**Appendix**

Source Code:

Data\_collection.py

def collect\_data(type="training"):

import datetime as dt

import pandas\_datareader.data as pdr

date\_entry\_start = input('Enter start date in YYYY-MM-DD format: ')

year, month, day = map(int, date\_entry\_start.split('-'))

start = dt.date(year, month, day)

date\_entry\_end = input('Enter end date in YYYY-MM-DD format: ')

year, month, day = map(int, date\_entry\_end.split('-'))

end = dt.date(year, month, day)

ticker = input('Enter company ticker name: ')

data = pdr.DataReader(ticker, 'yahoo', start, end)

if type == 'training':

data.to\_csv('training.csv')

elif type == 'testing':

data.to\_csv('testing.csv')

return

neural\_network.py

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from sklearn.preprocessing import

sc = MinMaxScaler(feature\_range=(0, 1))

def importing\_the\_training\_set\_testing\_set():

dataset\_train = pd.read\_csv('training.csv')

training\_set = dataset\_train.iloc[:, 1:5].values

output\_set = dataset\_train.iloc[:, 3:4].values

return [training\_set, output\_set]

def feature\_scaling(training\_set, output\_set):

training\_set\_scaled = sc.fit\_transform(training\_set)

output\_set\_scaled = sc.fit\_transform(output\_set)

return [training\_set\_scaled, output\_set\_scaled]

def reshaping(X\_train, rows):

return np.reshape(X\_train, (rows-1, 1, 4))

def building\_the\_rnn(X\_train, Y\_train):

# Importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import GRU

# Initialising the RNN

regressor = Sequential()

# Adding the input layer and the GRU layer

regressor.add(GRU(units=4, activation='sigmoid', input\_shape=(1, 4)))

# Adding the output layer

regressor.add(Dense(units=1))

# Compailing the RNN

regressor.compile(optimizer='adam', loss='mean\_squared\_error')

# Fitting the RNN to the Training set

regressor.fit(X\_train, Y\_train, batch\_size=32, epochs=200)

return regressor

def getting\_the\_real\_stock\_price():

dataset\_test = pd.read\_csv('testing.csv')

rows\_t, columns\_t = dataset\_test.shape

real\_stock\_price = dataset\_test.iloc[0:rows\_t-1, 1:5].values

real\_stock\_price\_output = dataset\_test.iloc[1:rows\_t, 3:4].values

real\_stock\_price\_output\_df = pd.DataFrame(real\_stock\_price\_output)

real\_stock\_price\_output\_df.to\_csv('real\_stock\_price\_output.csv')

return (

real\_stock\_price,

rows\_t,

real\_stock\_price\_output,

real\_stock\_price\_output\_df

)

def getting\_the\_predicted\_stock\_price(real\_stock\_price, rows\_t, regressor):

inputs = real\_stock\_price

inputs = sc.transform(inputs)

inputs = np.reshape(inputs, (rows\_t-1, 1, 4))

predicted\_stock\_price = regressor.predict(inputs)

predicted\_stock\_price\_output = sc.inverse\_transform(predicted\_stock\_price)

predicted\_stock\_price\_df = pd.DataFrame(predicted\_stock\_price\_output)

predicted\_stock\_price\_df.to\_csv('predicted\_stock\_price.csv')

return predicted\_stock\_price\_output, predicted\_stock\_price

def visualizing\_the\_results(

real\_stock\_price\_output,

predicted\_stock\_price\_output

):

plt.plot(real\_stock\_price\_output, color='red', label='Real Stock Price')

plt.plot(

predicted\_stock\_price\_output,

color='blue',

label='Predicted Stock Price')

plt.title('Stock Price Prediction')

plt.xlabel('Time')

plt.ylabel('Stock Price')

plt.legend()

plt.show()

plt.savefig('stock\_market\_prediction.png')

def making\_predictions\_and\_visualizing\_results(regressor):

real\_stock\_price, rows\_t, real\_stock\_price\_output,\

real\_stock\_price\_output\_df = getting\_the\_real\_stock\_price()

predicted\_stock\_price\_output, predicted\_stock\_price = \

getting\_the\_predicted\_stock\_price(

real\_stock\_price,

rows\_t,

regressor

)

visualizing\_the\_results(

real\_stock\_price\_output,

predicted\_stock\_price\_output

)

evaluating\_the\_rnn(

real\_stock\_price\_output\_df,

real\_stock\_price\_output,

predicted\_stock\_price

)

def evaluating\_the\_rnn(

real\_stock\_price\_output\_df,

real\_stock\_price\_output,

predicted\_stock\_price

):

import math

from sklearn.metrics import mean\_squared\_error

mean\_real\_stock\_price\_output\_df = real\_stock\_price\_output\_df.mean()

rmse = math.sqrt(

mean\_squared\_error(real\_stock\_price\_output, predicted\_stock\_price)

)

rmse\_percentage = rmse/mean\_real\_stock\_price\_output\_df

print(rmse\_percentage)

app.py

from data\_collection import collect\_data

from neural\_network import (

importing\_the\_training\_set\_testing\_set,

feature\_scaling,

reshaping,

building\_the\_rnn,

making\_predictions\_and\_visualizing\_results

)

# collect data for training

collect\_data(type="training")

# collect data for testing

collect\_data(type="testing")

training\_set, output\_set = importing\_the\_training\_set\_testing\_set()

# counting rows

rows, columns = training\_set.shape

training\_set\_scaled, output\_set\_scaled = feature\_scaling(

training\_set, output\_set)

# Getting the inputs and the outputs

X\_train = training\_set\_scaled[0:rows-1, :]

Y\_train = output\_set\_scaled[1:rows, :]

X\_train = reshaping(X\_train, rows)

# Part 2: Building the RNN

regressor = building\_the\_rnn(X\_train, Y\_train)

# Part 3:

making\_predictions\_and\_visualizing\_results(regressor)