Task B5: Machine Learning 2

Conducted by:

Group 1

Team Members	*Email*
Thanh Tam Vo	103487596@student.swin.edu.au
Tai Minh Huy Nguyen	104220352@student.swin.edu.au

March 28, 2024

Table of Contents

1 Introduction

In this **Task B5: Machine Learning 2**, we will solve the multistep and multivariate prediction using many features (Open Price, High Price, Low Price, and Volume). The predicted *Close Price* in the next ***k*** days are performed by 3 models: RNNs, LSTM, and GRU.

2 Importing Dependencies

```
# Task B2
import numpy as np
import pandas as pd
import yfinance as yf
import datetime as dt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import os

# Task B3
import plotly.graph_objects as go
import plotly.express as px

# Task B4
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input,SimpleRNN,LSTM, GRU, Dense,
Dropout
import matplotlib.pyplot as plt
import sys

# Task B5
from tensorflow.keras import metrics
                                                                  Python
```

Figure 1: Importing Python Packages for Task B5

There are only one new several package that are need to be imported to evaluate the predicted result (Figure 1): **`metrics`** from **keras**.

3 Hyperparameters

Hyperparameters

```python
# Task B2
TICKER = "AAPL"
START_DATE = "2010-01-01"
END_DATE = "2022-12-31"
LOOK_UP_DAYS = 20
TRAINING_RATIO = 0.8    # 0.7 == 70%
SCALE_DATA = True
SCALING_METHOD = "MinMax"       # MinMax, Standard

# Task B3
TRADING_PERIOD = 60
CONSECUTIVE_DAYS = 300

# Task B4
NUMBER_OF_LAYER = 2
NUMBER_OF_HIDDEN_UNITS = 80
MODEL_NAME = "RNN"      ## "RNN". "LSTM", "GRU"
DROP_OUT_RATE = 0     ## dropout rate in [0,1]
NUMBER_OF_EPOCHS = 10
BATCH_SIZE = 12
FEATURE_PREDICT = "Close"     ## "Open", "High", "Close", "Low"
LOSS_FUNCTION = "huber_loss" ## "mean_squared_error", "mean_absolute_error", "huber_loss"
OPTIMIZER = "adam"  ## "adam", "RMSprop", "SGD"

# Task B5
K_SEQUENCE = 30
START_TEST_DATE = "2024-01-01"
END_TEST_DATE = "2024-03-16"
```

[2] Python

Figure 2: New hyperparameters for Task B5

In Figure 2, there are new added constants:

- **K_SEQUENCE** is the k sequential days that we need to predict the *Close Price*
- **START_TEST_DATE** is the beginning date in the testing set
- **END_TEST_DATE** is the end date in the testing set

Plus, there are some modifications compared to the previous tasks:

- **START_DATE** is the beginning date in the training set
- **END_DATE** is the end date in the training set

4 Dataset Preparation

In this task B5, the goal is looking at the set of features (*Open Price, High Price, Low Price, Volumes*) of the last **`LOOK_UP_DAYS`** to predict *Close Price* of the next **`K_SEQUENCE`** days.

Let:

- d denotes the number of **`LOOK_UP_DAYS`**
- k denotes the number of **`K_SEQUENCE`**
- s denotes the start date in the training set
- e denotes the last date in the training set
- c_i denotes the *Close Price* of the i^{th} date.
- o_i denotes the *Open Price* of the i^{th} date.
- h_i denotes the *High Price* of the i^{th} date.
- l_i denotes the *Low Price* of the i^{th} date.
- v_i denotes the *Volumes* of the i^{th} date.
- $\mathbf{x}_i = [o_i, h_i, l_i, v_i]^T$ (Transpose of vector $[o_i, h_i, l_i, v_i]$)
- $\mathbf{y}_{i:k} = [c_i, c_{i+1}, \cdots, c_{i+k-1}]^T$ (Transpose of vector $[c_i, c_{i+1}, \cdots, c_{i+k-1}]$)

Hence, our training data is a Matrix $\mathbf{X}$ and $\mathbf{Y}$ as below:

$$\mathbf{X} = \begin{bmatrix} \begin{bmatrix} \mathbf{x}_s & \mathbf{x}_{s+1} & \cdots & \mathbf{x}_{s+d} \end{bmatrix} \\ \begin{bmatrix} \mathbf{x}_{s+1} & \mathbf{x}_{s+2} & \cdots & \mathbf{x}_{s+d+1} \end{bmatrix} \\ . \\ . \\ . \\ \begin{bmatrix} \mathbf{x}_{e-d} & \mathbf{x}_{e-d+1} & \cdots & \mathbf{x}_e \end{bmatrix} \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} \mathbf{y}_{(s+d+1):k} \\ \mathbf{y}_{(s+d+2):k} \\ . \\ . \\ . \\ \mathbf{y}_{(e+1):k} \end{bmatrix}$$

5 Recurrent Neural Networks

The built RNN Function in task B5 is the same as Task B4, but in this time, an Input layer is added as Figure 3 below:

```python
def RecurrentNeuralNetworks(InputLayer,layerNums=NUMBER_OF_LAYER,
hidden_units=NUMBER_OF_HIDDEN_UNITS, loss_type=LOSS_FUNCTION,
optimizerType=OPTIMIZER, dense_unit=K_SEQUENCE, activation=["tanh",
"linear"], dropoutRate = DROP_OUT_RATE):
    model = Sequential()

    for i in range(layerNums):
        if i == (layerNums - 1):
            model.add(InputLayer)
            model.add(SimpleRNN(hidden_units, activation=activation[0]))
            model.add(Dropout(dropoutRate))
        else:
            model.add(SimpleRNN(hidden_units, activation=activation[0],
            return_sequences=True))
            model.add(Dropout(dropoutRate))

    model.add(Dense(units=dense_unit, activation=activation[1]))
    model.compile(loss=loss_type, metrics=[metrics.MeanSquaredError(),
    metrics.MeanAbsoluteError(), metrics.R2Score()],
    optimizer=optimizerType)
    return model
```
Python

Figure 3: Implementing RNN Function in Task B5

6 Long Short-term Memory and Gated Recurrent Unit

```python
def LongShortTermMemory(InputLayer, layerNums=NUMBER_OF_LAYER,
hidden_units=NUMBER_OF_HIDDEN_UNITS, loss_type=LOSS_FUNCTION,
optimizerType=OPTIMIZER, dense_unit=K_SEQUENCE, activation=["tanh",
"linear"], dropoutRate = DROP_OUT_RATE):
    model = Sequential()

    for i in range(layerNums):
        if i == (layerNums - 1):
            model.add(InputLayer)
            model.add(LSTM(hidden_units, activation=activation[0]))
            model.add(Dropout(dropoutRate))
        else:
            model.add(LSTM(hidden_units, activation=activation[0],
            return_sequences=True))
            model.add(Dropout(dropoutRate))

    model.add(Dense(units=dense_unit, activation=activation[1]))
    model.compile(loss=loss_type, metrics=[metrics.MeanSquaredError(),
    metrics.MeanAbsoluteError(), metrics.R2Score()],
    optimizer=optimizerType)
    return model
```
Python

Figure 4: Implementing LSTM Function in Task B5

```python
def GatedRucurrentUnit(InputLayer, layerNums=NUMBER_OF_LAYER,
hidden_units=NUMBER_OF_HIDDEN_UNITS, loss_type=LOSS_FUNCTION,
optimizerType=OPTIMIZER, dense_unit=K_SEQUENCE, activation=["tanh",
"linear"], dropoutRate = DROP_OUT_RATE):
    model = Sequential()

    for i in range(layerNums):
        if i == (layerNums - 1):
            model.add(InputLayer)
            model.add(GRU(hidden_units, activation=activation[0]))
            model.add(Dropout(dropoutRate))
        else:
            model.add(GRU(hidden_units, activation=activation[0],
            return_sequences=True))
            model.add(Dropout(dropoutRate))

    model.add(Dense(units=dense_unit, activation=activation[1]))
    model.compile(loss=loss_type, metrics=[metrics.MeanSquaredError(),
    metrics.MeanAbsoluteError(), metrics.R2Score()],
    optimizer=optimizerType)
    return model
```

Python

Figure 5: Implementing GRU Function in Task B5

7 Results

There are 3 evaluations in total. In each evaluation, we evaluate the predicted results from three models: RNN, LStM, GRU. Also, there are three metrics to estimate the accuracy: Mean Squared Error, Mean Absolute Error, R2 Score.

7.1 Dataset

The dataset is provided by Yahoo Finance: a CSV File including the stock price of Apple Inc. as below:

Table 1: Stock Table of Apple Inc.

Date	**Open**	**High**	**Low**	**Close**	**Adj Close**	**Volume**
2010-01-04	7.622	7.660	7.585	7.643	6.470	493729600
2010-01-05	7.664	7.699	7.616	7.656	6.481	601904800
2010-01-06	7.656	7.686	7.526	7.534	6.378	552160000

In our experiment, there are 2373 records in the **Training set**. The **Testing set** is the Stock Record of Apple Inc. from **`START_TEST_DATE`** to **`END_TEST_DATE`** (Figure 2)

7.2 Hyperparameters

In order to compare the result from 3 model: RNNs, LSTM, GRU; they must be evaluated with the same configuration:

- Three hidden layers
- Activation Function at each hidden layer is the *tanh* Function
- 80 units per layer
- Fully connected (No dropout)
- Optimizer: Adam Optimizer Algorithm
- Loss Function: Mean Squared Error
- Number of epochs: 30
- All records are scaled using MinMax Scaler
- In each epoch, 20% of the training set will be used for evaluation set

7.3 Evaluation 1

In this evaluation, the model will *learn* the Stock Record (*Open Price, High Price, Low Price, Volumes*) of the last 20 days to predict the *Close Price* of the **next 10 days**. The Figure 6 below is the output result.

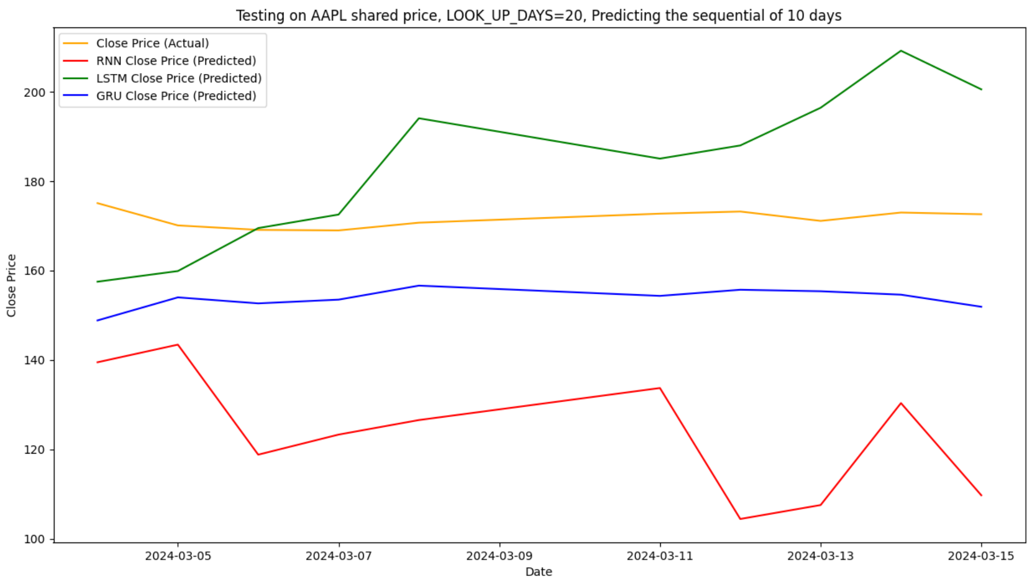

Figure 6: Predicted Result for Evaluation 1

At the last epoch, the Table 2 shows the metrics for the Evaluation 1.

Table 2: Metrics of Evaluation 1

	MSE	**MAE**	**R2**
RNN	$1.6229e^{-04}$	0.0096	0.9807
LSTM	$7.3958e^{-05}$	0.0057	0.9913
GRU	$9.1260e^{-05}$	0.0064	0.9902

7.4 Evaluation 2

In this evaluation, the model will *learn* the Stock Record (*Open Price, High Price, Low Price, Volumes*) of the last 20 days to predict the *Close Price* of the **next 20 days**. The Figure 7 below is the output result.

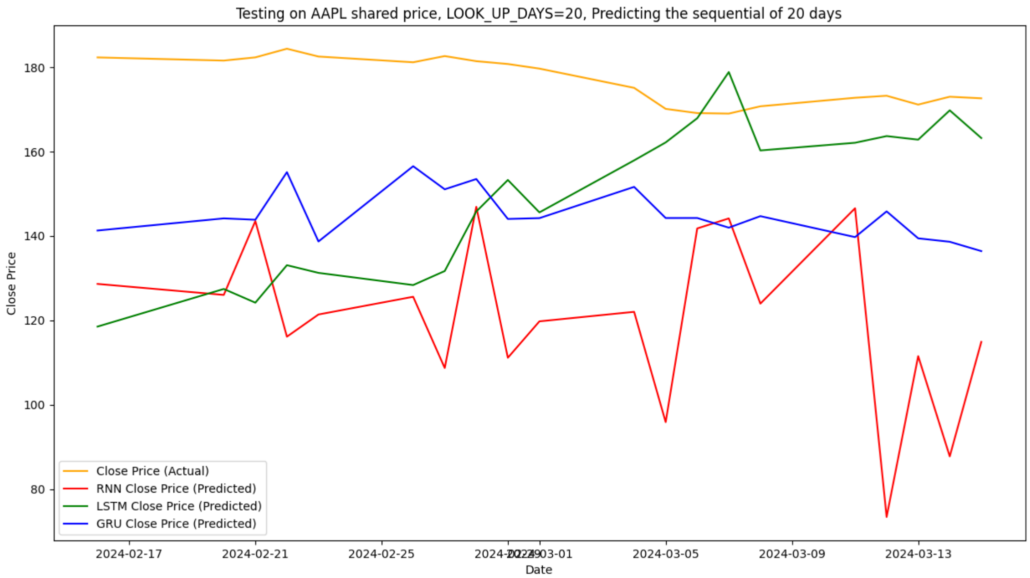

Figure 7: Predicted Result for Evaluation 2

At the last epoch, the Table 3 shows the metrics for the Evaluation 2.

Table 3: Metrics of Evaluation 2

	MSE	**MAE**	**R2**
RNN	$2.3335e^{-04}$	0.0111	0.9730
LSTM	$1.3875e^{-04}$	0.0080	0.9839
GRU	$1.5823e^{-04}$	0.0083	0.9817

7.5 Evaluation 3

In this evaluation, the model will *learn* the Stock Record (*Open Price, High Price, Low Price, Volumes*) of the last 20 days to predict the *Close Price* of the **next 30 days**. The Figure 8 below is the output result.

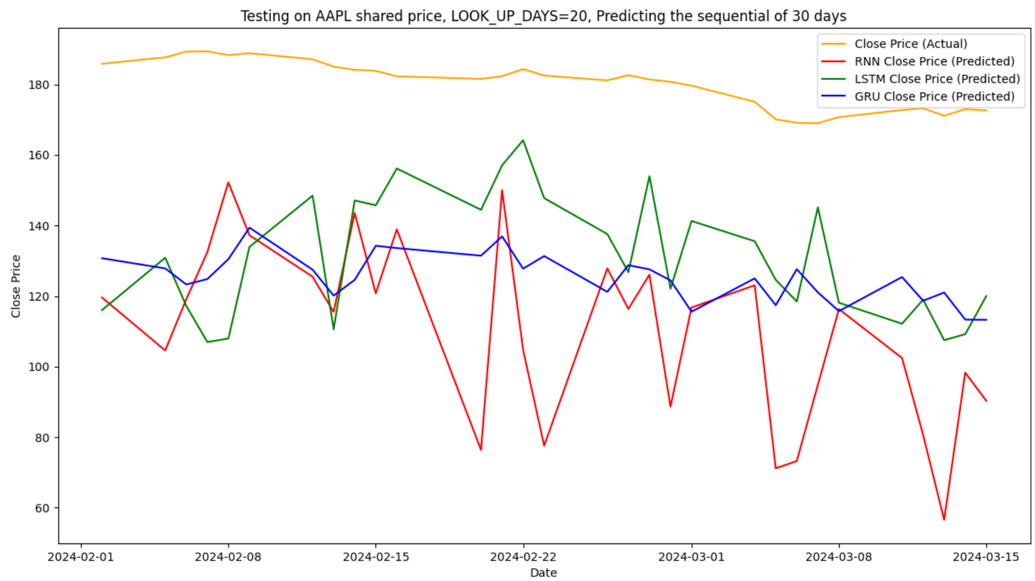

Figure 8: Predicted Result for Evaluation 3

At the last epoch, the Table 4 shows the metrics for the Evaluation 3.

Table 4: Metrics of Evaluation 3

	MSE	**MAE**	**R2**
RNN	3.2241^{-04}	0.0129	0.9624
LSTM	$1.6587e^{-04}$	0.0086	0.9807
GRU	$2.2431e^{-04}$	0.0096	0.9739

7.6 Justifications

Based on the observation on Figure 6, 7, and 8; we can admit the fact that: the more `K_SEQUENCE` is, the less accuracy we get. According to Table 2, 3, and 4, although the error scores (MSE and MAE) approach 0 and the R^2 Score approaches 1, but before the training process, we have scaled the data into the range of $[0, 1]$. So, we can not make a conclusion based on the Perfomance Metrics (Table 2, 3, and 4). The result from the RNN model from 3 evaluations is not good compared to others. This can be explained by its architecture: RNN has the basic architecture in Sequecne Model, it does not have *Gates* for further advanced prediction.

8 Conclusion

Obviously, solving the multistep prediction problem is much more challenging than single prediction. Hence, we will try to apply more machine learning techniques (Ensemble Learning) to improve the quality of the prediction.