042 Time Series Basics with Pandas and Finance Data

Project Plan

Ho Ka Leung (3035106449)
# Table of Contents

**Background** .......................................................................................................................... 4

**Introduction** ............................................................................................................................... 5

**Scope** ......................................................................................................................................... 5

**Focuses** ....................................................................................................................................... 5

- Data ............................................................................................................................................... 5
- Evaluation ....................................................................................................................................... 5

**Deliverable** .................................................................................................................................. 5

- Deep Neural Network Model ....................................................................................................... 5

**Expected Outcome** ..................................................................................................................... 6

**Project Methodologies** .................................................................................................................. 7

- Data ............................................................................................................................................... 7

- Pre-processing ............................................................................................................................ 7

- Training ........................................................................................................................................... 7

  - Model .......................................................................................................................................... 7
  - Input ............................................................................................................................................. 8
  - Output ......................................................................................................................................... 8
  - Learning Algorithm .................................................................................................................... 8

- Validation ....................................................................................................................................... 8

  - Validation Algorithm ................................................................................................................ 8
  - Hyperparameters ....................................................................................................................... 8

- Testing and Evaluation .................................................................................................................. 8

  - Loss Function ........................................................................................................................... 8
  - Target Accuracy ......................................................................................................................... 9

**Technical Details** .......................................................................................................................... 9

- Programming Language ............................................................................................................. 9
- Environment ................................................................................................................................. 9
- Framework ..................................................................................................................................... 9

**Implementation Details** .............................................................................................................. 10

- Dataset ......................................................................................................................................... 10

  - Format ....................................................................................................................................... 10
  - Collection ................................................................................................................................. 10
  - Currency .................................................................................................................................... 15
  - Non-Currency .......................................................................................................................... 16
Conclusion

Possible Improvement

Analysis

Project Status

Fulfillment of Proposal

Other Files

Experiments and Results

Conclusion and Future Works

Fulfillment of Proposal

Project Status

Analysis

Possible Improvement

Conclusion
Background

New technology is always a strong driving force of economics and financial market. Google, Facebook have brought numerous profits to their investors previously. An emerging big trend is Cryptocurrency. However, for any technology that intends to thrive and bring profit must stay relevant and serve its purpose. One of the controversies of Cryptocurrency is that they are treated as asset more than a currency\(^1\), which violates the philosophy of this latest financial advancement.

In order to evaluate the potential of Cryptocurrency, this project aims to explore how Cryptocurrency resembles fiat currency in the market in terms of price behavior.

The first stage of the project will be to train a deep learning model that can accurately classify currency and non-currency (e.g. security, gold) by learning the features of their standardized time series data. After that, in the second stage, time series data of different Cryptocurrency will be fed into the model to calculate the degree of likelihood to be classified as a currency. The Cryptocurrency with highest score is deemed to be the most promising one to replace traditional currency.

---

\(^1\) "Bitcoin Is An Asset, Not A Currency"  
https://www.forbes.com/sites/jeffreydorfman/2017/05/17/bitcoin-is-an-asset-not-a-currency/#1c0531362e5b
Introduction

Scope
This project will focus on various financial data. By analyzing 7-8 years of financial data, a deep learning model will be used to evaluate the price behavior of Cryptocurrency. The models and algorithms used are limited to ones that were shown to be effective in time series analysis.

The project involves Microsoft Cognitive Toolkit (CNTK) and pandas (a Python data library), which is a project requirement from Microsoft.

Focuses
There will be two major focuses of this project, data and evaluation.

Data
The focus of earlier phase is to understand financial data. Different types of financial data will be studied. Subsequently, some of them will be selected for processing to produce clean time series data that are ready for training. The data processing techniques will differ based on different metrics and types of stock.

Evaluation
The focus of later phase is to evaluate the performance of the neural network model. The difference between the predicted result of the neural network and the actual result should be accurately measured and calculated to facilitate improvements on the network. The final performance of the network should also be used to demonstrating the possibilities of the proposed approach.

Deliverable

Deep Neural Network Model
The final product of the project is a well-trained deep neural network model that takes some sequence of standardized financial data of as input and return whether it is from currency or non-currency.
Expected Outcome

The final neural network is expected to have more than 95% of accuracy.
Project Methodologies

Data

20 sets of data will be used. Half of them should be exchange rates of major currencies. The other 10 are price data of different securities and assets, like stock and gold. Data from 2010 to 2017 are used. 80% of them is used for training and validation, 20% is used for evaluation.

Pre-processing

Pandas, an open-source Python library providing data structures and data analysis tools will be used to grab, parse, process and store the data. All data will be standardized using the formula:

$$\text{Standardized data} = \frac{\text{price} - \mu}{\sigma}$$

Training

Model

Long Short-term Memory (LSTM), a recurrent neural network that use a complex structure\(^2\) to resemble the human brain’s mechanism of remembering and forgetting information, has shown to be suitable for classifying, processing and predicting time series\(^3\).

\(^2\) The repeating module in an LSTM contains four interacting layers.

**Input**

The standardized price data will be split into numerous small sequences of data to feed into the learning model.

**Output**

The model will return a class label [0, 1] or [1, 0], representing non-currency and currency respectively. The result will then be compared to the actual label.

**Learning Algorithm**

A commonly used learning method for deep neural network, Stochastic Gradient Descent, is used as the learning algorithm.

**Validation**

**Validation Algorithm**

k-fold cross-validation is used as the validation algorithm. 80% of the data will be split into k subsets (usually k=5 or k=10). There will be k times of training. Each time k - 1 sets of data are used on training and the remaining one is used to validate the result. The final result is calculated by finding the average error rate. This method can maximize the data usage.

**Hyperparameters**

During validation stage, some hyperparameters will be tweeted to fine-tune the model according to the validation result. These include optimal length of sequence, epoch, minibatch size, learning rate, how to split data and whether using a dropout layer (to avoid overfitting)

**Testing and Evaluation**

**Loss Function**

Cross entropy with softmax loss with this formula is used to calculate error rate:
\[
\text{softmax}(x) = \left[ \frac{\exp(x_1)}{\sum_i \exp(x_i)}, \frac{\exp(x_1)}{\sum_i \exp(x_i)}, \ldots, \frac{\exp(x_1)}{\sum_i \exp(x_i)} \right]
\]

cross entropy with softmax(o,t) = \(-\sum_i t_i \log(\text{softmax}(o)_i)\)

**Target Accuracy**

The target accuracy of the model is 95%. After it is achieved, the model can be used to evaluate Cryptocurrencies.

**Technical Details**

**Programming Language**

Python will be the programming language used in this project due to its functional paradigm and vast support on machine learning.

**Environment**

Jupyter Notebook deployed on Azure Virtual Machine is used as the IDE of the project.

**Framework**

Microsoft Cognitive Toolkit (CNTK) is the deep learning framework, developed by and Microsoft. It supports wide range of deep learning algorithms, including LSTM. It also is optimized on different computing structure and takes advantage of Azure environment.
Implementation Details

Dataset

Data used in the project consists of 10 sets of exchange rates of major currencies and 10 sets of price data of different securities and assets. Data from 01/08/2010 to 12/31/2017 are used because the price data of Bitcoin was not put in Bloomberg terminal before July 2010.

Format

The data are the last trading price on each day and are converted to USD.

Collection

Data was directly collect from Bloomberg Terminal in KKL Building using the following settings:

1.
2.

Spreadsheet Builder
Use this Spreadsheet Builder to customize the data and layout of your Excel file.

- **Blank Data Table**: Creates a data table comprised of data fields and securities
- **Historical Data Table**: Creates a data table comprised of historical data fields and securities
- **Intraday Data Table**: Creates a data table comprised of intraday data fields and securities

Bloomberg
3. 

```
 Spreadsheet Builder

All Securities

- Search for one or more securities
- Recently Used
- Security Monitor
- Shared Monitor
- NW Monitor
- Shared NW Monitor
- Portfolio
- Equity Index
- Fixed Income Index
- Custom Index (CIK)
- Benchmark
- EMS Orders
- Equity Screen
- Security List (LST)
- Fund Screen (FSRC)
- Recent Securities
- Bloomberg Peers
- Currency Baskets

Selected Securities
- EURUSD Currency
- JPYUSD Currency

Bloomberg
```

4. Bloomberg
### Spreadsheet Builder

#### Date Calendar

- **From:** 08/01/2010
- **To:** 12/31/2017
- **Today:**
- **Periodicity:** Daily

#### Period Calendar

#### Relative Calendar

#### Optional Parameters

**Fill and Alignment Settings**

- **Fill with:** Carry over Last Value
- **Alignment Calendar:** 7D-7 BUS DAY NO HOLIDAY

**Normalization**

- **Normalization Type:** No Normalization

**Currency and Pricing Source**

- **Currency:** USD US Dollar
- **Pricing Source:**

**Quote and Quote Type**

- **Quote (Yield quoted securities only):**
- **Yield:**
- **Price:**
- **Quote Calculation:**
- **Period End:**
- **Period Average:**

---

**Bloomberg**
The data is then saved as a .csv file

Currency

G10 Currencies, 10 most heavily traded currencies in the world\(^4\), are chosen as the 10 sets of currency data. As USD, one of G10 Currencies, is chosen as the standard base currency of all the data sets, it is replaced by HKD. The finalized ten currencies are:

\(^4\) https://en.wikipedia.org/wiki/G10_currencies
<table>
<thead>
<tr>
<th>Name</th>
<th>ISO code</th>
<th>Bloomberg Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euro</td>
<td>EUR</td>
<td>EURUSD Curncy</td>
</tr>
<tr>
<td>Japanese yen</td>
<td>JPY</td>
<td>JPYUSD Curncy</td>
</tr>
<tr>
<td>Pound sterling</td>
<td>GBP</td>
<td>GBPUSD Curncy</td>
</tr>
<tr>
<td>Swiss franc</td>
<td>CHF</td>
<td>CHFUSD Curncy</td>
</tr>
<tr>
<td>Australian dollar</td>
<td>AUD</td>
<td>AUDUSD Curncy</td>
</tr>
<tr>
<td>New Zealand dollar</td>
<td>NZD</td>
<td>NZDUSD Curncy</td>
</tr>
<tr>
<td>Canadian dollar</td>
<td>CAD</td>
<td>CADUSD Curncy</td>
</tr>
<tr>
<td>Swedish krona</td>
<td>SEK</td>
<td>SEKUSD Curncy</td>
</tr>
<tr>
<td>Norwegian krone</td>
<td>NOK</td>
<td>NOKUSD Curncy</td>
</tr>
<tr>
<td>Hong Kong dollar</td>
<td>HKD</td>
<td>HKDUSD Curncy</td>
</tr>
</tbody>
</table>

**Non-Currency**

Various type of asset is chosen to represent the price behavior of non-currency. The standard for choosing a non-currency financial product is purchasable, just like currency. The finalized ten non-currencies are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Market represented</th>
<th>Bloomberg Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic 1st ‘GC’ Future</td>
<td>Precious Metal - Gold</td>
<td>GC1 Comdty</td>
</tr>
<tr>
<td>Generic 1st ‘SI’ Future</td>
<td>Precious Metal - Silver</td>
<td>SI1 Comdty</td>
</tr>
<tr>
<td>Generic 1st ‘CL’ Future</td>
<td>Commodity - Crude Oil</td>
<td>CL1 Comdty</td>
</tr>
<tr>
<td>Generic 1st ‘C’ Future</td>
<td>Commodity - Corn</td>
<td>C1 Comdty</td>
</tr>
<tr>
<td>PowerShares QQQ Trust Series 1</td>
<td>Stock – NASDAQ-100</td>
<td>QQQ US Equity</td>
</tr>
<tr>
<td>SPDR Dow Jones Industrial Average ETF Trust</td>
<td>Stock - Dow Jones Industrial Average</td>
<td>DIA US Equity</td>
</tr>
<tr>
<td>SPDR S&amp;P 500 ETF Trust</td>
<td>Stock - S&amp;P500</td>
<td>SPY US Equity</td>
</tr>
<tr>
<td>Annaly Capital Management Inc</td>
<td>Real Estate - Mortgage</td>
<td>NLY US Equity</td>
</tr>
<tr>
<td>Simon Property Group Inc</td>
<td>Real Estate – Regional Malls</td>
<td>SPG US Equity</td>
</tr>
</tbody>
</table>
Precious Metal
Chemically inactive metal that has high economic and industrial value. Usually deemed as value-preserving. Futures is chosen as the representing financial product.

Commodity
Commonly used resources in human activities. Futures is chosen as the representing financial product.

Stock
Shared ownership of publicly listed companies that reflects the value of the company. Exchange Traded Fund (ETF), which cover major stocks in an index as its portfolio, is chosen over stock exchange index as it can be traded directly.

Real Estate
Property consisting land or property. As there are various types and markets of real estate, Real Estate Investment Trust (REITs), which collectively invests in real estate market and pay dividend based on rental profit, chosen to represent the price behavior of real estate market.

Python Notebooks
There are three Python Notebookss in the final deliverables, namely “Experiments.ipynb”, “Validation.ipynb” and “Model.ipynb”. Experiments.ipynb was used to do some explorative experiments on the data. Validation.ipynb is to train and validate the model. Model.ipynb is a finalized model that can evaluate any given sequence.

Common Methods
Methods used in more than one scripts with the most general variables.

5 One example in Hong Kong is Link REIT
**get_data(file_name, is_currency)**

It takes the argument `file_name`, which is relative path string link to a .csv file and read it as a dataframe. It then populates ‘is_currency’ field using 1-hot encoding according to the Boolean value `is_currency`. Finally, it returns the dataframe.

<table>
<thead>
<tr>
<th>Is_currency</th>
<th>1-hot encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>[1, 0]</td>
</tr>
<tr>
<td>False</td>
<td>[0, 1]</td>
</tr>
</tbody>
</table>

**normalize(data)**

Minus all the numeric fields of a dataframe `data` by the mean and then divided by standard deviation to have a z-score sequence.

**Example**

\[1181.7, 1183.4, 1185.2, 1193.7, 1197.2, 1203.4, 1200.7, 1196.2, 1197.5, 1214.8, 1214.9\] is transformed to \[-1.44, -1.28, -1.12, -0.323, 0.00424, 0.583, 0.331, -0.0891, 0.0323, 1.65, 1.66\]

**prepare_data(data, input_size, dataset_names, time_steps=1)**

Split time series `data[dataset_names]` into small sequences. Return two array compactible to CNTK model as input and output. Variable `input_size` determines the length of each sequence where `time_steps` determines how many steps it takes to capture one sequence. When `time_steps` is smaller than 1, it is set to `input_size` automatically. Output is always an array of field ‘is_currency’, which is a constant in any `data`, with length equal to number of inputs.

**Examples**

Let `data[dataset_names]=[-1.44, -1.28, -1.12, -0.323, 0.00424, 0.583, 0.331, -0.0891, 0.0323, 1.65, 1.66]` and `data[‘is_currency’]=[[[1, 0]]] * 11`

When `input_size=3` and `time_steps=1`, return input as `[[[-1.44], [-1.28 ], [-1.12]], [[-1.28], [-1.12], [-0.323]], [[-1.12], [-0.323], [0.00424]], [[-0.323], [0.00424], [0.583]], [[0.00424], [0.583], [0.331]], [[0.583], [0.331], [-0.0891]], [[0.331], [-0.0891], [0.0323]], [[-0.0891], [0.0323], [1.65]], [[0.0323], [1.65], [1.66]]]` and output as `[[[1, 0]]] * 9`
When $input_size=3$ and $time_steps=2$, return input as $\begin{bmatrix} [-1.44], [-1.28], [-1.12] \\ [-1.28], [-0.323], [0.00424] \\ [-1.12], [0.00424], [0.583], [0.331] \\ [0.331], [-0.0891], [0.0323] \\ [0.0323], [1.65], [1.66] \end{bmatrix}$ and output as $[[1.0]] * 5$

When $input_size=3$ and $time_steps=0$ (Adjust to 3), return input as $\begin{bmatrix} [-1.44], [-1.28], [-1.12] \\ [-0.323], [0.00424], [0.583] \\ [-1.12], [0.00424], [0.331] \\ [0.331], [-0.0891], [0.0323] \end{bmatrix}$ and output as $[[1.0]] * 3$

$create_model(input_variable, input_size, with_dropout=True)$

Return a CNTK model with 3 layers. The first layer is recurrent neural network that accepts sequence with length $input_size$. The second and optional layer is a dropout layer, which is used to avoid overfitting, its presence is determined by $with_dropout$. The last layer is a dense layer with 2 fully connected nodes, representing two classes to be classified.

$next_batch(x, y, batch_size)$

Generate mini-batches for training and testing from $x, y$ (format is same as returned values of $prepare_data()$). Variable $batch_size$ determines how many cases are in one batch.

$Experiments.ipynb$

For content others than function declaration, please refers to Section $Experiments$ and Results

$train(input_size, epochs, batch_size, learning_rate, pos_ds='EURUSD Curncy', neg_ds='CL1 Comdty')$

Handles training logic. It first splits the data specified by $pos_ds$ and $neg_ds$ into sequences with length $input_size$. The data is then stratified split into training set (75%) and testing set (25%). Created by method $create_model()$, the model is then trained with training set and tested against testing set. Progress is printed thorough training.
Format of progress printing

`gen_data(is_currency, length)`

Basically an alternative for `get_data()` but instead of reading data from a .csv file, it generates two sequences using functions $\sin(x)$ and $\sin(2x)$ when $x = 0, 1, 2, \ldots, \text{length} - 1$

`experiment(time_steps, epochs, batch_size, learning_rate, dataset_names)`

A variant of `train()` but 1.) it takes one dict `dataset_names` instead of `pos_ds` and `neg_ds`, and 2.) instead of splitting one sequence (one from non-currency dataset and one from currency dataset, therefore two in actual case, and same applies afterwards unless specified) into training set and testing set, it takes sequences specified by `dataset_names` as training and testing set. Therefore, no splitting is performed on the sequences.

**Validation.ipynb**

For content others than function declaration, please refers to Section **Experiments and Results**

`validate(input_size, epochs, batch_size, learning_rate, dataset_names, jump=1, with_dropout=False, optimized=False)`

Similar to `experiment()` in Experiment.ipynb but with same modifications for using in grid search logic: 1.) Add variable `jump` that specifies how to split the sequence, it is
directly pass through to `prepare_data()` (for training set only) as variable `time_steps`,
2.) add variable `with_dropout` determines whether to use dropout layer in the
construction of the model, it is directly pass through to `create_model()`, 3.) add
variable `optimized` determining whether the program will continuous train the model
even after the supplied `epochs` is ran, hoping to maximize accuracy without over
fitting, 4.) Turn off progress printer and 5.) return the testing error

```
train(input_size, epochs, batch_size, learning_rate, dataset_names, jump=1,
with_dropout=False, optimized=False)
```

Similar to `tvalidate()` but with same modifications: 1.) Turn on progress printer and 2.)
return the trainer object, model object and a list storing error on test set in each epoch

![Graph](image.png)

*The theoretical stopping point when `optimized=True`, supposed given `epochs` did not pass that point*
Model.ipynb

test(file_name, field_name)
When given any specific sequence data field_name in .csv file given by file_name, a trained model (given by model file) will evaluate how many of its subsequences are classified as currency

Other Files

model
Stored the trained state of model, used by model.ipynb to quickly restore the model and testing

trainer and trainer.ckp
Stored trainer state when finishing training the model. Can be used for further training and modification

Experiments and Results
Before doing experiments, all random seeds are fixed for replicable result. CNTK is also set to use GPU whenever possible.

Experiment 1 (Cells 7 - 40 of Experiments.ipynb)

Goals
1. To derive a training equation that can help fine-tuning hyperparameters
2. To test whether the constructed model can differentiate two time series with different behavior
3. If Goal 1 is achieved, then to establish a standard setting for any upcoming experiments

Datasets
2 sequences, namely sin(x) and sin(2x) are used while the former is classified as negative and latter positive
Training and Test Split

75% of all data (half positive and half negative, same applies from now on) as training set and 25% as test set.

**Procedures**

1. Generate data (Cell 7)
   
   With method `gen_data()`.

2. Normalize data (Cell 7)
   
   With method `normalize()`.

**Plotting normalized data (10% only)**

3.1 Initial training (Cells 9)

- `input_size` - Set to 1 week as base value;
- `epochs` - Set arbitrarily to have an acceptable training time;
- `batch_size` and `learning_rate` - Using settings from CNTK documentation example

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>20</td>
<td>200</td>
<td>0.0005</td>
<td>49.19%</td>
</tr>
</tbody>
</table>

3.2. Establish training equation (Cells 10 - 17)

Propose and verify equation

\[
\text{training progress} = \text{number of training cases} * \frac{\text{epochs} \times \text{learning_rate}}{\text{batch_size}}
\]

---

6 https://cntk.ai/pythondocs/sequence.html
where `input_size` has negligible effect on training progress and thus treated as an independent variable.

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>20</td>
<td>200</td>
<td>0.0005</td>
<td>54.96%</td>
</tr>
<tr>
<td>28</td>
<td>20</td>
<td>200</td>
<td>0.0005</td>
<td>51.04%</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>200</td>
<td>0.0005</td>
<td>49.04%</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>200</td>
<td>0.0005</td>
<td>49.93%</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>100</td>
<td>0.0005</td>
<td>49.04%</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>50</td>
<td>0.0005</td>
<td>49.93%</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>200</td>
<td>0.001</td>
<td>49.04%</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>200</td>
<td>0.002</td>
<td>49.93%</td>
</tr>
</tbody>
</table>

Observations:

1. The result is consistent with the equation, although -
2. increase in training progress does not guarantee higher accuracy, and
3. Sequence of length 14 i.e. two weeks has the best result

3.3. By fixing training progress, achieve least training time without compromising accuracy (Cells 18 - 23)

First combine the best results from last stage, then tweet the parameters

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>20</td>
<td>200</td>
<td>0.001</td>
<td>55.19%</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>200</td>
<td>0.02</td>
<td>55.41%</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>400</td>
<td>0.04</td>
<td>55.41%</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>800</td>
<td>0.08</td>
<td>55.41%</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1600</td>
<td>0.16</td>
<td>56.22</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>3200</td>
<td>0.32</td>
<td>56.07</td>
</tr>
</tbody>
</table>

Observation:

1. Equation falls apart when values go too extreme, must check the accuracy for every update
3.4. Continuously increase training progress to maximize accuracy (Cells 24 - 32)

3.4.1

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>10</td>
<td>1600</td>
<td>0.16</td>
<td>61.63%</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>160</td>
<td>0.16</td>
<td>48.81%</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1600</td>
<td>1.6</td>
<td>50%</td>
</tr>
</tbody>
</table>

3.4.2

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>100</td>
<td>1600</td>
<td>0.16</td>
<td>60.52%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>160</td>
<td>0.16</td>
<td>75.85%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>1600</td>
<td>1.6</td>
<td>50%</td>
</tr>
</tbody>
</table>

3.4.3

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>100</td>
<td>160</td>
<td>0.16</td>
<td>100%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>16</td>
<td>0.16</td>
<td>100%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>160</td>
<td>1.6</td>
<td>100%</td>
</tr>
</tbody>
</table>

Observations:

1. 100% accuracy can be achieved

<table>
<thead>
<tr>
<th>average</th>
<th>since</th>
<th>average</th>
<th>since</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>loss</td>
<td>last</td>
<td>metric</td>
<td>last</td>
<td></td>
</tr>
</tbody>
</table>

   Learning rate per minibatch: 0.16
   0.694 0.694 0.5 0.5 16
   0.695 0.695 0.521 0.531 48
   0.694 0.693 0.518 0.516 112
   0.695 0.696 0.533 0.547 240
   0.696 0.697 0.538 0.543 496
   0.696 0.695 0.522 0.506 1000
   0.694 0.692 0.502 0.482 2032
   0.693 0.693 0.491 0.481 4088
   0.685 0.677 0.449 0.487 8176
   0.385 0.8844 0.231 0.0139 16368
   0.193 0.000945 0.116 0.0 32752

   accuracy on an unseen minibatch: 1.0

2. From the training process of setting {input_size=14, epochs=100, batch_size=160, learning_rate=0.16}, it can be observed that very low loss was achieved back examples 16368 i.e. epochs~25.
3. Hypothesis: Less training can be done to achieve same result

3.5. Optimize training progress as well as training time to prevent overfitting (Cells 33 - 40)

3.5.1 Reduce the last three settings by 75% respectively

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>25</td>
<td>160</td>
<td>0.16</td>
<td>100%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>64</td>
<td>0.16</td>
<td>100%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>160</td>
<td>0.4</td>
<td>100%</td>
</tr>
</tbody>
</table>

3.5.2 Further reduce the three settings by 20%

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>20</td>
<td>160</td>
<td>0.16</td>
<td>92.22%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>80</td>
<td>0.16</td>
<td>92.07%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>160</td>
<td>0.32</td>
<td>91.47%</td>
</tr>
</tbody>
</table>

3.5.3 Pick the setting with highest drop in accuracy and decrease epoch by 1 each time instead

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>9</td>
<td>160</td>
<td>0.4</td>
<td>100%</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>160</td>
<td>0.4</td>
<td>90.22%</td>
</tr>
</tbody>
</table>

**Conclusion**

1. Two sequences are differentiable with enough training
2. Setting {input_size=14, epochs=9, batch_size=160, learning_rate=0.4} is set to the starting point for any upcoming experiments

**Experiment 2 (Cells 41 - 68 of Experiments.ipynb)**

**Goals**

1. To further test whether the constructed model can differentiate currency and c time series

**Datasets**

2 sequences, EURUSD Curncy from currency.csv and CL1 Comdty from non-currency.csv
Training and Test Split
75% of all data as training set and 25% as test set.

Procedures

1. Read data from csv (Cell 41)
   With method `get_data()` to read currency.csv and non-currency.csv

2. Normalize data (Cell 41)
   With method `normalize()`.

![Plotting normalized data](image)

3.1 Initial training (Cells 43)

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>9</td>
<td>160</td>
<td>0.4</td>
<td>50%</td>
</tr>
</tbody>
</table>

3.2 Double training (Cells 44 - 46)

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>18</td>
<td>160</td>
<td>0.4</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>80</td>
<td>0.4</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>160</td>
<td>0.8</td>
<td>50%</td>
</tr>
</tbody>
</table>

3.3 Quadruple training (Cells 47 - 49)
<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>36</td>
<td>160</td>
<td>0.4</td>
<td>48.67%</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>40</td>
<td>0.4</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>160</td>
<td>1.6</td>
<td>50%</td>
</tr>
</tbody>
</table>

3.4 10 times training (Cells 50 - 52)

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>90</td>
<td>160</td>
<td>0.4</td>
<td>55.48%</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>16</td>
<td>0.4</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>160</td>
<td>4</td>
<td>50%</td>
</tr>
</tbody>
</table>

3.5 Continuously double training to maximize accuracy (Cells 53 - 68)

3.5.1

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>180</td>
<td>160</td>
<td>0.4</td>
<td>56.67%</td>
</tr>
<tr>
<td>14</td>
<td>90</td>
<td>80</td>
<td>0.4</td>
<td>56.30%</td>
</tr>
<tr>
<td>14</td>
<td>90</td>
<td>160</td>
<td>0.8</td>
<td>57.85%</td>
</tr>
</tbody>
</table>

3.5.2

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>180</td>
<td>160</td>
<td>0.8</td>
<td>58.67%</td>
</tr>
<tr>
<td>14</td>
<td>90</td>
<td>80</td>
<td>0.8</td>
<td>58.59%</td>
</tr>
<tr>
<td>14</td>
<td>90</td>
<td>160</td>
<td>1.6</td>
<td>56.07%</td>
</tr>
</tbody>
</table>

3.5.3

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>360</td>
<td>160</td>
<td>0.8</td>
<td>59.70%</td>
</tr>
<tr>
<td>14</td>
<td>180</td>
<td>80</td>
<td>0.8</td>
<td>60.30%</td>
</tr>
<tr>
<td>14</td>
<td>180</td>
<td>160</td>
<td>1.6</td>
<td>58.07%</td>
</tr>
</tbody>
</table>

3.5.4

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>360</td>
<td>80</td>
<td>0.8</td>
<td>59.33%</td>
</tr>
<tr>
<td>14</td>
<td>180</td>
<td>40</td>
<td>0.8</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>180</td>
<td>80</td>
<td>1.6</td>
<td>61.41%</td>
</tr>
</tbody>
</table>
3.5.5

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>360</td>
<td>160</td>
<td>1.6</td>
<td>59.93%</td>
</tr>
<tr>
<td>14</td>
<td>180</td>
<td>80</td>
<td>1.6</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>180</td>
<td>160</td>
<td>3.2</td>
<td>50%</td>
</tr>
</tbody>
</table>

**Conclusion and Observation**

3. Two sequences are roughly differentiable (up to 60%) with enough training.

4. For 3.5.4, when some updates cease to increase in accuracy, the better one shows more converging trend in loss, possibly indicating some is still fitting the data while some is overfitting.

Best case in 3.5.4, showing strong converging trend
Slightly dropped but still differentiable, somehow still converging

Worst case, showing no converging trend at all

5. Therefore, apart from high accuracy, converging can still be treated as a sign of good learning
Experiment 3 (Cells 70 - 82 of Experiments.ipynb)

**Goals**

1. To verify whether using more dataset to train will decrease the performance due to more noises

**Datasets**

2 sets of datasets as a controlled experiment

**Dataset 1**

- Training set: EURUSD Curncy from currency.csv and CL1 Comdty from non-currency.csv
- Testing set: JPYUSD Curncy from currency.csv and C 1 Comdty from non-currency.csv

**Dataset 2**

- Training set: EURUSD Curncy and GBPUSD Curncy from currency.csv and CL1 Comdty and QQQ US Equity from non-currency.csv
- Testing set: JPYUSD Curncy from currency.csv and C 1 Comdty from non-currency.csv

**Procedures**

1. Read data from csv (Cell 41)
   
   With method `get_data()` to read currency.csv and non-currency.csv

2. Normalize data (Cell 41)
   
   With method `normalize()`. 
3.1 First comparing (Cells 73 - 74)

Using final setting from experiment 1 but with epoch 10 and 5 for dataset 1 and dataset 2 respectively because the actual data amount of dataset 1 is half of dataset 2

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>10</td>
<td>160</td>
<td>0.4</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>160</td>
<td>0.4</td>
<td>50%</td>
</tr>
</tbody>
</table>
Observation:
1. No changed, but both of them are 50% i.e. pure guess, therefore should use more settings to further investigate

3.2 Using doubled setting (Cells 75 - 80)

Using final setting from experiment 1 but with epoch 10 and 5 for dataset 1 and dataset 2 respectively because the actual data amount of dataset 1 is half of dataset 2

3.2.1

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>20</td>
<td>160</td>
<td>0.4</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>160</td>
<td>0.4</td>
<td>52.22%</td>
</tr>
</tbody>
</table>

3.2.2

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>10</td>
<td>80</td>
<td>0.4</td>
<td>45.23%</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>80</td>
<td>0.4</td>
<td>48.40%</td>
</tr>
</tbody>
</table>

3.2.3

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>10</td>
<td>160</td>
<td>0.8</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>160</td>
<td>0.8</td>
<td>52.28%</td>
</tr>
</tbody>
</table>

Observation:
1. All three increased with different values, but may need one more verification from very different settings

3.3 Using setting from experiment 2 (Cells 81 - 82)

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>180</td>
<td>80</td>
<td>1.6</td>
<td>50%</td>
</tr>
<tr>
<td>14</td>
<td>90</td>
<td>80</td>
<td>1.6</td>
<td>58.01%</td>
</tr>
</tbody>
</table>

**Conclusion**

1. Different from what was expected, more data increase accuracy. That may be because more data can let model generalize better
2. The results show that using all data to full train is promising, but may need a quick experiment
Experiment 4 (Cells 83 - 86 of Experiments.ipynb)

Goals

1. To have a glance on using all data to train the model

Datasets

Training Set

- EURUSD Currency, JPYUSD Currency, GBPUSD Currency, CHFUSD Currency, AUDUSD Currency, NZDUSD Currency, CADUSD Currency and SEKUSD Currency from currency.csv
- GC1 Comdty, SI1 Comdty, CL1 Comdty, C 1 Comdty, QQQ US Equity, DIA US Equity, SPY US Equity and NLY US Equity from non-currency.csv

Testing set

- NOKUSD Currency and NOKUSD Currency from currency.csv
- SPG US Equity and PLD US Equity from non-currency.csv

Procedures

1. Read data from csv (Cell 41)

   With method `get_data()` to read currency.csv and non-currency.csv

2. Normalize data (Cell 41)

   With method `normalize()`.
3.3 Training (Cell 86)

Best setting from experiment 2 is chose over experiment 1 due to the poor initial performance in experiments 2 and 3. Variable *epochs* is set to $\frac{180}{8} = 22.5 \approx 23$ to compensate for increase in amount of data.

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>23</td>
<td>80</td>
<td>1.6</td>
<td>65.79%</td>
</tr>
</tbody>
</table>
Conclusion
1. Accuracy of 65.79% beating the maximum value in experiment 2 (61.41%, with only set of data to learning) without much fine-tuning, and
2. Loss showing converging trend is a promising result

Short Sum-up on Previous Experiment Results
1. 70% is a strong limitation for financial sequence data
2. Variable input_size always set to 14 based on the first training result may be biased
3. How data is split into sequences is not taken into account
4. Did not use dropout layer in all experiments
Solution: Grid Search with Cross Validation

Grid Search with Cross Validation (Cells 7 -8 of Validation.ipynb)

Goals
1. To find an optimal setting to use full training

Datasets
All training set used in experiment 4

Procedures
1. Read data from csv (Cell 7)
   With method get_data() to read currency.csv and non-currency.csv
2. Normalize data (Cell 7)
   With method normalize().
3. Define search scope (Cell 8)
   - input_size
     Independent variable.
   - epochs
When $jump=1$ i.e. the data is split as sliding window, the effect of $input_size$ on number of training cases is negligible. However, when $jump \neq 0$, the difference of number of training cases can be significant. Therefore, $epochs$ is treated as a dependent variable to compensate for more training data.

- **batch_size**
  Independent variable.
- **learning_rate**
  Dependent variable of $batch_size$ to retain similar (if not constant) training progress. It is scaled up like $batch_size$ does, with basic values given by experiment 2
- **dataset_names**
  Not in the scope of searching but rather put with different test sequence to calculate average loss of a particular setting
- **jump**
  Independent variable, indicating how many steps for it to split a sequence. To reduce search time, only 1 and 0 (adjusted to $input_size$) i.e. sliding window and discrete split is searched.
- **with_dropout**
  Independent variable, indicating whether use dropout layer
How different ways of splitting data can affect number of training cases

4. Search for best setting (Cell 8)

For each setting:

1. Initialize error

2. For each sequence in 8 datasets used to train
   I. Set the particular sequence as the testing set
   II. Set the remaining sequences as the training set
   III. Calculate parameter
   IV. Pass all arguments to \textit{validate}()
   V. Sum up the error returned by \textit{validate}()

3. Print the setting and average error
Results

Output of grid search, the setting with lowest error is boxed

Observations

1. Discrete split gives poorer result generally, possibly due to low utilization rate of data
2. $\text{input\_size}=14$ did not always give better result, relying on results from experiment 1 is clearly biased
3. $k=4$ i.e. $\text{batch\_size}=320$ and $\text{learning\_rate}=6.4$ consistently gives poor result, maybe due to too large $\text{learning\_rate}$
4. Dropout layer did help when data is split by sliding window

Full Train (Cells 9 – 12 of Validation.ipynb)

Goals

1. To train a model for final usage
Datasets
Identical to experiment 4

Procedures

1. Read data from csv (Cell 7)
   With method `get_data()` to read currency.csv and non-currency.csv

2. Normalize data (Cell 7)
   With method `normalize()`.

3. Training (Cell 10)
   Using setting given by grid search and epoch used in experiment 4

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Split way</th>
<th>Use of dropout layer</th>
<th>Optimized?</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>23</td>
<td>80</td>
<td>1.6</td>
<td>Moving window</td>
<td>Yes</td>
<td>Yes</td>
<td>69.21%</td>
</tr>
</tbody>
</table>

Observation:

1. New high in classifying financial time series data
2. Loss converges
3. Still could not beat 70%

Learning rate per minibatch: 1.6

<table>
<thead>
<tr>
<th>average loss</th>
<th>since last metric</th>
<th>average loss</th>
<th>since last metric</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.698</td>
<td>0.698</td>
<td>0.475</td>
<td>0.475</td>
<td>80</td>
</tr>
<tr>
<td>0.702</td>
<td>0.704</td>
<td>0.492</td>
<td>0.5</td>
<td>240</td>
</tr>
<tr>
<td>0.7</td>
<td>0.698</td>
<td>0.489</td>
<td>0.487</td>
<td>560</td>
</tr>
<tr>
<td>0.696</td>
<td>0.693</td>
<td>0.468</td>
<td>0.448</td>
<td>1200</td>
</tr>
<tr>
<td>0.696</td>
<td>0.695</td>
<td>0.471</td>
<td>0.474</td>
<td>2480</td>
</tr>
<tr>
<td>0.697</td>
<td>0.699</td>
<td>0.482</td>
<td>0.492</td>
<td>5940</td>
</tr>
<tr>
<td>0.697</td>
<td>0.697</td>
<td>0.481</td>
<td>0.481</td>
<td>18160</td>
</tr>
<tr>
<td>0.695</td>
<td>0.694</td>
<td>0.472</td>
<td>0.463</td>
<td>28400</td>
</tr>
<tr>
<td>0.689</td>
<td>0.683</td>
<td>0.453</td>
<td>0.434</td>
<td>48880</td>
</tr>
<tr>
<td>0.684</td>
<td>0.679</td>
<td>0.438</td>
<td>0.423</td>
<td>81840</td>
</tr>
<tr>
<td>0.68</td>
<td>0.675</td>
<td>0.427</td>
<td>0.417</td>
<td>153760</td>
</tr>
<tr>
<td>0.677</td>
<td>0.674</td>
<td>0.422</td>
<td>0.416</td>
<td>327600</td>
</tr>
<tr>
<td>0.669</td>
<td>0.661</td>
<td>0.409</td>
<td>0.396</td>
<td>655280</td>
</tr>
</tbody>
</table>

trained with 23 epochs
accuracy on test set: 0.6921356690272084

3. Still could not beat 70%
4. No further training after epoch 23, may be due to
   I. *epochs* passes the point when test accuracy stops increasing i.e. overfitted
   II. local optimal
   III. test accuracy does not converge with more training

4. Verification (Cell 4)

Train with large number of epoch and record accuracy in every epoch

<table>
<thead>
<tr>
<th>input_size</th>
<th>epochs</th>
<th>batch_size</th>
<th>learning_rate</th>
<th>Split way</th>
<th>Use of dropout layer</th>
<th>Test Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>23</td>
<td>80</td>
<td>1.6</td>
<td>Moving</td>
<td>Yes</td>
<td>50%</td>
</tr>
</tbody>
</table>

Result:

[Graph showing trend of error over epochs]

Observations:
1. Point of overfitting takes place in epoch 50, with *epochs*=23, the model is not overfitted
2. As error does not converge, logic of *optimized* does not work
3. 69.21% is second highest possible accuracy, the first being 69.88% (*epochs*=8), showing settings derived from previous experiments are indeed helpful.

**Evaluating Model (Cell 6 of Model.ipynb)**

As the model does not have high enough accuracy to make a reasonable analysis on cryptocurrency, a detailed evaluation on the model is done for further analysis instead
Goals

1. To analyze the weakness or limitations of the model

Datasets

All dataset from currency.csv and non-currency.csv

Procedures

Pass every sequence into the test() method and output the classification one by one

Results

- EURUSD Curncy in currency.csv: 77.34% of the data is classified as currency
- JPYUSD Curncy in currency.csv: 41.45% of the data is classified as currency
- GBPUSD Curncy in currency.csv: 86.40% of the data is classified as currency
- CHFUSD Curncy in currency.csv: 74.88% of the data is classified as currency
- AUDUSD Curncy in currency.csv: 74.39% of the data is classified as currency
- NZDUSD Curncy in currency.csv: 73.95% of the data is classified as currency
- CADUSD Curncy in currency.csv: 79.60% of the data is classified as currency
- SEKUSD Curncy in currency.csv: 83.90% of the data is classified as currency
- NOKUSD Curncy in currency.csv: 87.40% of the data is classified as currency
- HKDUSD Curncy in currency.csv: 93.48% of the data is classified as currency
- GC1 Comdty in non-currency.csv: 33.02% of the data is classified as currency
- SI1 Comdty in non-currency.csv: 31.90% of the data is classified as currency
- CL1 Comdty in non-currency.csv: 87.40% of the data is classified as currency
- C 1 Comdty in non-currency.csv: 32.95% of the data is classified as currency
- QQQ US Equity in non-currency.csv: 54.38% of the data is classified as currency
- DIA US Equity in non-currency.csv: 56.35% of the data is classified as currency
- SPY US Equity in non-currency.csv: 64.70% of the data is classified as currency
- NLY US Equity in non-currency.csv: 34.44% of the data is classified as currency
- SP6 US Equity in non-currency.csv: 63.44% of the data is classified as currency
- PLD US Equity in non-currency.csv: 40.59% of the data is classified as currency

Observations

- Model performs quite well on currency data except JPYUSD Curncy.
- CL1 Comdty is exceptionally classified as currency, which is actually a non-currency
- All stock data has less than 50% accuracy
Conclusion and Future Works

Fulfillment of Proposal

As the accuracy of the model could not achieve the proposed 95%, the project could not process to the final stage, which was planned to be using the trained model to evaluate the price behavior of cryptocurrency. However, all the milestones (construction of neural network, first full run and achieving 68% accuracy) are met.

Project Status

The project is currently suspended.

Analysis

The accuracy consistently limited at 70% clearly blocks the project from proceeding. The grid search performed on the training set should be sufficient to eliminate the possibility for limitations from settings. The next consideration is limitations from the problem scope. As the whole project was based on the assumption that currency and non-currency are differentiable by their price behavior and their differences are features that can be learned by the LSTM model. This assumption may not be completely true. Three possible sources of limitations, namely model, problem or dataset arises subsequently.

Model Limitation

Take sine wave and cosine wave as example, although the two sequence is different from each other. The behavior of sequences fit to the model are identical thus cannot be learned. If some features of the time series are like this, the model cannot differentiate them with high accuracy.
One possible case of sequence cannot be learned by the model

**Problem Limitation**

Currency and non-currency may be not highly differentiable as both are driven by market and market behavior can be unpredictable at all. The model could only be doing classification based on made up or temporal feature. In this case, the problem cannot be improved by using machine learning / deep learning.

**Dataset Limitation**

Firstly, non-currency and currency data can be implicitly correlated. Thorough the project, I have learned that US Treasury Bond, which was once a candidate of non-currency, is actually backing the issuance of USD. Therefore, there price behavior are very similar. There might be other similar cases I have overlooked and became noises during training the model. For example, the low accuracy on CL 1 Comdty may be attributed to the fact USD is used as the clearing currency of petroleum, and therefore they are somehow correlated.
Another possibility is the choice of datasets limited the accuracy. For example, The majority of selected currencies are Western countries, therefore the model to biased and could not identify currencies from non-Western countries like Japan yen.

**Possible Improvement**

**Adopting Different Models**
Apart from RNN and LSTM, there are other deep learning model, for example CNN, and other machine learning algorithm as powerful as deep neural network. By adopting various model, not only the learnability of the data can be better evaluated, new approaches can also be introduced to the problem.

**Evaluate Cryptocurrency without High Accuracy Model**
A model without high accuracy could be an obstacle in this project, but not one in evaluating the potential of cryptocurrency. Different approaches like calculate similarity between cryptocurrency with another currency data, or deep clustering currency, non-currency and crypto-currency could also be useful in tackling this problem.

**More Data**
20 sets of 8 years data can only represent a handful of currencies and non-currencies. Maybe with more amount data, the implicit differentiability between currency and non-currency will be accentuated to be learned by the machine. With more data, multi-class classification can also be possible.

**Conclusion**
Although the performance of the model is unsatisfactory, and the proposed project result was not achieved, this project is still a fruitful and rewarding one. This project gave me a chance to dive into the deep learning world and get a taste of joy and struggles faced by a researcher. I have learned a lot of deep learning and machine learning techniques. I hope I can continue to learn and advance in the machine learning field and eventually use them to solve important problems in our world.
I would also want to express my gratitude towards my supervisor Dr. Anthony Tam, who spent a lot time assisting and make recommendations to this project, and all my friends who have helped me in different domains. This project could not proceed to today without all these helps. Thank you!