. log return of the next timestep.

In this dissertation, **Tensorflow** is used for constructing our composite model. Tensorflow is a mature and popular machine learning framework to facilitate the development of machine learning models.

```
def build_model(inputShape):
    inp = Input(shape = inputShape)

    # LSTM before attention layers
    x = Bidirectional(LSTM(200, return_sequences=True))(inp)
    x = Bidirectional(LSTM(64, return_sequences=True))(x)

    x, slf_attn = MultiHeadAttention(n_head=3, d_model=300, d_k=64, d_v=64, dropout=0.1, mode=1)(x, x, x)

    avg_pool = GlobalAveragePooling1D()(x)
    max_pool = GlobalMaxPooling1D()(x)
    conc = concatenate([avg_pool, max_pool])
    conc = max_out(conc, 60)
    x = Dense(FUTURE_PERIOD_PREDICT, activation="sigmoid")(conc)

    model = Model(inputs = inp, outputs = x)
    model.compile(loss = "mean_squared_error", optimizer = "adam")

    return model
```


Figure 29 – Code Snippet of Composite Model (Bi-LSTM+BERT)

Take **USDJPY** as an example, the input(X) for each training sample is a 600x23 matrix.

where 600 is the number of past timesteps (**section 3.2.2**), and 23 is the number of features (**section 2.5.2.6**) in each timestep.

i.e. 600 vectors in a series, and the dimension of each vector is 23

The series of 600 vectors (23 dimensions) are then encoded with two Bi-LSTM. The encoded output from Bi-LSTMs is then processed by the self-attention layer from BERT. Finally, the output from self-attention layer is condensed into 1 value, i.e. Log Return of the next timestep.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 600, 23)]	0	
bidirectional (Bidirectional)	(None, 600, 400)	360000	input_1[0][0]
bidirectional_1 (Bidirectional)	(None, 600, 128)	238592	bidirectional[0][0]
time_distributed (TimeDistribut	(None, 600, 64)	8192	bidirectional_1[0][0]
time_distributed_1 (TimeDistrib	(None, 600, 64)	8192	bidirectional_1[0][0]
time_distributed_3 (TimeDistrib	(None, 600, 64)	8192	bidirectional_1[0][0]
time_distributed_4 (TimeDistrib	(None, 600, 64)	8192	bidirectional_1[0][0]
time_distributed_6 (TimeDistrib	(None, 600, 64)	8192	bidirectional_1[0][0]
time_distributed_7 (TimeDistrib	(None, 600, 64)	8192	bidirectional_1[0][0]
lambda (Lambda)	(None, 600, 600)	0	time_distributed[0][0] time_distributed_1[0][0]
lambda_2 (Lambda)	(None, 600, 600)	0	time_distributed_3[0][0] time_distributed_4[0][0]
lambda_4 (Lambda)	(None, 600, 600)	0	time_distributed_6[0][0] time_distributed_7[0][0]
activation (Activation)	(None, 600, 600)	0	lambda[0][0]
activation_1 (Activation)	(None, 600, 600)	0	lambda_2[0][0]
activation_2 (Activation)	(None, 600, 600)	0	lambda_4[0][0]
dropout (Dropout)	(None, 600, 600)	0	activation[0][0] activation_1[0][0] activation_2[0][0]
time_distributed_2 (TimeDistrib	(None, 600, 64)	8192	bidirectional_1[0][0]
time_distributed_5 (TimeDistrib	(None, 600, 64)	8192	bidirectional_1[0][0]
time_distributed_8 (TimeDistrib	(None, 600, 64)	8192	bidirectional_1[0][0]
lambda_1 (Lambda)	(None, 600, 64)	0	dropout[0][0] time_distributed_2[0][0]
lambda_3 (Lambda)	(None, 600, 64)	0	dropout[1][0] time_distributed_5[0][0]
lambda_5 (Lambda)	(None, 600, 64)	0	dropout[2][0] time_distributed_8[0][0]
concatenate (Concatenate)	(None, 600, 192)	0	lambda_1[0][0] lambda_3[0][0] lambda_5[0][0]
time_distributed_9 (TimeDistrib	(None, 600, 300)	57900	concatenate[0][0]
dropout_1 (Dropout)	(None, 600, 300)	0	time_distributed_9[0][0]
layer_normalization (LayerNorma	(None, 600, 300)	600	dropout_1[0][0]
global_average_poolingld (Globa	(None, 300)	0	layer_normalization[0][0]
global_max_poolingld (GlobalMax	(None, 300)	0	layer_normalization[0][0]
concatenate_2 (Concatenate)	(None, 600)	0	global_average_poolingld[0][0] global_max_poolingld[0][0]
Reshape (TensorFlowOpLayer)	[(None, 60, 10)]	0	concatenate_2[0][0]
Max (TensorFlowOpLayer)	[(None, 60)]	0	Reshape[0][0]
dense_10 (Dense)	(None, 1)	61	Max[0][0]

Total params: 730,881
Trainable params: 730,881
Non-trainable params: 0

Figure 30 – Composite Model Summary (USDJPY)

3.2.2 Input and Prediction Target

Similar to training the classification models, the subset of features (X) after feature reduction step (**section 2.5.2.6**) would be used. However, a series of past **600 vectors** would be used for prediction instead of just one vector in the past. Moreover, the prediction target (Y) is changed to the Log Return in the next timestep instead of using class label.

Timestep	Vectors (X)	Log Return (Y) (Section 2.4.1)
T+0	[Price, Price Action, Technical Indicators, Patterns Indicators]	R(T)
T+1	…	R(T+1)
…	…	R(…)
T+599	…	R(T+599)
T+600	…	R(T+600)
T+601	…	R(T+601)
…	…	…

If there are **"n"** timesteps in the dataset, **"n-600"** samples would be constructed.

$$Vector_i = [Feature_1(i), Feature_2(i), \dots, Feature_n(i)]$$

where n is the vector dimension (aka: number of features)

$$f([Vector_i, Vector_{i+1}, Vector_{i+2}, \dots, Vector_{i+599}]) = LogReturn_{i+600}$$

where i is the row index of each timestep, and f is the prediction function of machine learning model

3.2.3 Model Evaluation & Comparison

The data period of each currency pair is from **01-Jan-2005** to **18-May-2019**.

The samples constructed in **section 3.2.2** are split into two sets using the timestamp of the prediction target (Log Return):

	From	To
Training Set	01-Jan-2005	30-Jun-2018
Test Set	01-Jul-2018	18-May-2019

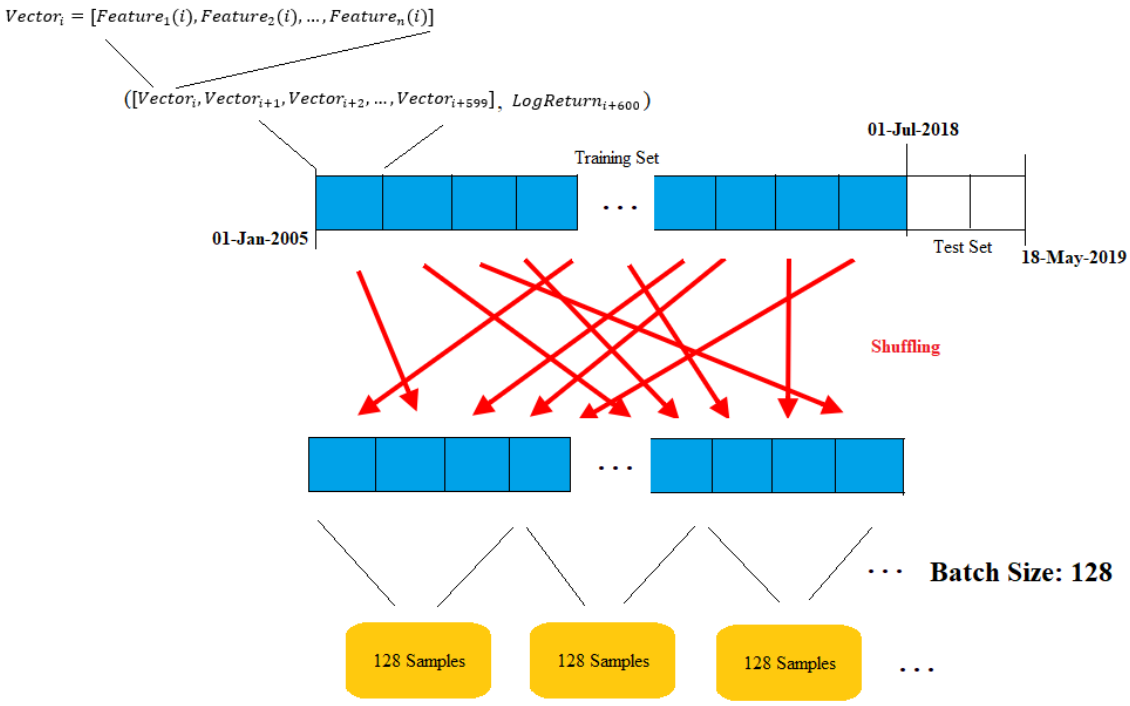


Figure 31 – Illustration of Processing Training Samples

The training set is first shuffled in random order. After that, the shuffled samples are split into batches per 128 samples. Batch size refers to the number of training examples utilized in one iteration. Essentially, the gradient and the neural network parameters are updated after each batch.

Batches are then feed into the model in **random order** to avoid bias towards batches with larger index. If we train the model by feeding the batches sequentially, the model would bias towards later batches.

During training, the model is trained for 5 epochs on the batches of training set.

$$1\ Epoch = Total\ Number\ of\ Batches$$

For example, if the training set has 100 batches, then 1 epoch iteration would iterate 100 batches. After each epoch training, **test set** (01-Jul-2018 to 18-May-2019) would be used for back-testing using following approach:

$$StrategySignal(t) = \begin{cases} +1\ if\ predicted\log return\ > 0 \\ 0\ if\ predicted\log return = 0 \\ -1\ if\ predicted\log return < 0 \end{cases}$$

$$StrategyLogReturn(t) = ActualLogReturn(t)\ \times StrategySignal(t)$$

$$AccumulatedLogReturn(t) = \sum_{i=0}^{t} StrategyLogReturn(i) \qquad (3.6)$$

Illustration of Accumulated Return Calculation

	Log Return	Strategy Signal	Strategy Log Return	Accumulated Log Return
T = 0	+0.01%	- 1	- 0.01%	- 0.01%
T = 1	- 0.01%	+1	- 0.01%	- 0.02%
T = 2	- 0.01%	- 1	+0.01%	- 0.01%
T = 3	+ 0.02%	+1	+0.02%	+0.01%
T = 4	+ 0.05%	+1	+0.05%	+0.06%
…	….	..	…	…

Back-Testing Results on Test Set (**01-Jul-2018** to **18-May-2019**):

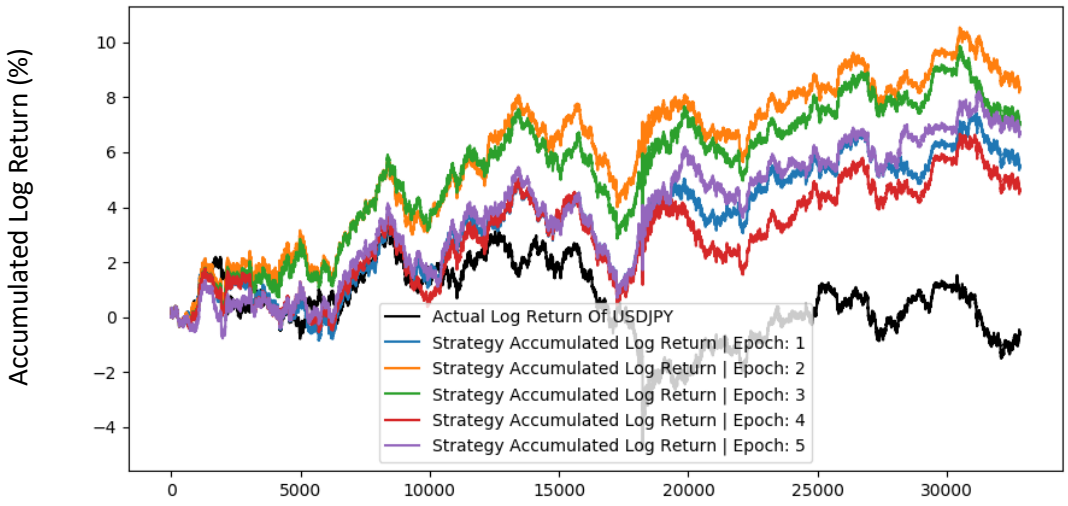


Figure 32 – USDJPY Back-Testing Result

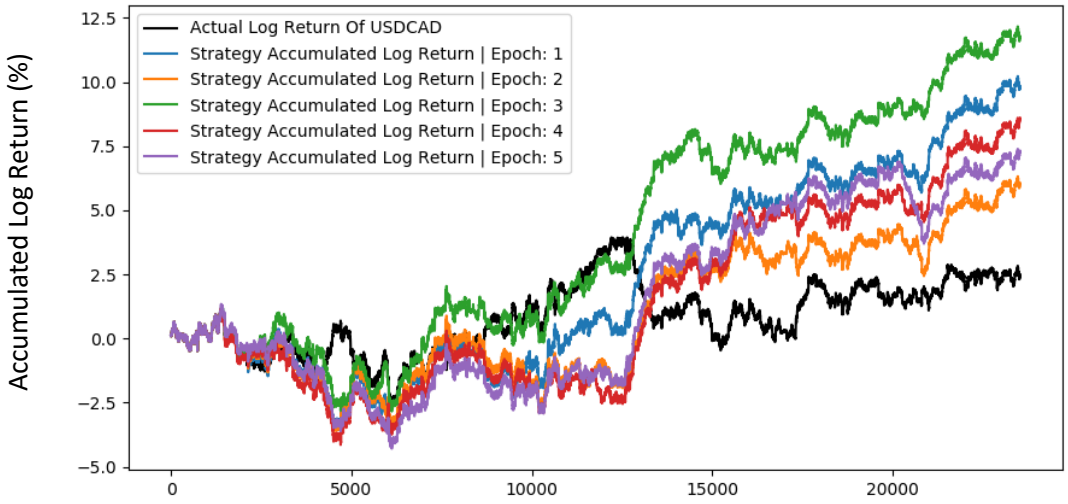


Figure 33 – USDCAD Back-Testing Result

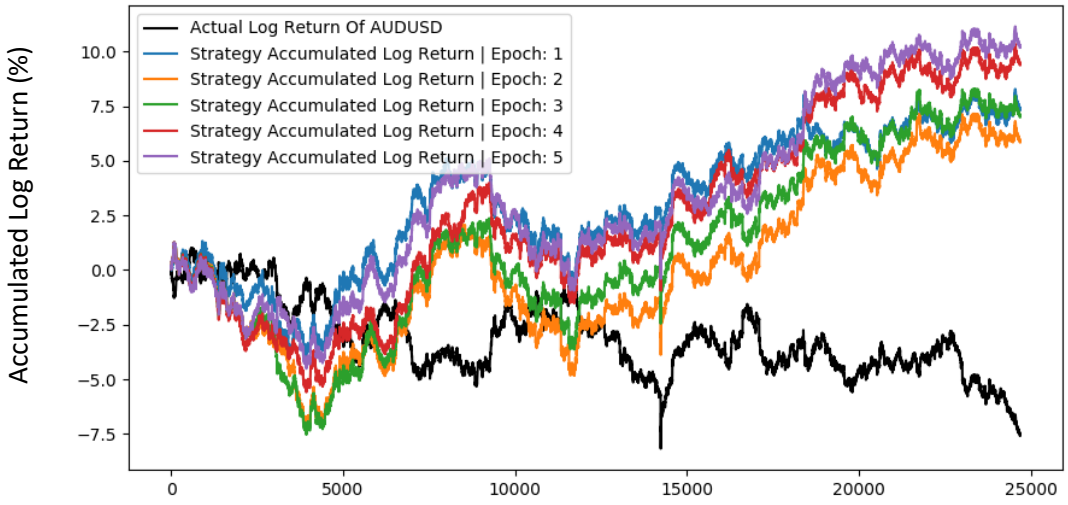


Figure 34 – AUDUSD Back-Testing Result

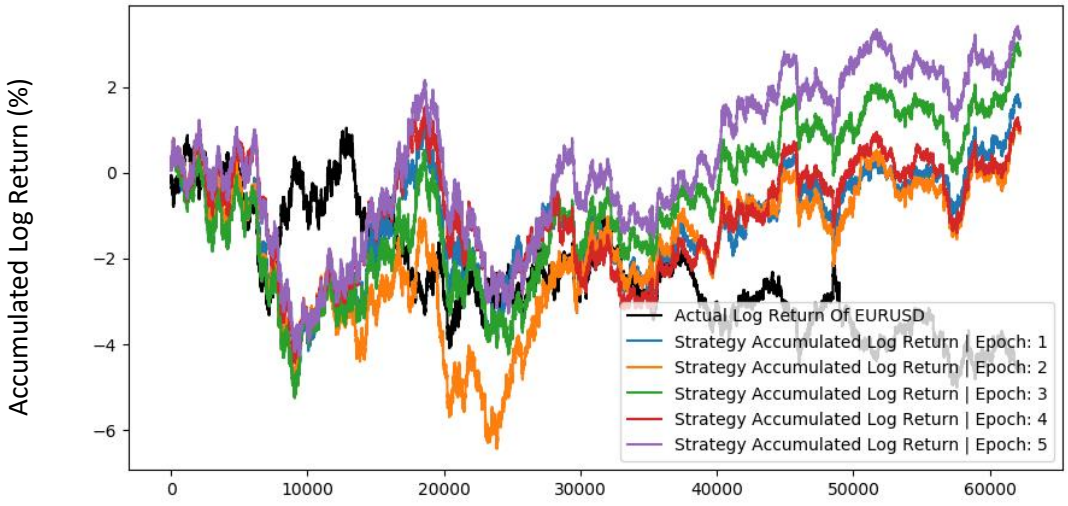


Figure 35 – EURUSD Back-Testing Result

Accumulated Return Summary:

	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
USDJPY	5.485%	**8.305%**	7.039%	4.617%	6.690%
USDCAD	9.850%	6.072%	**11.795%**	8.608%	7.323%
AUDUSD	7.323%	5.857%	7.012%	9.393%	**10.189%**
EURUSD	1.543%	0.931%	2.747%	0.999%	**3.132%**

Green colour indicates the highest value in a currency pair

From the result above, we can observe that the trend of back-testing returns varies with the number of epoch iterations.

For USDJPY, the return starts dropping after epoch 2.

For USDCAD, the return drops sharply (~3%) after epoch 3.

For AUDUSD and EURUSD, the highest return is at epoch 5

The dropping in the accumulated return for USDJPY and USDCAD suggests there might have overfitting. The model is trained with past samples for too many times, such that the model could not adapt to future variations in market environment.

Since the model does not take any samples from test set for training, it could not learn any new thing occurred in the test set. In order to overcome this limitation, the test set is predicted and trained **sequentially** batch by batch (batch size: 128).

```
predictions = []
While (remainBatches.length > 0):
        // Extract a batch with 128 samples in the front
        batchWith128Samples = remainBatches.pop(128)

        // Make predictions on the batch
        predictions.append( predict(batchWith128Samples) )

        train(batchWith128Samples)
```

In order to reduce the bias towards recent samples, the following approaches are used:

Function **train**(batchWith128Samples):

1. Get the recent 75 batches before "batchWith128Samples" (inclusively)
2. **Randomly** select 25 batches from **training set**
   i.e. the ratio of the number of recent batches to that of the past is **3:1**
3. Shuffle the 100 batches (75+25) randomly
4. Select 10 batches from the 100 batches for training the model

After iteratively predict and train on test set, the weights of model would be **restored** to original one which just trained on the training set, such that the weights trained by iterative training approach on each epoch would be

discarded to ensure the model does not have any information on test set before starting next epoch.

In short, the objective is to let the model train on the recent samples in the past for each epoch, such that the model could adapt to the changes in market environment.

Back-Testing Results on Test Set using **Iterative Training** approach:

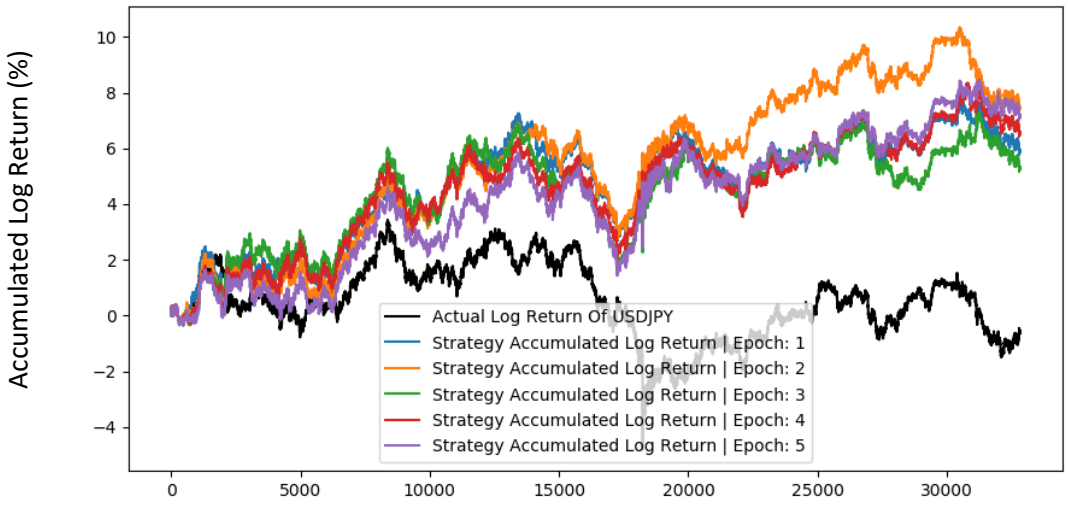


Figure 36 – USDJPY Back-testing Result (Iterative Training)

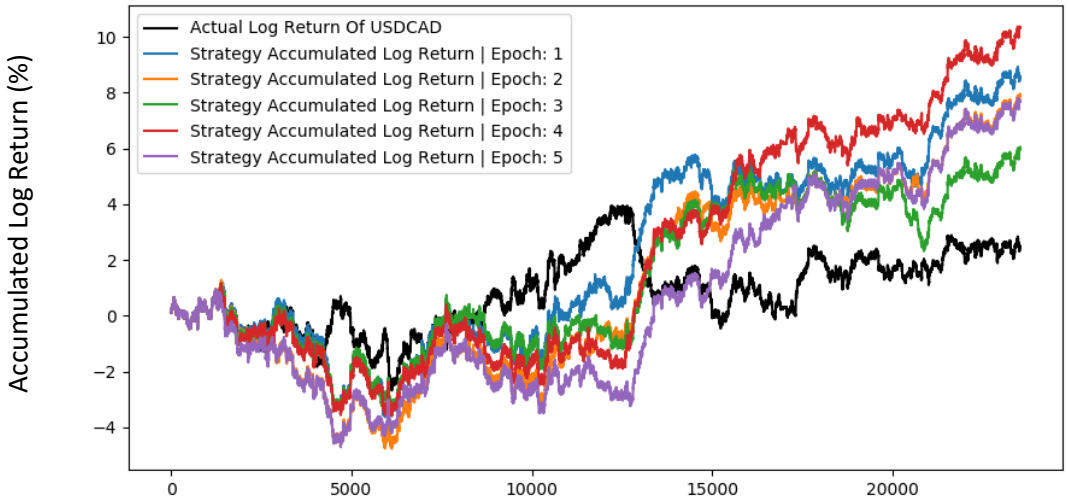


Figure 37 – USDCAD Back-testing Result (Iterative Training)

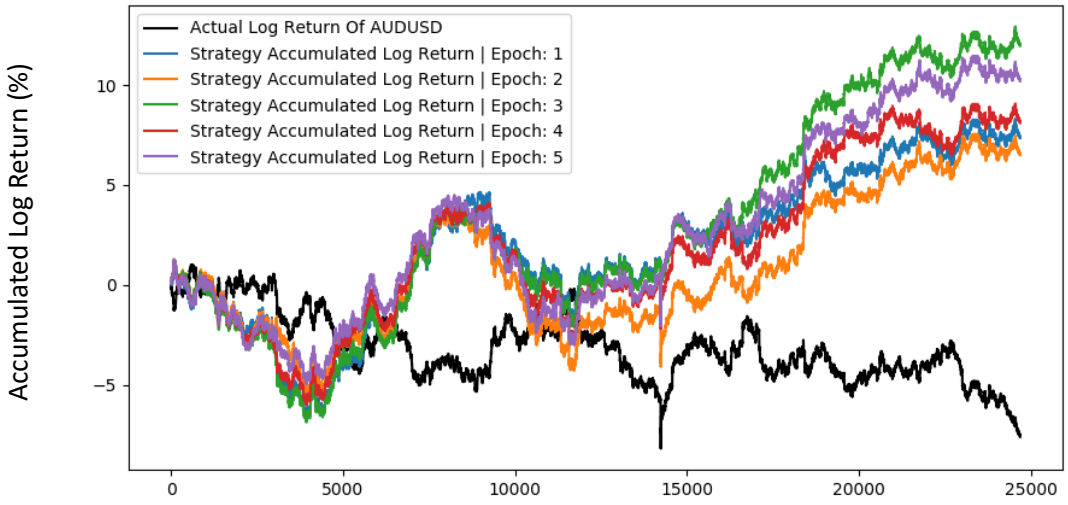


Figure 38 – AUDUSD Back-testing Result (Iterative Training)

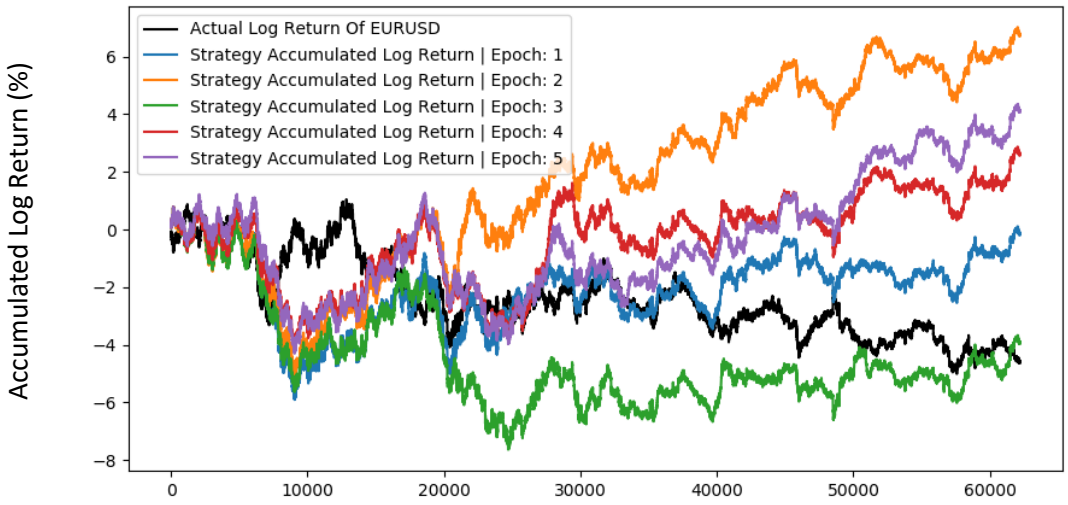


Figure 39 –EURUSD Back-testing Result (Iterative Training)

Accumulated Return Summary:

	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5	**Average**
USDJPY	5.485%	**8.305%**	7.039%	4.617%	6.690%	6.43%
USDCAD	9.850%	6.072%	**11.795%**	8.608%	7.323%	8.73%
AUDUSD	7.323%	5.857%	7.012%	9.393%	**10.189%**	7.95%
EURUSD	1.543%	0.931%	2.747%	0.999%	**3.132%**	1.87%
Iterative Training						
USDJPY	5.902%	**7.476%**	5.301%	6.561%	7.166%	6.48%
USDCAD	8.596%	7.949%	6.046%	**10.322%**	7.663%	8.12%
AUDUSD	7.348%	6.498%	**11.959%**	8.105%	10.197%	8.82%
EURUSD	(-0.174%)	**6.731%**	(-3.956%)	2.590%	4.081%	1.85%

Average best return of 4 currency pairs

= (8.305+11.795+10.189+3.132) / 4 = 8.36%

Average best return of 4 currency pairs (Iterative Training)

= (7.476+10.322+11.959+6.731) / 4 = 9.12%

After using iterative training approach, no substantial improvement on average log return is observed. Notes that the return of two epochs of EURUSD even becomes negative.

Nonetheless, the best epoch of AUDUSD and EURUSD is shifted to left, such that AUDUSD achieves best performance at epoch 3 (Original: epoch 5) while EURUSD get the best result at epoch 2 (Original: epoch 5). In other words, it means the model achieves best result earlier, and the training time (Number of Epoch) is less than before.

Also, substantial improvement on the best return is observed in AUDUSD and EURUSD when using iterative training approach. The return of the best epoch for AUDUSD increases by **1.77%**, while that of EURUSD also increases by **3.599%**. It might suggest these two currency pairs behave differently in the past, such that the past training samples could not help the model to predict recent price changes. By allowing the model to have access to recent samples, the model could achieve remarkable improvement. Notes that USDJPY and USDCAD does not show improvement in the best return, it might suggest that the recent market behaviour is similar than that of the past.

To conclude, the composite neural network model using Bi-LSTM and self-attention technique from BERT demonstrates its prediction power on 4 currency pairs. When training the model, iterative approach could help the model adapt to the market environment changes with less training time.

Chapter 4: Conclusion

4.1 Summary of Findings

In this dissertation, we presented the composite model which makes use of Bi-LSTM and self-attention layer extracted from BERT for predicting the forex market using multiple features in past timesteps.

For data pre-processing, some techniques from previous academic papers are used for cleaning and resampling the tick data. A popular library, TALib, is used for generating momentum and pattern indicators. Custom features are also added alongside the technical indicators. Moreover, a subset of relatively important features are selected by random forest algorithm to reduce the vector dimension in each timesteps.

With the help of self-attention layers from BERT and the bidirectional LSTM encoder, the model outperforms the underlying asset, and achieves 8.36% average best return for four currency pairs (USDJPY, USDCAD, AUDUSD, EURUSD) during 01-Jul-2018 - 18-May-2019. After applying the iterative training approach, substantial improvement is observed in AUDUSD and EURUSD. Also, the average best return increases to 9.12%.

Still, there are some assumptions that are not possible in real market. For example, we did not consider the slippage and transaction cost. Extra works need to be done before applying the model to the real market.

Chapter 5: Appendix

5.1 List of References

[1] Excel Markets “Best Times to Trade” [Online] Available:

https://www.excelmarkets.com/best-times-to-trade/

[2] Forex Minister “Forex Trading” [Online] Available:

http://forexminister.com/forex_inner.php

[3] Dukascopy “Historical Daata Feed” [Online] Available:

https://www.dukascopy.com/swiss/english/marketwatch/historical/

[4] Mineo A.M., Romito F. (2007) A method to “clean up” ultra high-frequency data, Statistica & Applicazioni, 5, 167–186

[5] MARCOS LÓPEZ DE PRADO (2018) “Advances in Financial Machine Learning”. Wiley

[6] scikit-learn “MinMaxScaler” [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[7] TALib “Momentum Indicators” [Online] Available: http://mrjbq7.github.io/ta-lib/func_groups/momentum_indicators.html

[8] TALib “Pattern Recognition” [Online] Available: http://mrjbq7.github.io/ta-lib/func_groups/pattern_recognition.html

[9] Wikipedia “Feature Selection” [Online] Available:

https://en.wikipedia.org/wiki/Feature_selection

[10] Wikipedia “Random Forest” [Online] Available:

https://en.wikipedia.org/wiki/Random_forest

[11] scikit-learn “RandomForestClassifier” [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[12] scikit-learn “K-Fold cross-validator” [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

[13] scikit-learn “k-nearest neighbors vote” [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[14] scikit-learn “Linear Support Vector Classification” [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[15] scikit-learn “Logistic Regression Classifier” [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[16] scikit-learn “Gaussian Naïve Bayes” [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[17] scikit-learn “Perceptron” [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

[18] Wikipedia “Long short-term memory” [Online] Available: https://en.wikipedia.org/wiki/Long_short-term_memory

[19] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735

[20] Wikipedia “Bidirectional Recurrent Neural Networks” [Online] Available: https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks

[21] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." Signal Processing, IEEE Transactions on 45.11 (1997): 2673-2681.2. Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. Google Scholar

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, "Attention is all you need", 2017, [online] Available: https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[24] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733, 2016.

[25] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In Empirical Methods in Natural Language Processing, 2016.

[26] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017.

[27] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130, 2017.