ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

A Deep Neural Network Based Quantitative Strategy for the CSI 300 Index

Yuzhe Fang*

School of Economics, Shanghai University, Shanghai, China
fangyuzhe1998@163.com
**Corresponding author*

Abstract: *The four core financial technologies of Artificial Intelligence (AI), Blockchain, Cloud computing, and Big Data have laid a solid foundation for the financial industry to move towards large-scale "quantitative" and "automated" scenario applications. Nowadays, quantitative trading has been an important research direction in FinTech. This paper, based on the Deep Neural Network in machine learning, input data of opening price, closing price, high price, low price, volume, yield, first-order difference of yield, second-order difference of yield and MACD to predict the rise and fall of CSI 300 index on the next day.*

Keywords: *Fintech, Quantitative Investment, Machine Learning, Deep Neural Network.*

1. Introduction

Deep Neural Network was introduced by Prof. Hinton in 2006, which is usually a neural network structure with multiple implicit layers, and its training process is usually based on abstracting feature data layer by layer, and then tuning the whole network model. Deep Neural Networks contain more implicit layers and implicit nodes, which can be used to obtain better feature representation by layer-by-layer abstraction and learning, so that it can better learn the essential features of the data, and thus can improve the prediction accuracy of the model.

Due to the powerful learning ability of Deep Neural Networks, many scholars have started to apply them to the study of financial data in recent years, however, due to the high noisiness and randomness of financial data, the gap between the effectiveness of their application is more obvious compared to that of image recognition and other fields. The prediction of financial asset price trend is a challenging task in itself, and most of the existing models have mediocre results. Considering the features of deep neural networks, which do not rely on a priori knowledge and extract features from a large amount of raw data, it still has great potential for the study of financial data. Therefore, it is a very necessary research direction to further explore the application of Deep Neural Networks in quantitative investment, and the research in this paper is born in this context.

With the development of quantitative trading in recent years, quantitative trading researchers have also researched many analytical frameworks for quantitative trading by combining knowledge from different fields. In the research of nonlinear timing of quantitative trading, Sidorowich and Farmer introduced dynamics and wavelet theory into the time series analysis framework for timing analysis; Mandelbrot introduced the fractal theory in topology into the analysis of security prices, which laid the foundation for the mainstream Hurst index analysis framework nowadays; Suraphan Thawornwong and David Enke utilized artificial neural networks to analyze financial asset indicators and obtained better results.

Xiong et al. (2008) used Google's domestic trends as an indicator of public sentiment and macroeconomic factors, and applied a long-term short-term memory neural network to simulate the volatility of the S&P 500 index, and the study showed that Deep Neural Networks have great potential for understanding stock behavior. Dixon et al. (2015) used Deep Neural Networks to study price movements in futures markets, and the study 45 commodity futures were selected as the subject of study and the average prediction accuracy of the experiment reached 73%. Dixon et al. (2016) selected the 5-minute prices of 43 commodity futures and foreign exchange as the subject of study, used deep neural network models to make predictions on the direction of price movements, and used the predictions to build a quantitative investment strategy, which achieved an annualized Sharpe ratio of

ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

3.29.

Since machine learning can fit the variation of nonlinear systems better than traditional econometric models. Therefore, it has better results than traditional models in a complex system like securities investment. Therefore, in the foreseeable future, machine learning and portfolio investment will be more closely integrated.

This study aims to use deep neural networks to predict the rise and fall of the next day based on the opening price, closing price, high price, low price, volume, yield, first-order difference of yield, second-order difference of yield, and MACD indicators of the previous 5 days of the CSI 300 index.

2. Machine Learning Model

2.1. The Basic Concept and Common Model of Neural Network

A neural network is a mesh structure composed of a large number of neurons. The neurons in the upper layer are weighted and summed by the activation function to generate the neurons in the next layer. Each neuron is multiplied by its corresponding weight and linked to a new neuron subject. The output values of these neurons are summed and added to a bias value, and then fed to the next layer of neurons via an activation function. A large number of these structures are put together to form a vast neural network are shown in Figures 1.

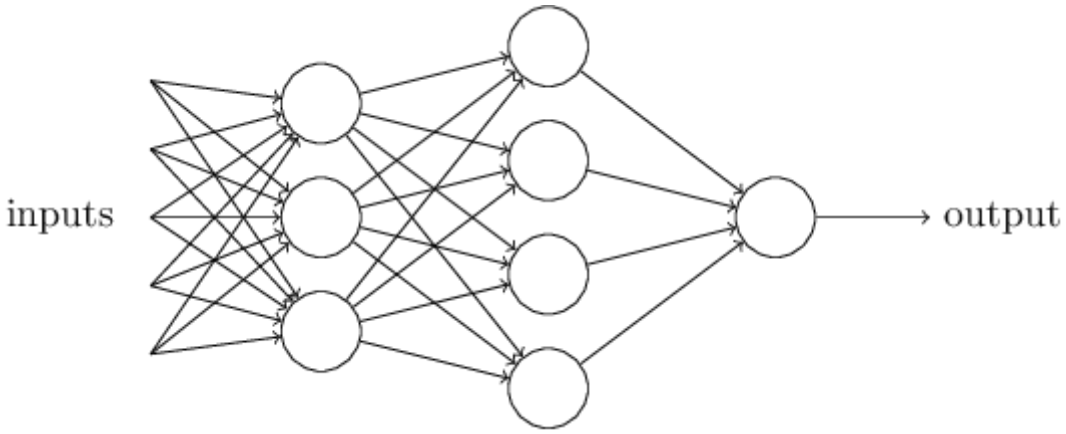

Figure 1: Schematic diagram of neural network layer

2.2. Training Set and Test Set

Since most machine learning models are highly complex and have a large number of parameters, such models can often be highly fitted to experimental data, which can lead to overfitting problems, so that the overall generalization and inference ability of the model is insufficient. Therefore, in machine learning, the entire dataset is generally divided into two parts, i.e., the training set and the test set. The training of the model is performed on the training set, and the test set is used to evaluate whether the model as a whole is overfitted.

2.3. Overview of Deep Neural Networks

Usually, Deep Neural Networks have an input layer, an implicit layer and an output layer, where the implicit layer is usually more than one layer. Compared with shallow neural networks, Deep Neural Networks have two advantages: first, for some problems lack of depth will lead to insufficient learning, deep neural networks solve this problem; second, the cognitive process is a layer-by-layer, layer-by-layer abstraction process, Deep Neural Networks have such characteristics.

So how does a Deep Neural Network learn?

Hinton's proposed deep neural network approach is divided into two steps: the first step is the pre-training phase, which abstracts the data layer by layer through unsupervised learning; the second step is the overall tuning phase, which tunes the entire deep neural network model through the BP algorithm.

ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

3. Machine Learning Data Filtering and Model Building

3.1. Data Selection and Pre-Processing

The data used for the machine learning models in this section are data downloaded from tushare. In this paper, the time span of the data for the machine learning model is from March 10, 2015 to March 11, 2021, and the time span of the quantitative back-test is from April 12, 2019 to March 12, 2021. For the selection of stocks, CSI 300, the core asset of A-share market, is selected as the benchmark in this paper. For the selection of data dimensions, this paper selects the following dimensions for each stock for the past 5 days.

Opening price, closing price, high price, low price, volume, return, first-order difference of return, second-order difference of return and MACD.

The total input dimension is 45 dimensions.

The output dimension is 2-dimensional, with the next day's rise recorded as 1 and the next day's fall recorded as 0.

In this paper, we set the mode to change the learning rate during the learning process. The initial learning rate is set to 1, but when epoch > 100, the learning rate changes as follows.

$$\text{Learn rate} = \text{alpha} / (1 + 0.0008 * (\text{epoch} - 100)) \quad (1)$$

3.2. Quantified Back Test Data Sources and Simple Trading Strategies

The data used in this section of back-testing are from the data downloaded on tushare.

Since the mainstream quantitative trading platforms (e.g. Jukuan, Umini) lack the pytorch module, it is not possible to perform machine learning quantitative back-testing on the platform. Therefore, this paper wrote a quantitative back-testing software using python for local back-testing.

The trading strategy in this paper is relatively simple, if the model predicts that the probability of the next day's rise is greater than 0.95, then buy long, if the model predicts that the probability of the next day's fall is greater than 0.95, then sell short. The entire principal amount is traded unleveraged each day, and then the return is calculated cumulatively.

3.3. Trading Strategy Back Test Results

Using the back-testing model framework in the previous section, this paper back-tested the CSI 300 separately, and the results of the three back-tests are shown in Figures 2 to 4, respectively (the black line is the return of the strategy, and the blue line is the return of the CSI 300).

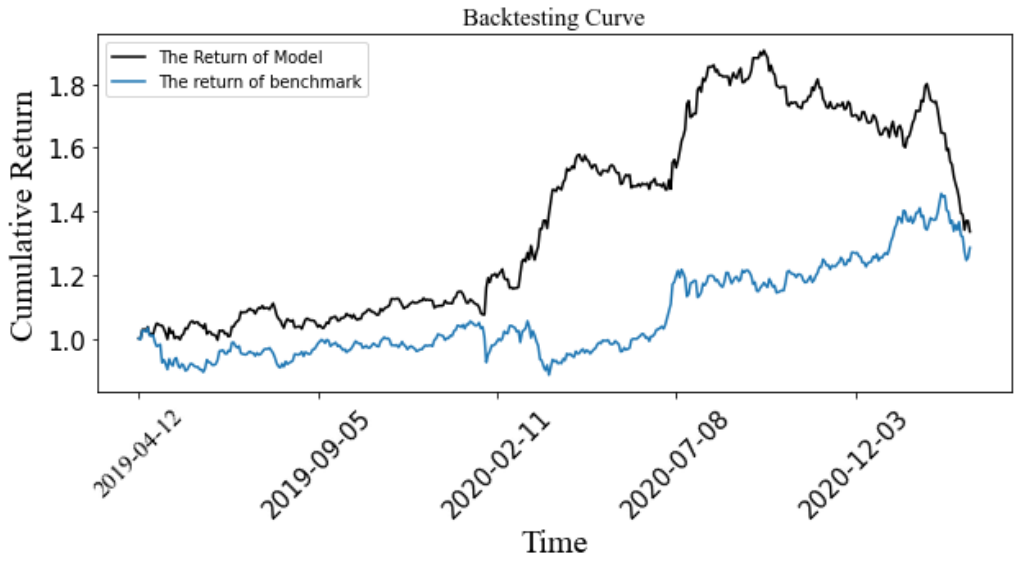

Figure 2: The first back-test yield comparison chart

ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

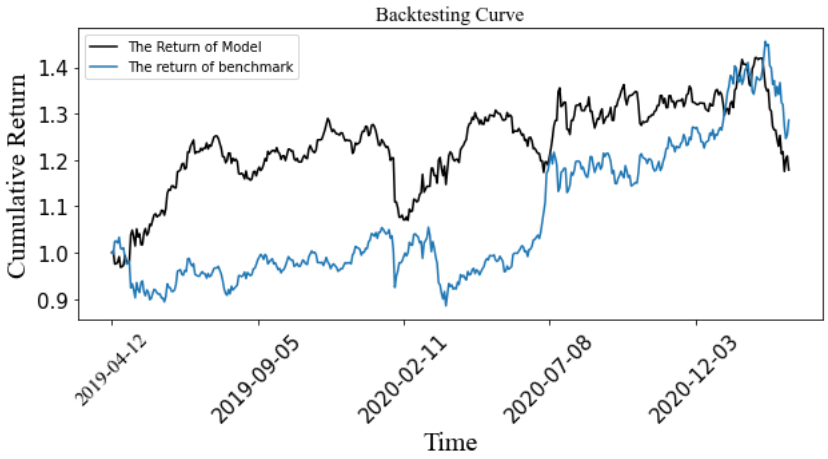

Figure 3: The Second back-test yield comparison chart

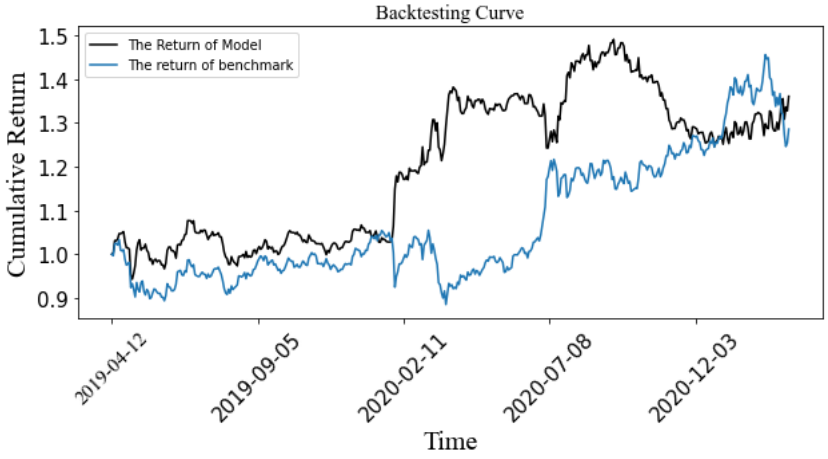

Figure 4: The Third back-test yield comparison chart

As can be seen from the back-test return comparison chart, although the overall return is reasonable, the model in this paper has a large retracement and is prone to larger losses in extreme quotes, which needs to be improved.

4. Conclusions and Prospects

The paper constructs a simple quantitative strategy using technical analysis as the core indicator and deep neural networks in machine learning models. After building the quantitative strategy with the predicted results, this paper get basically satisfactory results, but there are still many shortcomings.

However, there are still some parts of this paper that are worth improving. First, the accuracy of the machine learning model prediction is still worth improving, which can reduce the risk of the whole strategy stepping short, or not closing the position in time. For that problem, the author should try to actively explore more factors and incorporate them into the machine learning in order to obtain a better fit of the nonlinear relationship.

Then, the Deep Neural Network taken in this paper is not the most cutting-edge machine learning model at present, and the author will continue to learn and introduce more models into the FinTech market.

In addition, for the construction of quantitative strategies, more decision factors can be added so as to circumvent the problem of model failure in certain market environments.

Finally, due to the limitation of computing resources, the author is unable to extend the strategy to more stocks in order to obtain conclusions with broader applicability. This is also something that deserves further improvement in the future.

ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

References

[1] Dixon M, Klabjan D, Bang J H. Classification-based Financial Markets Prediction using Deep Neural Networks. Social Science Electronic Publishing, 2016
[2] Dixon M, Klabjan D, Jin H B. Implementing deep neural networks for financial market prediction on the Intel Xeon Phi// The Workshop on High PERFORMANCE Computational Finance. ACM, 2015:1-6
[3] Dunis C L, Nathani A. Quantitative trading of gold and silver using nonlinear models. Neural Network World, 2007, 17(2):93-111
[4] Shambora W E, Rossiter R. Are there exploitable inefficiencies in the futures market for oil? Energy Economics, 2007, 29(1):18-27
[5] Xiong R, Nichols E P, Shen Y. Deep Learning Stock Volatilities with Google Domestic Trends. Computer Science, 2008, 10(2):1:7
[6] Sayyed Abdolmajid Jalaee, Mehrdad Lashkary, Amin GhasemiNejad The Phillips curve in Iran: econometric versus artificial neural networks [J] Heliyon, 2019, 5(8)
[7] Rashmi Malhotra, D.K Malhotra Evaluating consumer loans using neural networks [J] Omega, 2003, 31(2)
[8] Haiyan Mo, Jun Wang, Hongli Niu Exponent back propagation neural network forecasting for financial cross-correlation relationship[J] Expert Systems With Applications, 2016, 53
[9] Anonymous DATAMONITOR: Cloudy forecast for Salesforce.com's on-demand service initiative; Will social networking drive customer service in the future?[J] M2 Presswire, 2009
[10] Wiz Maps is developing unique forecasting models and mobile data mapping technology that fixes the huge disconnect and risk between a property and the surrounding environment, which helps real estate professionals and businesses[J] M2 Presswire, 2016

Appendix

```
import torch
from torch.autograd import Variable
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
import pandas as pd
import talib
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from torchvision import datasets
from torchvision import transforms
import tensorflow as tf
from tensorflow import keras
import sympy
from sympy import Matrix
import tushare as ts

#hyber parameters
alpha = 1

# %config InlineBackend.figure_format = 'svg'

data = ts.get_k_data('hs300',start = '2015-01-01',end='2021-03-12')
data.set_index('date',inplace = True)
data

data_cleaned = pd.DataFrame()
data_cleaned['Close0'] = np.log(data['close'])
data_cleaned['H_L'] = data['high'] - data['low']
data_cleaned['C_O'] = data['close'] - data['open']
data_cleaned['volume'] = np.log10(data['volume'])

data_cleaned['Close1']=data_cleaned['Close0'].shift(1)
```

ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

```
data_cleaned['Close2']=data_cleaned['Close0'].shift(2)
data_cleaned['Close3']=data_cleaned['Close0'].shift(3)
data_cleaned['Close4']=data_cleaned['Close0'].shift(4)
data_cleaned['Close5']=data_cleaned['Close0'].shift(5)

data_cleaned['H_L1']=data_cleaned['H_L'].shift(1)
data_cleaned['H_L2']=data_cleaned['H_L'].shift(2)
data_cleaned['H_L3']=data_cleaned['H_L'].shift(3)
data_cleaned['H_L4']=data_cleaned['H_L'].shift(4)
data_cleaned['H_L5']=data_cleaned['H_L'].shift(5)

data_cleaned['C_01']=data_cleaned['C_O'].shift(1)
data_cleaned['C_02']=data_cleaned['C_O'].shift(2)
data_cleaned['C_03']=data_cleaned['C_O'].shift(3)
data_cleaned['C_04']=data_cleaned['C_O'].shift(4)
data_cleaned['C_05']=data_cleaned['C_O'].shift(5)

data_cleaned['volume1']=data_cleaned['volume'].shift(1)
data_cleaned['volume2']=data_cleaned['volume'].shift(2)
data_cleaned['volume3']=data_cleaned['volume'].shift(3)
data_cleaned['volume4']=data_cleaned['volume'].shift(4)
data_cleaned['volume5']=data_cleaned['volume'].shift(5)

#5 day yield
data_cleaned['return1']=(data_cleaned['Close0']-data_cleaned['Close0'].shift(-1)).shift(1)
data_cleaned['return2']=(data_cleaned['Close0']-data_cleaned['Close0'].shift(-1)).shift(2)
data_cleaned['return3']=(data_cleaned['Close0']-data_cleaned['Close0'].shift(-1)).shift(3)
data_cleaned['return4']=(data_cleaned['Close0']-data_cleaned['Close0'].shift(-1)).shift(4)
data_cleaned['return5']=(data_cleaned['Close0']-data_cleaned['Close0'].shift(-1)).shift(5)

#5 day yield first order difference
data_cleaned['return_ood1']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(1)
data_cleaned['return_ood2']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(2)
data_cleaned['return_ood3']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(3)
data_cleaned['return_ood4']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(4)
data_cleaned['return_ood5']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(5)

#5 day yield second order difference
data_cleaned['return_sod1']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(1)
data_cleaned['return_sod2']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(2)
data_cleaned['return_sod3']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(3)
data_cleaned['return_sod4']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(4)
data_cleaned['return_sod5']=(data_cleaned['return1']-data_cleaned['return1'].shift(-1)).shift(5)

#5 day MACD
data_cleaned['macd1']=talib.MACD\
    (data_cleaned['Close0'], fastperiod=9, slowperiod=12, signalperiod=26)[2].shift(1)
data_cleaned['macd2']=talib.MACD\
    (data_cleaned['Close0'], fastperiod=9, slowperiod=12, signalperiod=26)[2].shift(2)

data_cleaned['macd3']=talib.MACD\
    (data_cleaned['Close0'], fastperiod=9, slowperiod=12, signalperiod=26)[2].shift(3)

data_cleaned['macd4']=talib.MACD\
    (data_cleaned['Close0'], fastperiod=9, slowperiod=12, signalperiod=26)[2].shift(4)

data_cleaned['macd5']=talib.MACD\
    (data_cleaned['Close0'], fastperiod=9, slowperiod=12, signalperiod=26)[2].shift(5)
```

ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

```
data_cleaned['Close0_shift(-1)'] = data_cleaned['Close0'].shift(-1)
data_cleaned['label'] = 0.5 + 0.5 * np.sign(data_cleaned['Close0_shift(-1)'] - data_cleaned['Close0'])
#np.sign()


data_cleaned.dropna(inplace = True)
data_cleaned.drop(['Close0_shift(-1)'], axis = 1, inplace = True)
data_cleaned


#Begin to train
training_set = data_cleaned.iloc[:1000,:]
test_set = data_cleaned.iloc[1000:,:]

#Train the model
class DeepNeuraNetworkModel(nn.Module):
    def __init__(self):
        super(DeepNeuraNetworkModel, self).__init__()
        self.FC_layer1 = nn.Linear(44, 88)
        self.FC_layer2 = nn.Linear(88, 44)
        self.FC_layer3 = nn.Linear(44, 24)
        self.FC_layer4 = nn.Linear(24, 12)
        self.FC_layer5 = nn.Linear(12, 2)
    def forward(self, input_data):    #dim of input_data:N*24
        z1_ = self.FC_layer1(input_data)
        z1 = torch.sigmoid(z1_)

        z2_ = self.FC_layer2(z1)
        z2 = torch.sigmoid(z2_)

        z3_ = self.FC_layer3(z2)
        z3 = torch.sigmoid(z3_)

        z4_ = self.FC_layer4(z3)
        z4 = torch.sigmoid(z4_)

        z5_ = self.FC_layer5(z4)

        return z5_

X = training_set.iloc[:,0:44]
y = training_set.iloc[:,44]
X = np.array(X)
y = np.array(y)
X

X = Variable(torch.FloatTensor(X))
y = Variable(torch.LongTensor(y))
X.shape


DNN_Model = DeepNeuraNetworkModel()
optimizer = torch.optim.SGD(DNN_Model.parameters(),lr= alpha)
loss_function = nn.CrossEntropyLoss()

# Change learning rate
def adjust_learning_rate(optimizer, epoch):
    if epoch <= 100:
        lr = alpha
```

ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

```
    elif epoch > 100:
        lr = alpha / (1 + 0.0008* (epoch - 100))

    for param_group in optimizer.param_groups:
        param_group['lr'] = lr

Iter_times = 20000
loss_list = []
for i in range(Iter_times):
    outputs = DNN_Model.forward(X) #forward propagation
    loss = loss_function(outputs,y)
    loss.backward() #backward propagation
    optimizer.step() #update parameters
    optimizer.zero_grad() #reset grad to 0
    if (i+1)%500 == 0:
        print(i+1,'iterations have been completed!')
        print('          -> Now loss=', loss)
        print('=====================================================')
    adjust_learning_rate(optimizer, i)
    loss_list.append(loss)
    if loss < 0.0005:
        break

Probability_Calculator= nn.Softmax(dim=1)
pred =[]
prob = Probability_Calculator(DNN_Model.forward(X)).detach().numpy()

for i in range(prob.shape[0]):
    pred.append(np.argmax(prob[i,:]))
pred = np.array(pred)
pred.shape
accuracy_score(pred,y)

X = test_set.iloc[:,0:44]
y = test_set.iloc[:,44]
X = np.array(X)
y = np.array(y)
X = Variable(torch.FloatTensor(X))
y = Variable(torch.LongTensor(y))
pred = []
prob = Probability_Calculator(DNN_Model.forward(X)).detach().numpy()

for i in range(prob.shape[0]):
    if np.max(prob[i,:]) >= 0.95:
        pred.append(np.argmax(prob[i,:]))
    else:
        pred.append(np.nan)
pred = np.array(pred)
pred

hs300 = np.exp(test_set['Close0'])
pred = 2 * pred -1

rtn_temp = 1.0
cum_rtn = []
cum_rtn.append(rtn_temp)

for i in range(pred.shape[0]-1):
    if pred[i] ==1 or pred[i] == -1:
        rtn_temp = rtn_temp*(1.0+((hs300[i+1]-hs300[i])/hs300[i])*pred[i])
```

ISSN 2706-655X Vol.3, Issue 10: 84-92, DOI: 10.25236/IJFET.2021.031011

```
        else:
            rtn_temp = rtn_temp
        cum_rtn.append(rtn_temp)

cum_rtn = np.array(cum_rtn)
cum_rtn

time = test_set.index

fontTNR = 'Times New Roman'
plt.figure(figsize = (10,4))
plt.plot(time,cum_rtn,color='black',label='The Return of Model')
#Benchmark
plt.plot(time,np.exp(test_set["Close0"])/np.exp(test_set["Close0"][0]),label='The return of benchmark')
plt.xticks(test_set.index[::100],rotation=45)
plt.grid

plt.title('Backtesting Curve',fontsize = 15, fontproperties = fontTNR)
plt.xlabel('Time', fontsize = 20, fontproperties = fontTNR)
plt.ylabel('Cumulative Return', fontsize = 20, fontproperties = fontTNR)
plt.legend()
plt.xticks(fontsize = 15,fontproperties =fontTNR)
plt.yticks(fontsize = 15,fontproperties =fontTNR)
plt.show()
```