香　港　大　學

THE UNIVERSITY OF HONG KONG

Final Report

Course Info: COMP 4801 Final Year Project

Project Title: Financial Market Prediction by Deep Learning Neural Networks

Students: 3035142596 LIU Jiayao

Date of Submission: 15/04/2018

Abstract

In recent years, traditional machine learning models such as linear regression and logistic regression have been extensively applied in the financial market. Deep learning with its contributions to Artificial Intelligence has also drawn researchers and investors to utilize it to predict stock price movement. This report explores the application of the traditional machine learning models including linear regression, logistic regression, support vector machine (SVM) in conjunction with multilayer perceptron (MLP) and convolutional neural network (CNN) to predict the value and trend of Dow Jones Industrial Average. A qualitative objective is to evaluate the MSE of regression models, while a quantitative objective is to achieve accuracy above the larger percentage of the positive or negative samples for classification models. The technical difficulty lies in determining the optimal hyperparameters of the neural network models. Mock trading is conducted on the constructed models for 10 business days. An analysis of the performance and limitations of the models is presented in the report.

Acknowledgement

We would like to express our very great appreciation to Dr. Kwok-Ping Chan, our project supervisor, who offered constructive suggestions and provided a GPU to the development of this final year project.

Table of Contents

List of Figures 6

List of Tables 7

Abbreviations 9

1. Introduction 10

1.1 Background Information 10

1.1.1 Machine Learning 10

1.1.2 Financial Market and Dow Jones Industrial Average 11

1.2 Business Goal and Technical Objective 11

1.3 Related Work 12

1.4 Outline of the Report 13

2. Methodology 13

2.1 Binary Classification Problem 14

2.2 Regression Problem 14

2.3 Data Collection 14

2.4 Data Preprocessing 15

2.5 Machine Learning Models 19

2.5.1 Traditional Machine Learning Models 20

2.5.1.1 Linear Regression 20

2.5.1.2 Logistic Regression 20

2.5.1.3 Support Vector Machine 21

2.5.2 Deep Learning 21

2.5.2.1 Multilayer Perceptron 22

2.5.2.2 Convolutional Neural Network 22

2.5.2.2.1 Convolutional Layer 23

2.5.2.2.2 Pooling Layer 23

2.6 Model Training 24

2.6.1 Hyperparameters and Parameters 25

2.6.1.1 Traditional Learning Models 25

2.6.1.2 Neural Network Models 25

2.6.2 Backpropagation of Neural Networks 26

2.6.3 Tuning the Hyperparameters 26

2.7 Assessment of the Model 27

2.8 Software Setup **28**
3. Results and Findings **29**
3.1 Regression Models **29**
3.1.1 Linear Regression **29**
3.1.1.1 Lasso Regression (L1 norm) **30**
3.1.1.2 Ridge Regression (L2 norm) **31**
3.1.2 Multilayer Perceptron (Regression) **32**
3.1.2.1 MLP-R-Model 1 **32**
3.1.2.2 MLP-R-Model 2 **33**
3.1.2.3 MLP-R-Model 3 **34**
3.2 Classification Models **34**
3.2.1 Logistic Regression **34**
3.2.1.1 Logistic Regression with L2 Penalty **35**
3.2.1.2 Logistic Regression with L1 Penalty **36**
3.2.2 Support Vector Machine **37**
3.2.3 Multilayer Perceptron (Classification) **38**
3.2.3.1 MLP-C-Model 1 **38**
3.2.3.2 MLP-C-Model 2 **39**
3.2.3.3 MLP-C-Model 3 **39**
3.2.4 1D Convolutional Neural Network (Univariate) **40**
3.2.4.1 Conv-UC-Model 1 **40**
3.2.4.2 Conv-UC-Model 2 **41**
3.2.4.3 Conv-UC-Model 3 **41**
3.2.5 1D Convolutional Neural Network (Multivariate) **42**
3.2.5.1 Conv-MC-Model 1 **42**
3.2.5.2 Conv-MC-Model 2 **43**
3.2.5.3 Conv-MC-Model 3 **43**
3.3 Summary of Results **44**
3.3.1 Regression Models **44**
3.3.1.1 Linear Regression **44**
3.3.1.2 Multilayer Perceptron (Regression) **44**
3.3.2 Classification Models **44**
3.3.2.1 Logistic Regression **44**

3.3.2.2 Support Vector Machine **44**
3.3.2.3 Multilayer Perceptron (Classification) **45**
3.3.2.4 1D Convolutional Neural Network (Univariate) **45**
3.3.2.5 1D Convolutional Neural Network (Multivariate) **45**
4. Mock Trading **45**
4.1 Regression Models **46**
4.1.1 Linear Regression **46**
4.1.2 Multilayer Perceptron (Regression) **46**
4.2 Classification Models **47**
4.2.1 Logistic Regression **47**
4.2.2 Support Vector Machine **47**
4.2.3 Multilayer Perceptron (Classification) **48**
4.2.4 1D Convolutional Neural Network (Univariate) **48**
4.2.5 1D Convolutional Neural Network (Multivariate) **49**
4.3 Summary of Mock Trading Results **50**
5. Discussions and Analysis **50**
5.1 Evaluations and Explanations **50**
5.2 Limitations and Difficulties **51**
6. Future Improvement and Recommendations **51**
7. Conclusion **52**
References **53**

List of Figures

Figure 2.1 Three technical procedures of the project 13
Figure 2.2 Five technical procedures of data preprocessing 16
Figure 2.3 A regular three-layer neural network 22
Figure 2.4 How the data from image pixels are recognized as patterns by the convolutional neural network. 22
Figure 2.5 How an image with three channels (RGB) is processed by the filters in the convolutional layer 23
Figure 2.6 Maxpooling with a filter of size 2x2 and a stride of 2 24
Figure 2.7 Demonstration regarding fitting model on the datasets 26
Figure 2.8 How the technical procedures are implemented in Python 28
Figure 3.1 MSE of training and test sets by lasso regression with varying alpha from 0.1 to 0.9 30
Figure 3.2 MSE of training and test sets by ridge regression with varying alpha from 1 to 99 31
Figure 3.3 The first 100 predicted prices against real prices by lasso regression with alpha = 0.02 32
Figure 3.4 The first 100 predicted prices against real prices by MLP-R-Model 1 33
Figure 3.5 Accuracy of training and test sets by logistic regression with L2 penalty with C varying from 0.1 to 2.0 35
Figure 3.6 Accuracy of logistic regression with L1 penalty with C varying from 0.1 to 2.0 36

List of Tables

Table 2.1 Breakdown of the six variables in the input dataset 14
Table 2.2 Illustration of the first three samples of Univariate models 16
Table 2.3 Illustration of the first sample of Multivariate model 17
Table 2.4 Illustration of the first three samples of Univariate models after standardization 18
Table 2.5 Illustration of the first sample of Multivariate model after standardization 18
Table 2.6 Sample size of training and test sets 19
Table 2.7 Distribution of positive and negative samples 19
Table 2.8 Four scenarios of the accuracy of training dataset and the accuracy of validation dataset 27
Table 3.1 MSE of training and test sets by varying alpha from 0.01 to 0.09 by lasso regression 30
Table 3.2 MSE of training and test sets without regularization by linear regression 30
Table 3.3 Detailed structure and hyperparameters of MLP-R-Model 1 32
Table 3.4 MSE of training and test sets by MLP-R-Model 1 33
Table 3.5 Detailed structure and hyperparameters of MLP-R-Model 2 33
Table 3.6 MSE of training and test sets by MLP-R-Model 2 34
Table 3.7 Detailed structure and hyperparameters of MLP-R-Model 3 34
Table 3.8 MSE of training and test sets by MLP-R-Model 3 34
Table 3.9 Accuracy of training and test sets by logistic regression with L1 and L2 penalty with default C=1.0 34
Table 3.10 Accuracy of training and test sets by logistic regression with L2 penalty with C varying from 0.01 to 0.1 35
Table 3.11 Accuracy of training and test sets by logistic regression with L1 penalty with C varying from 1.3 to 1.6 37
Table 3.12 Accuracy of training and test sets with linear kernel and RBF kernel 37
Table 3.13 Accuracy of training and test sets with RBF kernel by grid search in C_range and gamma_range 37
Table 3.14 Detailed structure and hyperparameters of MLP-C-Model 1 38
Table 3.15 Accuracy and loss of training and test sets by MLP-C-Model 1 39
Table 3.16 Detailed structure and hyperparameters of MLP-C-Model 2 39

Table 3.17 Accuracy and loss of training and test sets by MLP-C-Model 2 39
Table 3.18 Detailed structure and hyperparameters of MLP-C-Model 3 39
Table 3.19 Accuracy and loss of training and test sets by MLP-C-Model 3 40
Table 3.20 Detailed structure and hyperparameters of Conv-UC-Model 1 (BatchNormalization omitted) 40
Table 3.21 Accuracy and loss of training and test sets by Conv-UC-Model 1 40
Table 3.22 Detailed structure and hyperparameters of Conv-UC-Model 2 (BatchNormalization omitted) 41
Table 3.23 Accuracy and loss of training and test sets by Conv-UC-Model 2 41
Table 3.24 Detailed structure and hyperparameters of Conv-UC-Model 3 (BatchNormalization omitted) 41
Table 3.25 Accuracy and loss of training and test sets by Conv-UC-Model 3 42
Table 3.26 Detailed structure and hyperparameters of Conv-MC-Model 1 (BatchNormalization, Flatten omitted) 42
Table 3.27 Accuracy and loss of training and test sets by Conv-MC-Model 1 42
Table 3.28 Detailed structure and hyperparameters of Conv-MC-Model 2 (BatchNormalization, Flatten omitted) 43
Table 3.29 Accuracy and loss of training and test sets by Conv-MC-Model 2 43
Table 3.30 Detailed structure and hyperparameters of Conv-MC-Model 3 (BatchNormalization, Flatten omitted) 43
Table 3.31 Accuracy and loss of training and test sets by Conv-MC-Model 3 44
Table 3.32 Mock trading results for linear regression model 46
Table 3.34 Mock trading results for MLP-R-Model 1 46
Table 3.34 Mock trading results for logistic regression model 47
Table 3.35 Mock trading results for SVM model 47
Table 3.36 Mock trading results for MLP-C-Model 1 48
Table 3.37 Mock trading results for Conv-UC-Model 1 48
Table 3.38 Mock trading results for Conv-MC-Model 1 49
Table 3.39 Summary of mock trading results in descending order of gain 49

Abbreviations

CNN	Convolutional neural network
DJIA	Dow Jones Industrial Average
MLP	Multilayer Perceptron
SVM	Support Vector Machine
LSE	Least Squares Estimation
SSE	Sum of Squared Error
MSE	Mean Squared Error
MLE	Maximum Likelihood Estimation
RBF	Radial Basis Function
RSI	Relative Strength Index

1. Introduction

The introduction part will first present some background information of our project including basic knowledge with respect to machine learning, especially in relation to deep learning. In addition, some basic information regarding the financial market and the predicted objective of the project – Dow Jones Industrial Average will also be introduced. Then the business goal and the technical objective will be discussed followed by an outline of the remainder of the report.

1.1 Background Information

1.1.1 Machine Learning

Because of the increasing quantity of and accessibility to data and the augmenting computation power of machines, nowadays stock prediction models that combine big data and machine learning models are employed to identify any arbitrage opportunity in the stock market. Common machine learning classification models to predict the movement of stock price include linear regression, logistic regression, SVM etc.

Deep learning, as a class of machine learning, has contributed substantially to Artificial Intelligence. It has been considerably applied in image recognition, speech processing and even in games such as AlphaGo. In contrast to the traditional machine learning algorithms, deep learning possesses the advantage to identify the features of data itself and excels at exploring the complex relationship, especially nonlinear relationship, among the variables while machine learning models produce the output by mainly studying how to combine the features that are already formed by humans (Ketkar, 2017). Therefore, compared to traditional machine learning models, neural network models reduce feature engineering.

There is a growing trend to apply deep learning to stock prediction, and we have found that Multilayer Perceptron (MLP) are utilized more frequently than other types of neural networks.

In this project, the performance of the traditional machine learning models including

linear regression, logistic regression, SVM and the neural network models including MLP and CNN on predicting the value and trend of Dow Jones Industrial Average (DJIA) will be evaluated.

1.1.2 Financial Market and Dow Jones Industrial Average

One of the most active financial markets is the stock market. The performance of the stock market is related to the overall performance of a nation's economy while stock indices serve as barometers of the stock market. For example, launched in 1896, Dow Jones Industrial Average (DJIA) is a well-known price-weighted stock index which consists of the stocks of 30 blue chip companies. All of these 30 component companies are listed in the New York Stock Exchange and NASDAQ and they are representative of American industries (S&P Dow Jones Indices, 2017). Because it is of good diversity and liquidity of the stock market, it is not only an effective indicator of the market, but is also suitable for testing trading model.

There exist different theories and hypotheses regarding the financial market. According to random walk theory, the price changes are random and are thereby unpredictable, which is consistent with the Efficient Market Hypothesis. On the contrary, technical analysts believe that they can earn excess return by identifying the trends in the financial markets with some analytic tools such as analyzing charts of historical price movements.

1.2 Business Goal and Technical Objective

Our project intends to build models to predict the trend and the value of Dow Jones Industrial Average. The trading data of the past 30 business days will be analyzed to predict the exact value or the trend in the 31st day. For value prediction problem, we hope to gain a predicted price as close as possible to the real price while with regards to trend prediction, the target is to achieve an accuracy above the larger percentage of the positive or negative samples in the dataset, because for instance, if the positive samples take up 60% of the whole dataset, then a simple classifier is to predict all the samples to be positive to achieve 60% accuracy from the perspective of probability.

The technical difficulty of the project lies in the determination of optimal

hyperparameters in the model to produce a better prediction. Analysis regarding the performance and the limitation of the established models will be conducted.

1.3 Related Work

Deep learning is used widely in the field of financial prediction in recent decades. Krauss, Do and Huck (2015) applied a four-layer standard neural network and the ensemble models combining artificial neural network (ANN) and other machine learning models such as random forests to generate one-day-ahead trading signals with data of S&P 500. Simple returns over different periods of the constituents of S&P 500 are generated as the feature space to serve as the input of the model while whether the "one-period return of stock *s* is larger than the corresponding cross-sectional median return computed over all stocks" is the response variable. This problem is defined as a binary classification problem. The undervalued and overvalued stocks are picked out according to the largest and smallest predicted probabilities and then be bought and be sold respectively. The deep learning model achieves 0.33% return per day.

According to the preliminary research, CNN has significant achievements in image recognition. Lecun (1998) developed LeNet to recognize handwritten characters, which are the first successful applications of CNN to explore the spatial relationship between pixels by convolution. Krizhevsky, Sutskever, and Hinton (2012) developed AlexNet which applies CNN to classify the 1.2 million high-resolution images into 1000 classes. Instead of tanh and average pooling, AlexNet adopts ReLU nonlinearity and max pooling and is trained on multiple GPU, substantially improving the computation speed.

There exists argument between value predictions and sign predictions in stock prediction problem. The former prediction is usually evaluated by the deviation between the predicted and real values. Nevertheless, as suggested by Leung, Daouk and Chen (2000), the trading signals generated by value predictions with small deviations are not necessarily as profitable as a sign prediction model. The empirical results of comparing some multivariate classification models and level (i.e. value) estimation models show that classification models surpass the value prediction models by earning more returns.

Even though the predicted value is close to the real value, if the sign is predicted falsely, a loss will still be incurred. In contrast, if the trend is predicted accurately, we can ensure the gain without considering the magnitude. Furthermore, considering that pattern and image recognition are classification problems, the focus on our project is to explore the application of neural networks including MLP and CNN in stock trend prediction. In addition, linear regression and MLP will also be applied to predict the exact values to evaluate their performance of stock price prediction.

1.4 Outline of the Report

After the introduction session, the remainder of this report will first state how we are going to collect and preprocess the data, followed by the theories and mechanisms of different models applied in the project. Subsequently, the most critical procedure of model training will be presented. Then we will report the model training results and the mock trading results based on the constructed models. After that, discussions as well as the future improvement of the project will also be given. Finally, the report ends with a brief evaluation of our project.

2. Methodology

Fundamental analysis and technical analysis are two main methodologies to analyze the financial market. In our project, technical analysis is applied to study the past trading data which we believe will give implications on the future price movements.

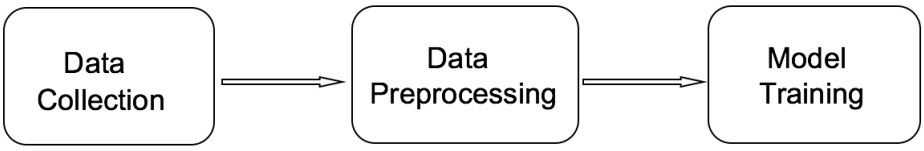

Figure 2.1 Three technical procedures of the project

Figure 2.1 shows the general technical flow of the project. The data will be collected and preprocessed before they are fitted to the model.

The following session will first define our stock prediction problem and then present

how the raw data are collected and preprocessed for the model construction. Before the discussion of the most crucial part of model training, the theories and mechanisms of the models will be introduced and the assessment of the models will be stated to facilitate the elaboration of the model training process. Finally, how the programming language python is employed to implement each procedure will be explained.

2.1 Binary Classification Problem

In essence, our stock trend prediction problem is a supervised binary classification problem because the data will be labelled and the model will generate a discrete binary result (Y = 0 or 1) to represent the trend.

Logistic Regression, SVM, MLP, 1D CNN (univariate) and 1D CNN (multivariate) will be adopted to directly predict the trend and apply accuracy as the evaluation metrics.

2.2 Regression Problem

The stock price prediction problem is a supervised regression problem, for which we intend to find out the expectation of the price of the 31^{st} day (response) given the price values (predictors) of the past 30 days. As a matter of fact, based on the difference between the predicted adjusted closing price of the 31^{st} day and the real adjusted closing price of the 30^{th} day, we can also get a predicted trend.

Linear Regression and MLP will be applied to directly predict the adjusted closing price.

2.3 Data Collection

6 stock variables of DJIA of each trading day with the span of 30 years, from Oct. 5, 1987 to Sep. 29, 2017 are collected through Yahoo Finance API in python.

The collected 6 variables with the span of 30 years are as follows:

Table 2.1 Breakdown of the six variables in the input dataset

Variable name	Variable Definition (from Investopedia)	Data Source

Open	The price at which a security first trades upon the opening of an exchange on a given trading day.	Yahoo Finance
High	A security's intraday high trading price.	
Low	A security's intraday low trading price.	
Close	The final price at which a security is traded on a given trading day.	
Adjusted Close	A stock’s closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time prior to the next day's open.	
Volume	The number of shares or contracts traded in a security or an entire market during a given time period.	

In the period from Oct. 5, 1987 to Sep. 29, 2017, there are in total 7560 valid trading days collected. As adjusted closing price adjusts the closing price for stock splits, dividends and rights offerings, adjusted closing price is selected instead of the closing price to indicate the close price of a stock in the market in our project. The trend and value of adjusted closing price become the response variables.

2.4 Data Preprocessing

For the sake of brevity, in the following part of the report, Univariate models will refer to Linear regression, Logistic regression, SVM, MLP and 1D CNN (univariate) while Multivariate model will refer to 1D CNN (multivariate) because the input data to these two groups are different. The input data of Univariate models only contain single variable while the Multivariate model contains 6 variables in the input.

Univariate models:

The predictors (X) consist of the adjusted closing price of 30 consecutive days. The response (Y) can be the adjusted closing price of DJIA of the 31st day for the regression models or the trend of DJIA in the 31st day for classification models.

Multivariate model:

The input (X) are composed of 6 variables of DJIA of 30 days. The 6 variables are Open, High, Low, Adjusted Close, Volume and Percentage Change. The definition of Percentage Change at time T is:

$$Percentage\ Change_T = \frac{Price_T - Price_{T-1}}{Price_{T-1}}$$

Apart from price increase or decrease, percentage change also provides information about the relative magnitude of the price change. The response (Y) can be the adjusted closing price of DJIA of the 31st day for the regression models or the trend of DJIA in the 31st day for classification models

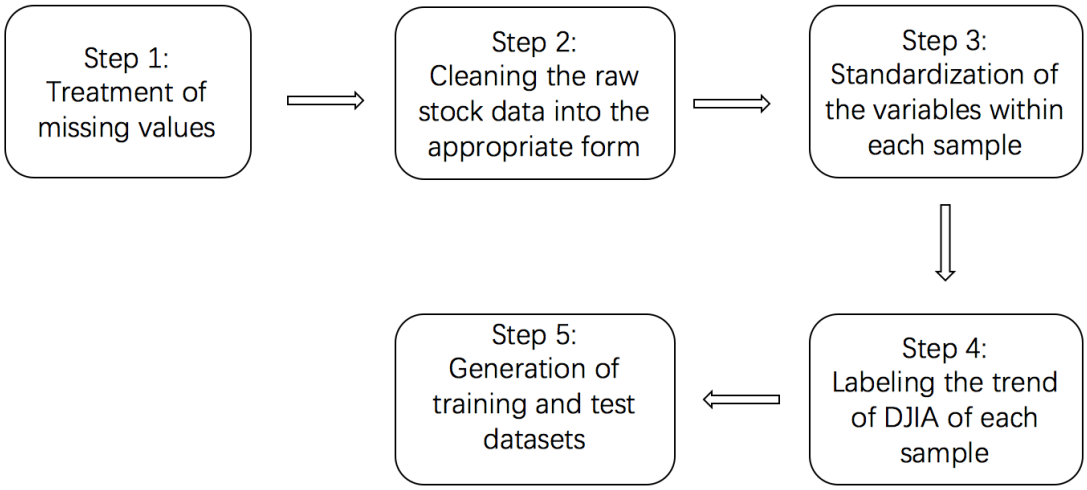

Figure 2.2 Five technical procedures of data preprocessing

Step 1 is a common data preprocessing technique. No missing values have been found in the dataset.

In Step 2, two different formats of input data are generated. As illustrated above, the format of input data to Univariate modes are as follows:

Table 2.2 Illustration of the first three samples of Univariate models

	Adjusted Close (X)					**Price (Y)**	**Trend (Y)**
	Day1	Day2	…	Day29	Day30	Day31	Day 31

Sample 1	2640.18	2548.63	…	1960.21	1935.01	1949.10	1
Sample 2	2548.63	2551.08	…	1935.01	1949.10	1922.25	0
Sample 3	2551.08	2516.64	…	1949.10	1922.25	1939.16	1

Each sample is composed of the adjusted closing price of any consecutive 30 days and the corresponding price or trend of the 31st day.

The input data of a CNN model can be 2-dimension or 3-dimension. For example, mnist dataset of handwritten digits contains greyscale images of dimension (28,28,1) while the dimension of a colored image can be (32,32,3) where the third dimension indicates the RGB channels. The structure of image data sheds light on the possible input dimension of the collected financial data in our project. Analogously, the raw stock data can be cleaned into the format of dimension (30, 6, 1) which corresponds to (number of days, number of variables, number of channel). The following matrix demonstrates the format of a sample to the Multivariate model:

Table 2.3 Illustration of the first sample of Multivariate model

	Date	**Open**	**High**	**Low**	**Adjusted Close**	**Volume**	**Percentage change**
Day 1	1987/10/5	2646.54	2658.79	2610.97	2640.18	19000000	-0.000307
Day 2	1987/10/6	2623.53	2632.83	2542.59	2548.63	24370000	-0.034676
…	…	…	…	…	…	…	…
Day 29	1987/11/12	1964.85	1983.92	1934.84	1960.21	28590000	0.032124
Day 30	1987/11/13	1957.07	1973.81	1923.08	1935.01	17070000	-0.012856

Each sample consists of a 2-dimensional input matrix of dimension (30, 6) and the corresponding trend of the 31st day.

Step 3 transfers the variables into a uniform scale, which eliminates the effect caused by different scales of different variables. Otherwise, the variable of larger scales such as Volume will be dominant, impeding the training process. As each sample contains a group of time series data, we focus on the relative price changes in this 30-day period and standardize each sample by its mean and standard deviation. After standardization, the samples exemplified above are as follows:

Table 2.4 Illustration of the first three samples of Univariate models after standardization

	Adjusted Close (X)					**Price (Y)**	**Trend (Y)**
	Day1	Day2	…	Day29	Day30	Day31	Day 31
Sample 1	1.9511	1.6184	…	-0.5203	-0.6119	-0.5607	1
Sample 2	1.8180	1.8275	…	-0.5641	-0.5094	-0.5607	0
Sample 3	2.0165	1.8752	…	-0.4526	-0.5627	-0.4934	1

Table 2.5 Illustration of the first sample of Multivariate model after standardization

	Date	**Open**	**High**	**Low**	**Adjusted Close**	**Volume**	**Percentage change**
Day 1	1987/10/5	1.8545	1.8458	1.8942	1.9511	-0.8287	0.1597
Day 2	1987/10/6	1.7729	1.7478	1.6604	1.6184	-0.5372	-0.4885
…	…	…	…	…	…	…	…
Day 29	1987/11/12	-0.5630	-0.7035	-0.4175	-0.5203	-0.3081	0.7713
Day 30	1987/11/13	-0.5906	-0.7417	-0.4577	-0.6119	-0.9335	-0.077

In the Step 4, the trend of DJIA of the 31st in each sample is labelled by comparing the value of the adjusted closing price of the 30th day and 31st day. If DJIA of the 31st day

decreases, then it is labelled as 0, otherwise labelled as 1.

In the final step, 7560 valid trading days which exclude the holidays and market suspensions are included in the dataset. If any consecutive 30 days along is regarded as an appropriate sample, the maximum number of valid samples will reach 7531 (7560-30+1=7531). With regards to time series data, to comply with the temporal order of the prices observed, we split the dataset in chronological order with the proportion approximately 9:1, a common ratio adopted in machine learning modelling.

Table 2.6 Sample size of training and test sets

Dataset	**Sample Size**	**Percentage**
Training	6700	0.8897
Test	831	0.1103
Total	**7531**	**1**

In the process of neural network model fitting, validation set can be split from the training set by specifying the validation_split = 0.1.

After the generation of training and test dataset, the percentages of positive and negative samples are as follows:

Table 2.7 Distribution of positive and negative samples

	Positive (1) %	**Negative (0) %**
Training set	0.5300	0.4700
Test set	0.5271	0.4729

This dataset is a balanced dataset. Since the percentage of positive samples in the test set is larger than 0.5, the quantitative objective of our project is to achieve the accuracy above 52.71% in the test set for classification models.

2.5 Machine Learning Models

A general form of the question to machine learning is to determine the mapping

function between the input (X) and the output (Y) F: X -> Y. With different machine learning algorithms which have different loss functions, evaluation metrics and methods to determinate parameters, machine learning models aim to identify this mapping by training the input data whereby we can make predictions on the new data.

Note that independent variables and predictors are interchangeable while dependent variable and response are interchangeable in the following explanation.

2.5.1 Traditional Machine Learning Models

This part will give a brief introduction to the commonly-used traditional machine learning models.

2.5.1.1 Linear Regression

A general form of multiple linear regression (MLR) of p-1 predictors is as follows:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_{p-1} x_{i,p-1} + \epsilon_i.$$

with the assumption that ϵ_i follows a normal distribution with mean 0 and constant variance σ^2. (x_i, y_i) refers to the i^{th} observation or data point while β_i measures the change in E(y) of per unit change in the associated predictor, ceteris paribus.

The parameters are estimated by Least Squares Estimation (LSE) which is to minimize the sum of squared errors (SSE).

$$\mathrm{SSE}(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

The coefficient of determination (R^2) and Mean Squared Error (MSE) can measure the goodness-of-fit of the linear regression model.

2.5.1.2 Logistic Regression

Logistic regression adopts logit link function to relate the predictors to a binary discrete response.

$$\log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right) \equiv \eta(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_{p-1} x_{p-1}$$

Logistic regression links the probability or odds of a case in the response to a linear combination of the predictors. The parameters are determined by Maximum Likelihood Estimation (MLE).

By comparing the predicted results to true labels, we can construct a confusion matrix based on which various evaluation indicators such as accuracy, sensitivity, precision etc. are derived.

ROC (Receiver Operating Characteristic) curve is also a prevalent graphical representation of the performance of the binary classifier by changing the discrimination threshold.

2.5.1.3 Support Vector Machine

The main idea of SVM is to find out an optimal hyperplane which can maximize the margin for separation of classes to this hyperplane. SVM algorithms adopt various kernels for the separating hyperplane. The options of kernels include linear, polynomial and RBF etc. When SVM is utilized for classification, the evaluation metrics are similar to Logistic Regression.

2.5.2 Deep Learning

In a neural network, the output of the current layer is passed as the input to the successive layer. The model functions as a hierarchical filter where the learned features are transformed and abstracted through the multiple layers at different degrees.

The input values are passed to the output layer through forward propagation while the parameters are updated by backpropagation in an effort to minimize the training error.

The mechanisms of MLP and CNN will be presented in the following part.

2.5.2.1 Multilayer Perceptron

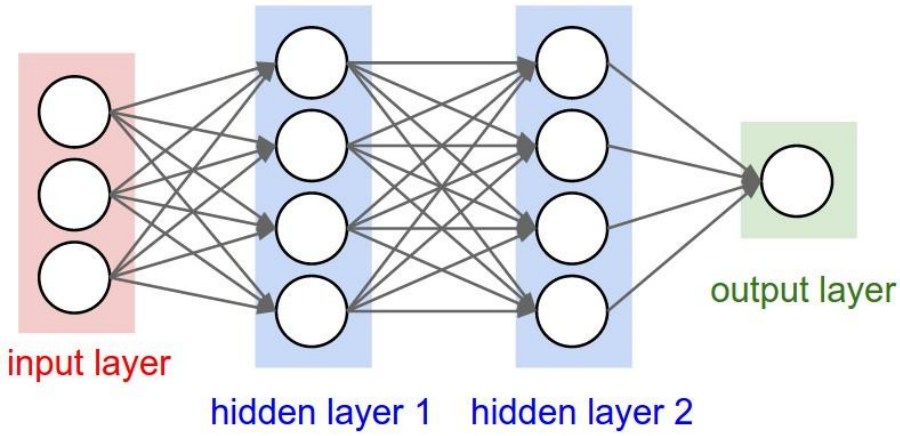

Figure 2.3 A regular three-layer neural network

Figure 2.3 shows the architecture of a regular three-layer (Hidden layer 1, Hidden layer 2 and Output layer) neural network where the first layer identifies the primitive features of the data input and feed these primitive features to the next layer to form more sophisticated features until it produces an output in the final layer. How the information of the previous layer is passed to the current layer is determined by the activation function of the current layer.

2.5.2.2 Convolutional Neural Network

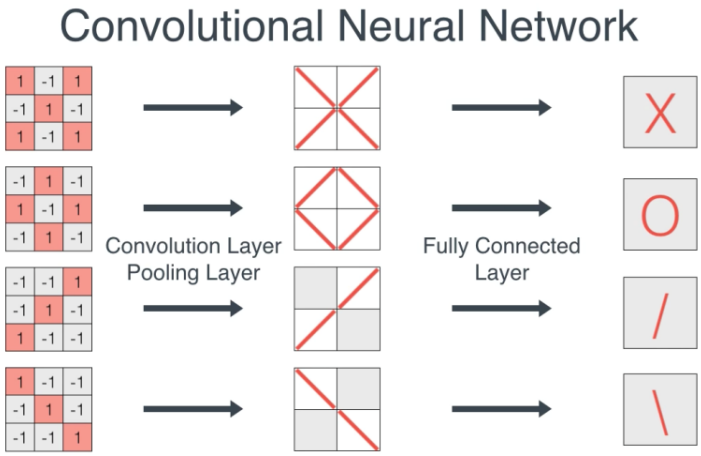

Figure 2.4 How the data from image pixels are recognized as patterns by the convolutional neural network.

Figure 2.4 displays a convolutional neural model that contains the three main types of

layers: Convolution Layer, Pooling Layer, and Fully Connected Layer. The convolutional and pooling layer first filter and abstract the features and then pass the features to the fully connected layer where each neuron has an effect on all the neurons of the next layer to produce the output. The structure of fully connected layer is similar to the hidden layer in the MLP introduced above. These three types of layers in conjunction with other possible kinds of layers are stacked together to form a CNN architecture.

2.5.2.2.1 Convolutional Layer

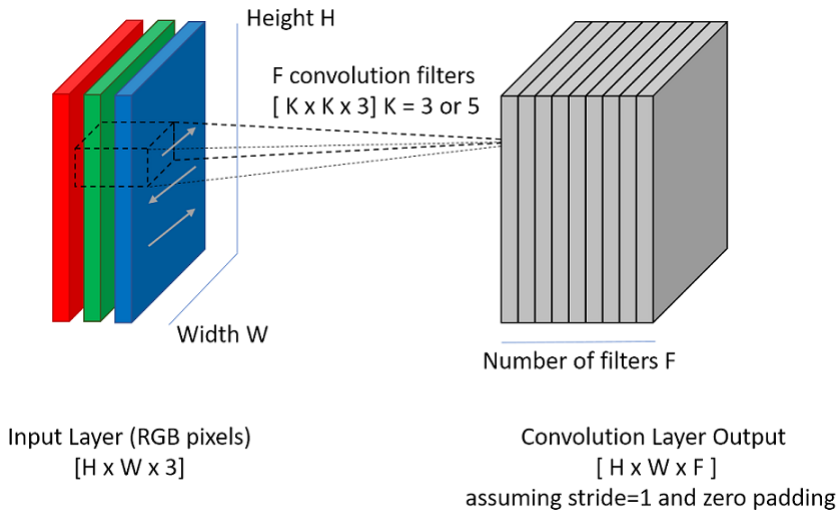

Figure 2.5 How an image with three channels (RGB) is processed by the filters in the convolutional layer

Figure 2.5 shows how a colored image of the dimension (H, W, 3) is processed by F filters of size K to form a feature map. The filter functions as a sliding window that convolves with the corresponding area of the input along its height and width at each depth. The dimension of the formed feature map is determined by the choices of number of filters, filter size, stride and zero padding.

2.5.2.2.2 Pooling Layer

A convolutional layer is commonly followed by a pooling layer to further downsample the feature map spatially. Maxpooling is the most prevalent pooling function nowadays.

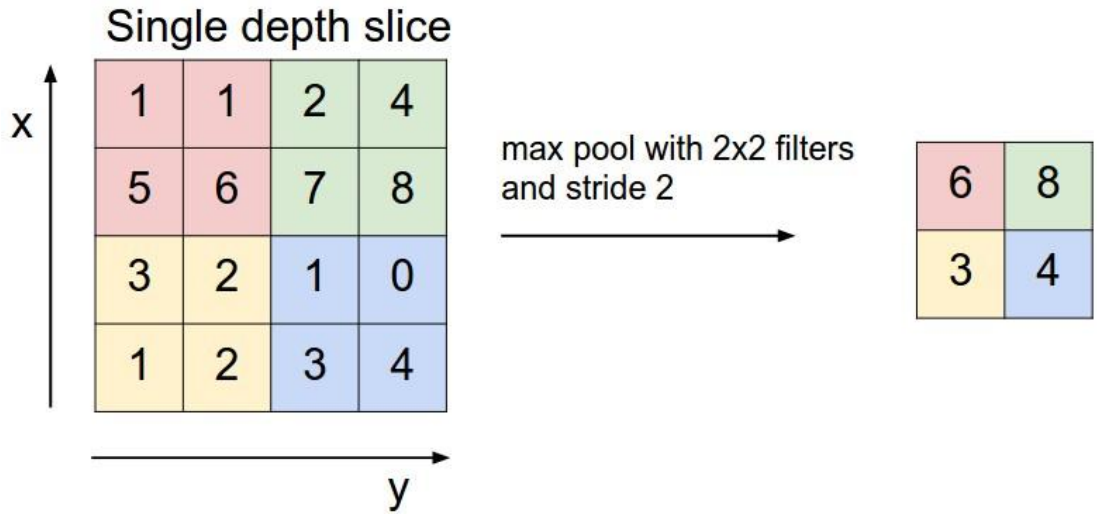

Figure 2.6 Maxpooling with a filter of size 2x2 and a stride of 2

Figure 2.6 shows how a feature map is downsampled by a filter of size 2x2 and a stride of 2 through Maxpooling. The maximum feature in the corresponding area of the feature map is extracted to formulate a new feature map. In addition, pooling layers do not contain the parameters to be trained.

Compared to the standard neural network, CNN decreases the number of parameters required in the network because of parameter sharing. Parameter sharing makes a reasonable assumption that if one feature is useful at one particular position, then the feature should also be useful at another position (cs231n, n.d.). Therefore, the identical or similar features can be detected by the same parameters.

Apart from the demand for fewer parameters, CNN excels at extracting spatial and temporal relationship in the data. As the financial data are time series data, 1-dimensional CNN is applied to explore the temporal relationship along the time dimension of the data. In 1D CNN, the filter is a vector instead of a 2-dimensional matrix. The 1D filter has also been applied in Natural Language Procession (e.g. sentence classification) as a sentence or a speech also has a temporal structure.

2.6 Model training

Model training is the most critical part that aims to resolve the most substantial

technical difficulty to our project. In the following discussion, hyperparameters and parameters that determine the structure of the models will be explained. Subsequently, a flow of model training process will be illustrated on how the hyperparameters will be tuned and what optimization procedures can be adopted based on the performance of the training and validation sets.

2.6.1 Hyperparameters and Parameters

2.6.1.1 Traditional Learning Models

In the Linear Regression and Logistic Regression, regularization such as lasso (L1 norm) and ridge regression (L2 norm) are applied to reduce overfitting and collinearity. When implementing regularization, the coefficient λ is a hyperparemeter that controls the degree of penalty on the complexity of the model.

The kernels of SVM are controlled by different hyperparameters. A commonly used kernel for SVM is the Radial Basis Function (RBF) kernel which is determined by two hyperparameters: C and gamma. C represents cost function which controls the smoothness of the decision surface while gamma decides the influence of a training sample.

Under different configurations of hyperparameters, different sets of parameters are trained.

2.6.1.2 Neural Network Models

The following hyperparameters and parameters are to be determined to build a neural network model. The hyperparameters of a MLP are identical to the fully-connected layer in the CNN.

Hyperparameters

- Number of layers
- Types of layers
- Number of filters in the convolutional layer
- Size of the filter in the convolutional layer

- Size of the filter in the pooling layer
- Number of hidden units of the fully-connected layer
- Activation function of the fully-connected layer

Parameters

- Weights and bias of each layer

2.6.2 Backpropagation of Neural Networks

There can be different combinations of hyperparameter settings. With a fixed setting of hyperparameters, the parameters weights and bias terms can be determined by means of Backpropagation. Backpropagation transfers the problem to a mathematical optimization problem to minimize the loss function. Loss function is a measurement of difference between the predicted results and the true values. For example, in terms of binary classification problem, the cross-entropy cost function is selected as the loss function to be minimized. Cross-entropy cost function is defined by

$$C = -\frac{1}{n}\sum_{i=0}^{n}[y_i log\hat{y_i} + (1 - y_i)\, log(1 - \hat{y_i})]$$

where $\hat{y_1}$ is the predicted output for the i^{th} sample

At each iteration, the weights and bias will be updated by calculating the gradient of loss function. The values of these two parameters will finally converge rather than diverge to the optimal solution after a number of iterations because the cross-entropy cost function is a convex function.

2.6.3 Tuning the Hyperparameters

Here we use the classification problem as an example to demonstrate how the hyperparameters will be tuned.

Figure 2.7 Demonstration regarding fitting model on the datasets

In Figure 2.7, with a specified combination of hyperparameters, the parameters including weights and bias terms of each layer are trained from the training set. Then

when fitting the model to make predictions on both the training set and validation set, we can get the accuracy of training dataset (A1) and the accuracy of validation dataset (A2).

Table 2.8 Four scenarios of the accuracy of training set and the accuracy of validation set

A1 (Training)	A2 (Validation)	Implication	Mitigation
Low	Low	Underfitting	Increase the complexity of the model
Low	High		
High	Low	Overfitting	Regularization
High	High	Satisfactory	None

Table 3 shows four possible scenarios of A1 and A2 by comparing A1 and A2 to 0.5271. Regardless of A2, if A1 is below 0.5271, it implies the problem of underfitting (high bias). To solve this problem, we can increase the complexity of the model such as increasing the number of layers, the number of filters or the number of hidden units. Another scenario is that A1 is high but the model cannot generalize the good performance to the new datasets, which leads to the problem of overfitting (high variance). Regularization techniques including L2 regularization and dropout can be applied to mitigate this problem.

To summarize, with a setting of hyperparameters, the weights and biases are updated by backpropagation in the neural network models. Evaluation concerning the performance of the setting of hyperparameters is based on the accuracy and loss of the model on the training and validation sets, which gives implications on how to tune the hyperparameters and what optimization procedures should be adopted. After a number of fitting trials, the group of hyperparameters that gives the best test set performance will be selected.

2.7 Assessment of the model

There are various evaluation metrics for different models. To ensure uniformity, we adopt accuracy as the assessment of the classification models while mean squared error (MSE) is utilized as the assessment of the regression models.

Classification models:

By fitting the model to the dataset, the trained model will produce a predicted label for each sample. The accuracy of a dataset can be acquired by comparing the predicted labels to the true labels in the specified dataset. The definition of the accuracy of a dataset is as follows:

$$Accuracy = \frac{the\ number\ of\ correctly\ classified\ samples\ in\ the\ dataset}{total\ number\ of\ samples\ in\ the\ dataset}$$

Regression models:

Similarly, by fitting the model to the dataset, the trained model will produce a predicted $\hat{Y}$ for each sample. The goodness-of-fit of the regression model is evaluated by mean squared error (MSE) which measures the mean value of the squared deviations of the predictions from the true values.

$Mean\ squared\ Error(MSE) = \frac{SSE}{n} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$ where n is the sample size

The objective of model fitting is to increase the accuracy or to reduce MSE.

2.8 Software Setup

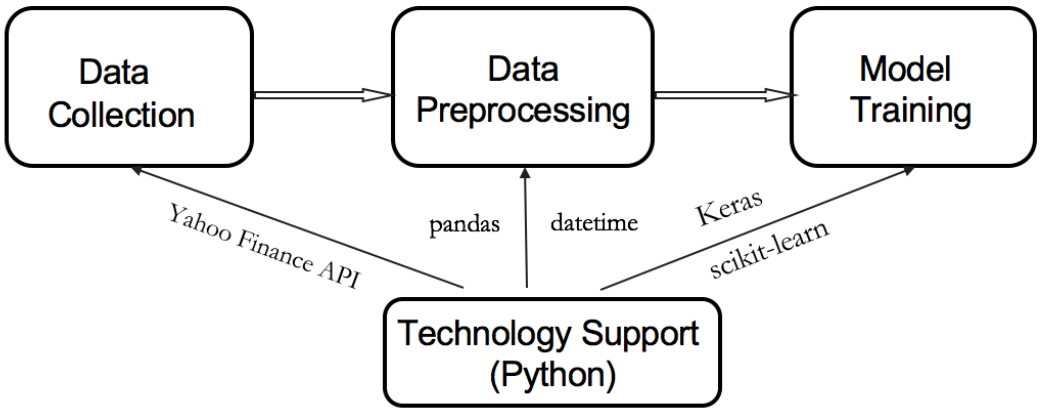

Figure 2.8 How the technical procedures are implemented in Python

Figure 2.8 shows how python contributes to implement the technical procedures. The

model will be established by the open source programming language Python. In comparison to other programming languages such as c++ and java, python is a higher-level language with built-in functions that can implement complicated operations. In Data Collection, Yahoo Finance API in python is employed to extract the stock data from Yahoo Finance. In addition, we make use of the package pandas and the module datetime in python to clean the raw data into a well-structured data frame with Date as the index in Data Preprocessing.

The Python package scikit-learn or sklearn provides abundant machine learning algorithms. We use sklearn to implement traditional machine learning models. In addition, there exist a number of frameworks for deep learning. Keras is a high-level neural networks API, written in Python and is capable of running using Tensorflow as the backend. It not only provides a variety of neural network layers with different arguments but also contains some popular gradient descent optimization algorithms such as Stochastic gradient descent (SGD), Adaptive moment estimation (Adam) etc. Furthermore, we can use Keras to apply dropout and regularization.

3. Results and Findings

The evaluation metrics of training and test sets will be displayed for each model. The process to determine optimal hyperparameter setting for traditional machine learning models will be shown while for neural network models, three models with the best performance will be reported.

3.1 Regression models

The predicted values of the first 100 data points in the test set will be plotted against the real price for the best linear regression model and MLP model.

3.1.1 Linear Regression

We use linear_model in the sklearn library. In this function λ is specified as alpha. The regularization strength is larger with a larger alpha.

Without any regularization, the mse of training and test sets are as follows:

Table 3.1 MSE of training and test sets without regularization by linear regression

Training MSE	Test MSE
9655.06	18352.1540

3.1.1.1 Lasso regression (L1 norm)

With lasso regression, the norm of some parameters may become zero, realizing variable selection.

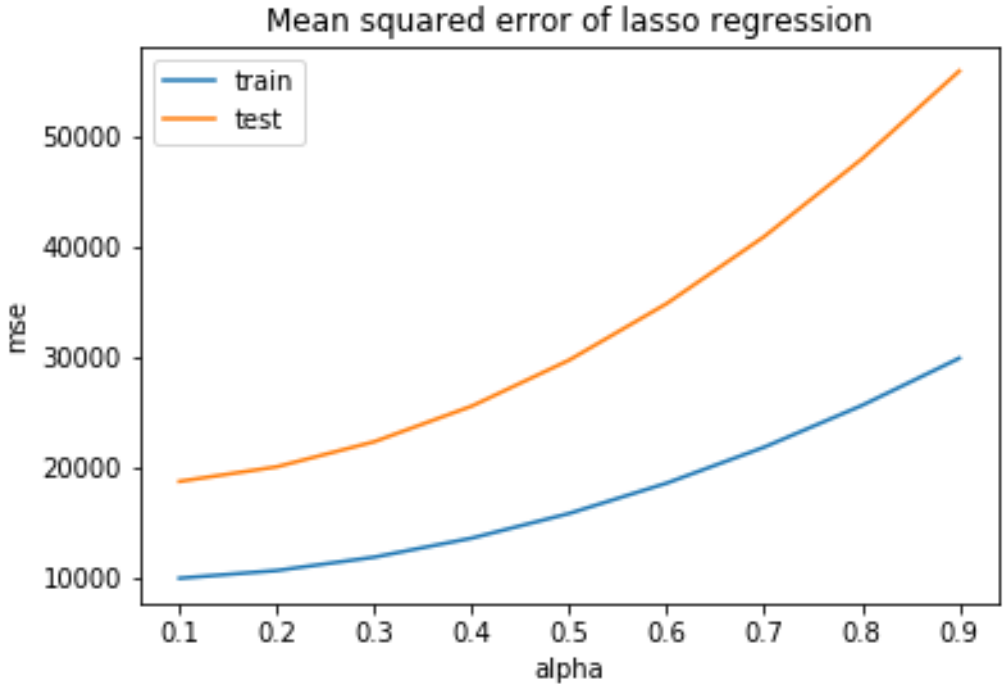

Figure 3.1 MSE of training and test sets by lasso regression with varying alpha from 0.1 to 0.9

Table 3.2 MSE of training and test sets by varying alpha from 0.01 to 0.09 by lasso regression

alpha	**Training MSE**	**Test MSE**
0.01	9689.1791	18325.1919
0.02	**9694.8950**	**18313.7340**
0.03	9701.0727	18327.7996
0.04	9712.3855	18351.3885

0.05	9728.8333	18384.5006
0.06	9750.4161	18427.1361
0.07	9777.1340	18479.2949
0.08	9808.9869	18540.9769
0.09	9845.9747	18612.18223

When alpha = 0.02, the MSE of test set by lasso regression reaches the minimum of 18313.73.

3.1.1.2 Ridge Regression (L2 norm)

Compared to lasso regression, L2 norm regularization gives far less penalty to the complexity of the model. The norm of the parameters will be close to 0 but will not be eliminated.

Figure 3.2 MSE of training and test sets by ridge regression with varying alpha from 1 to 99

Figure 3.2 shows that although the mse of test sets decreases as alpha increases. The mse only begins to be smaller than 18315 when alpha is as large as 80. Thus, lasso regression is more effective.

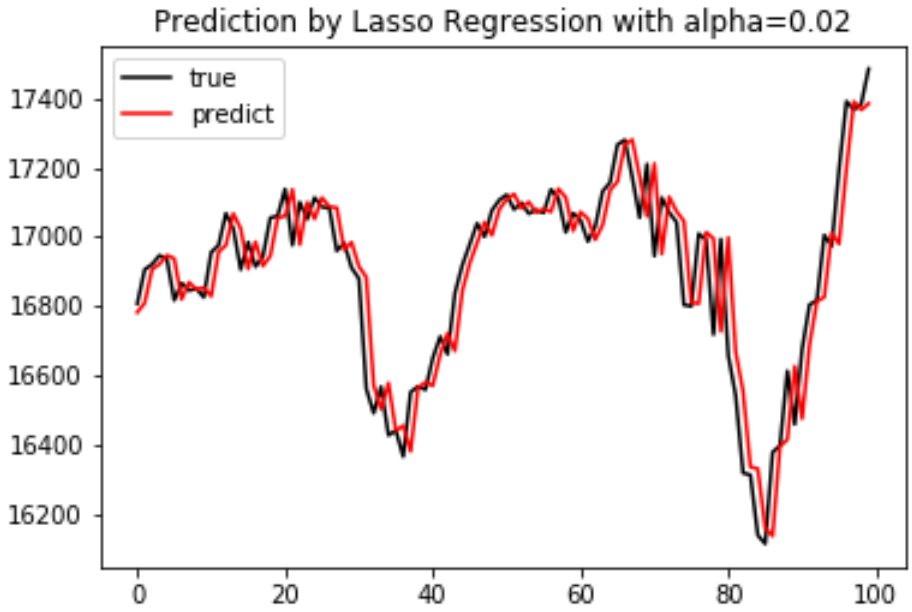

Figure 3.3 The first 100 predicted prices against real prices by lasso regression with alpha = 0.02

There exists postponement in prediction values compared to real values.

3.1.2 Multilayer Perceptron (Regression)

Three reported MLP models all achieve a lower MSE than the best linear regression model which has MSE equal to 18313.7340.

3.1.2.1 MLP-R-Model 1

Table 3.3 Detailed structure and hyperparameters of MLP-R-Model 1

Layer	**Activation**	**Regularization**
Dense (400)	LeakyReLU	L2(0.004)
Dropout		0.50
Dense (200)	LeakyReLU	L2(0.004)
Dropout		0.50
Dense (100)	LeakyReLU	L2(0.004)
Dropout		0.50

Output	Linear

Table 3.4 MSE of training and test sets by MLP-R-Model 1

	Training	Test
MSE	9570.3835	18226.6978

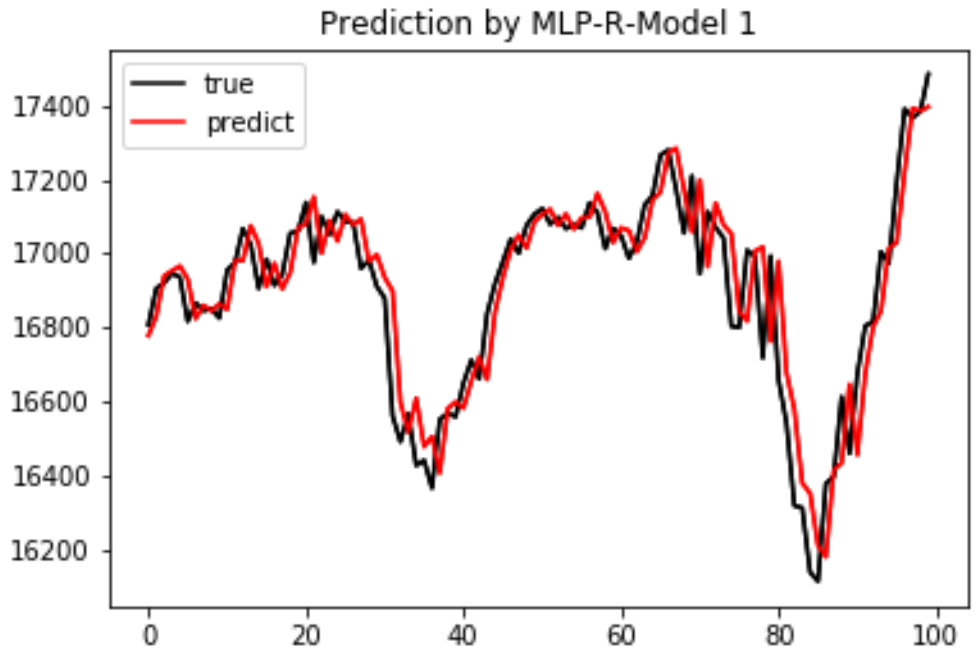

Figure 3.4 The first 100 predicted prices against real prices by MLP-R-Model 1

Similarly, there exists postponement in prediction values compared to real values.

3.1.2.2 MLP-R-Model 2

Table 3.5 Detailed structure and hyperparameters of MLP-R-Model 2

Layer	**Activation**	**Regularization**
Dense (400)	LeakyReLU	L2(0.003)
Dropout		0.45
Dense (200)	LeakyReLU	L2(0.003)
Dropout		0.45
Dense (100)	LeakyReLU	L2(0.003)

Dropout		0.45
Output	Linear	

Table 3.6 MSE of training and test sets by MLP-R-Model 2

	Training	Test
MSE	9645.0725	18233.0277

3.1.2.3 MLP-R-Model 3

Table 3.7 Detailed structure and hyperparameters of MLP-R-Model 3

Layer	**Activation**	**Regularization**
Dense (400)	LeakyReLU	L2(0.003)
Dropout		0.50
Dense (200)	LeakyReLU	L2(0.003)
Dropout		0.50
Dense (100)	LeakyReLU	L2(0.003)
Dropout		0.50
Output	Linear	

Table 3.8 MSE of training and test sets by MLP-R-Model 3

	Training	Test
MSE	9501.9817	18267.5403

3.2 Classification Models

3.2.1 Logistic Regression

sklearn.linear_model.LogisticRegression is applied to conduct logistic regression. Different from the counterpart alpha in the Linear Regression, the hyperparameter C which controls the regularization strength refers to the inverse of regularization strength, i.e. $alpha \propto \frac{1}{C}$. Smaller C specifies stronger regularization power. The default value of C is 1.0.

Table 3.9 Accuracy of training and test sets by logistic regression with L1 and L2

penalty with default C=1.0

Penalty	Training Accuracy	Test Accuracy
$l1$(C=1.0)	0.5421	0.5295
$l2$(C=1.0)	0.5397	0.5319

3.2.1.1 Logistic Regression with L2 Penalty

Figure 3.5 Accuracy of training and test sets by logistic regression with L2 penalty with C varying from 0.1 to 2.0

Figure 3.5 shows that the accuracy of test set may keep going up when C is decreasing below 0.1.

Table 3.10 Accuracy of training and test sets by logistic regression with L2 penalty with C varying from 0.01 to 0.1

C	Training accuracy	Test accuracy
0.01	0.5424	0.5187
0.02	0.5424	0.5259
0.03	0.5412	0.5307

0.04	0.5404	0.5319
0.05	0.5407	0.5331
0.06	0.5406	0.5343
0.07	**0.5406**	**0.5355**
0.08	**0.5407**	**0.5355**
0.09	0.5406	0.5343
0.1	0.5406	0.5331

When C = 0.07 or 0.08, the accuracy of test set both reaches its maximum of 0.5355 while the training accuracy is higher when C=0.08.

3.2.1.2 Logistic Regression with L1 Penalty

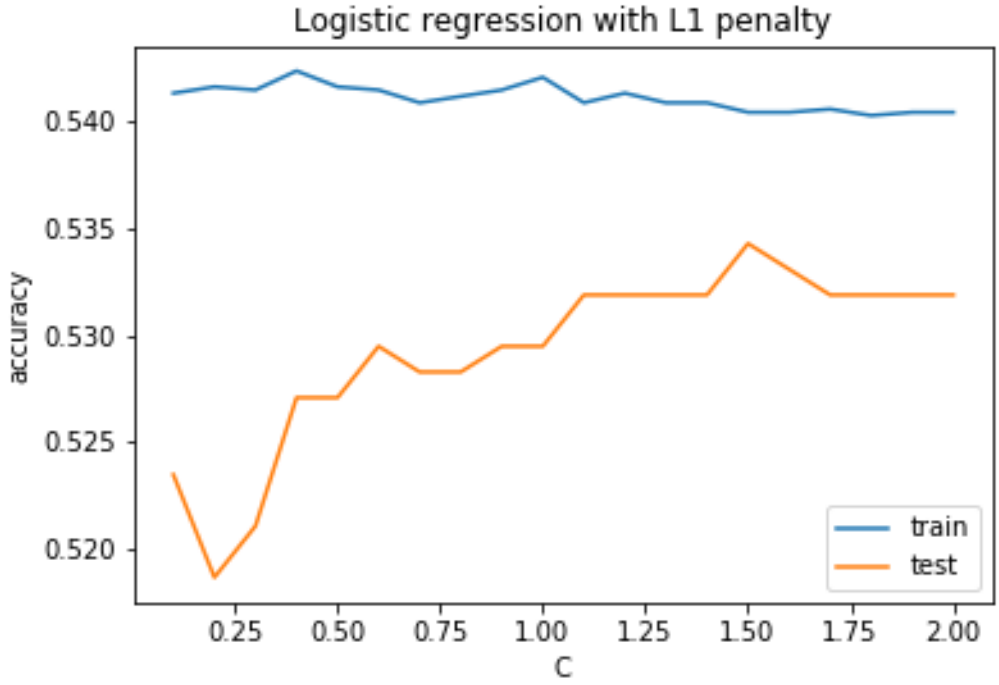

Figure 3.6 Accuracy of logistic regression with L1 penalty with C varying from 0.1 to 2.0

Figure 3.6 shows that the maximum accuracy occurs approximately in the interval (1.4, 1.6) of C.

Table 3.11 Accuracy of training and test sets by logistic regression with L1 penalty with C varying from 1.3 to 1.6

C	Training accuracy	Test accuracy
1.3	0.5409	0.5319
1.4	0.5409	0.5319
1.5	**0.5404**	**0.5343**
1.6	0.5404	0.5331

When C = 1.5, the accuracy of test set reaches its maximum of 0.5343.

3.2.2 Support Vector Machine

The linear kernel and the most commonly used RBF kernel are selected for the SVM model.

Table 3.12 Accuracy of training and test sets with linear kernel and RBF kernel and default C = 1.0

Kernel	Training Accuracy	Test Accuracy
Linear(C=1.0)	0.5401	0.5318
RBF (C=1.0)	0.6164	0.5307

It is found that the training accuracy and test accuracy remain unchanged with the linear kernel against various values of C. Thus the best accuracy by linear kernel is 0.5318.

In terms of RBF kernel, by grid search in the range C_range = [0.2,0.4,0.6,0.8,1.0,1.2] and gamma_range = [0.1, 0.5, 1], we get the following result:

Table 3.13 Accuracy of training and test sets with RBF kernel by grid search in C_range and gamma_range

C	gamma	Training accuracy	Test accuracy

0.2	0.1	0.5530	0.5187
0.2	0.5	0.5337	0.5283
0.2	1	0.5300	0.5271
0.4	0.1	0.6248	0.5114
0.4	0.5	0.5678	0.5187
0.4	1	0.5397	0.5283
0.6	**0.1**	**0.7069**	**0.5319**
0.6	0.5	0.9373	0.5174
0.6	1	0.9885	0.5283
0.8	0.1	0.7622	0.5259
0.8	0.5	0.9779	0.5066
0.8	1	0.9978	0.5223
1	0.1	0.7872	0.5223
1	0.5	0.9861	0.5054
1	1	0.9987	0.5126
1.2	0.1	0.8081	0.5235
1.2	0.5	0.9906	0.5102
1.2	1	0.9996	0.5054

When C = 0.6 and gamma = 0.1, the SVM model with RBF kernel achieves the maximum test set accuracy of 0.5319 by grid search in the specified range of C and gamma.

3.2.3 Multilayer Perceptron (Classification)

The architecture of the model is a regular three-layer neural network which consists of two hidden layers and one output layer. Three models with the highest accuracy on test set. The activation function ReLU and LeakyReLU are both adopted here.

3.2.3.1 MLP-C-Model 1

Table 3.14 Detailed structure and hyperparameters of MLP-C-Model 1

Layer	**Activation**	**Regularization**
Dense (600)	LeakyReLU	L2(0.0003)

Dropout		0.20
Dense (30)	LeakyReLU	L2(0.0003)
Dropout		0.20
Output	Sigmoid	

Table 3.15 Accuracy and loss of training and test sets by MLP-C-Model 1

	Training	**Test**
Accuracy	0.5543	0.5487
Loss	0.7002	0.7123

3.2.3.2 MLP-C-Model 2

Table 3.16 Detailed structure and hyperparameters of MLP-C-Model 2

Layer	**Activation**	**Regularization**
Dense (400)	ReLU	L2 (0.001)
Dropout		0.20
Dense (20)	ReLU	L2 (0.001)
Dropout		0.20
Output	Sigmoid	

Table 3.17 Accuracy and loss of training and test sets by MLP-C-Model 2

	Training	**Test**
Accuracy	0.5657	0.5463
Loss	0.7203	0.7320

3.2.3.3 MLP-C-Model 3

Table 3.18 Detailed structure and hyperparameters of MLP-C-Model 3

Layer	**Activation**	**Regularization**
Dense (600)	ReLU	L2 (0.0005)
Dropout		0.25
Dense (30)	ReLU	L2 (0.001)
Dropout		0.20

Output	Sigmoid

Table 3.19 Accuracy and loss of training and test sets by MLP-C-Model 3

	Training	**Test**
Accuracy	0.5696	0.5451
Loss	0.7077	0.7230

3.2.4 1D Convolutional Neural Network (Univariate)

BatchNormalization layer is inserted before the layer of nonlinear activation function. According to Ioffe and Szegedy (2015), this technique makes the layer inputs to take on a unit Gaussian distribution before activation, which can accelerate the training convergence and more robust to poor initialization.

It is omitted in the following tables for the sake of brevity.

3.2.4.1 Conv-UC-Model 1

Table 3.20 Detailed structure and hyperparameters of Conv-UC-Model 1 (BatchNormalization omitted)

Layer	**No. of filters**	**Filter size**	**Stride**	**Padding**	**Activation**	**Regularization**
Conv1D	32	3	1	Valid	ReLU	L2 (0.001)
MaxPooling1D	1	2	2	Valid		
Conv1D	32	3	1	Same	ReLU	
MaxPooling1D	1	2	2	Valid		
Dropout						0.30
Flatten						
Dropout						0.30
Dense (96)					ReLU	
Output					Sigmoid	

Table 3.21 Accuracy and loss of training and test sets by Conv-UC-Model 1

	Training	Test
Accuracy	0.5721	0.5548
Loss	0.6797	0.6983

3.2.4.2 Conv-UC-Model 2

Table 3.22 Detailed structure and hyperparameters of Conv-UC-Model 2 (BatchNormalization omitted)

Layer	**No. of filters**	**Filter size**	**Stride**	**Padding**	**Activation**	**Regularization**
Conv1D	32	3	1	Valid	ReLU	L2 (0.005)
MaxPooling1D	1	2	2	Valid		
Conv1D	32	5	1	Same	ReLU	
MaxPooling1D	1	2	2	Valid		
Dropout						0.30
Flatten						
Dropout						0.30
Dense (96)					ReLU	
Output					Sigmoid	

Table 3.23 Accuracy and loss of training and test sets by Conv-UC-Model 2

	Training	Test
Accuracy	0.5661	0.5475
Loss	0.6846	0.6938

3.2.4.3 Conv-UC-Model 3

Table 3.24 Detailed structure and hyperparameters of Conv-UC-Model 3 (BatchNormalization omitted)

Layer	**No. of filters**	**Filter size**	**Stride**	**Padding**	**Activation**	**Regularization**
Conv1D	32	3	1	Valid	ReLU	L2 (0.005)
MaxPooling1D	1	2	2	Valid		
Conv1D	32	5	1	Same	ReLU	

MaxPooling1D	1	2	2	Valid		
Dropout						0.30
Flatten						
Dropout						0.30
Dense (96)					ReLU	
Output					Sigmoid	

Table 3.25 Accuracy and loss of training and test sets by Conv-UC-Model 3

	Training	**Test**
Accuracy	0.5864	0.5463
Loss	0.6851	0.7004

3.2.5 1D Convolutional Neural Network (Multivariate)

Conv-MC-Model 1 and Conv-MC-Model 2 have the same accuracy of 0.5548.

3.2.5.1 Conv-MC-Model 1

Table 3.26 Detailed structure and hyperparameters of Conv-MC-Model 1 (BatchNormalization, Flatten omitted)

Layer	**No. of filters**	**Filter size**	**Stride**	**Padding**	**Activation**	**Regularization**
Conv1D	8	3	1	Same	ReLU	L2 (0.005)
MaxPooling1D	1	2	2	Valid		
Dropout						0.20
Conv1D	16	3	1	Same	ReLU	L2 (0.005)
MaxPooling1D	1	2	2	Valid		
Dropout						0.20
Dense (96)					ReLU	
Output					Sigmoid	

Table 3.27 Accuracy and loss of training and test sets by Conv-MC-Model 1

	Training	**Test**
Accuracy	0.6125	0.5548

Loss	0.6835	0.7138

3.2.5.2 Conv-MC-Model 2

Table 3.28 Detailed structure and hyperparameters of Conv-MC-Model 2 (BatchNormalization, Flatten omitted)

Layer	**No. of filters**	**Filter size**	**Stride**	**Padding**	**Activation**	**Regularization**
Conv1D	8	5	1	Same	ReLU	L2 (0.004)
MaxPooling1D	1	2	2	Valid		
Dropout						0.25
Conv1D	16	5	1	Same	ReLU	L2 (0.01)
MaxPooling1D	1	2	2	Valid		
Dropout						0.30
Dense (96)					ReLU	
Output					Sigmoid	

Table 3.29 Accuracy and loss of training and test sets by Conv-MC-Model 2

	Training	**Test**
Accuracy	0.5864	0.5548
Loss	0.6961	0.7108

3.2.5.3 Conv-MC-Model 3

Table 3.30 Detailed structure and hyperparameters of Conv-MC-Model 3 (BatchNormalization, Flatten omitted)

Layer	**No. of filters**	**Filter size**	**Stride**	**Padding**	**Activation**	**Regularization**
Conv1D	8	3	1	Same	ReLU	L2 (0.0025)
MaxPooling1D	1	2	2	Valid		
Dropout						0.20
Conv1D	8	3	1	Same	ReLU	L2 (0.0025)
MaxPooling1D	1	2	2	Valid		

Dropout		0.20
Dense (64)	ReLU	
Dropout		0.20
Output	Sigmoid	

Table 3.31 Accuracy and loss of training and test sets by Conv-MC-Model 3

	Training	**Test**
Accuracy	0.5637	0.5511
Loss	0.7000	0.7075

3.3 Summary of Results

The models with the best performance by each machine learning algorithm are summarized in this part and will be selected for mock trading in the subsequent session.

3.3.1 Regression Models

3.3.1.1 Linear Regression

The best hyperparameter setting for linear regression is lasso regression with alpha equal to 0.02. With this setting, the MSE on the test set is 18313.7340.

3.3.1.2 Multilayer Perceptron (Regression)

The best MLP model for regression is MLP-R-Model 1 which have three hidden layers with 400, 200, and 100 nodes and followed by a dropout layer with dropout probability equal to 0.50 respectively. The activation function in the convolutional layer is LeakyReLU while each convolutional layer implements L2 regularization with 0.004 as the regularization parameter. The MSE of this model on the test set is 18226.6978.

3.3.2 Classification Models

3.3.2.1 Logistic Regression

The best performance is given by the model with L2 penalty and with C equal to 0.08.

The accuracy of this model on the test set is 0.5355.

3.3.2.2 Support Vector Machine

The best SVM model uses RBF kernel with C equal to 0.6 and gamma equal to 0.1. This SVM model test set accuracy of 0.5319.

3.3.2.3 Multilayer Perceptron (Classification)

MLP-C-Model 1 gives the highest accuracy of 0.5487 in classifying the trend. This model contains two hidden layers with 600 and 30 nodes, followed by a dropout layer with dropout rate equal to 0.20 respectively. The activation function in the convolutional layers is LeakyReLU. L2 regularization with 0.0003 as penalty parameter is applied to all convolutional layers.

3.3.2.4 1D Convolutional Neural Network (Univariate)

Conv-UC-Model 1 achieves classification accuracy of 0.5548 on the test set and 0.5721 on the training set. The details of this model are discussed in 3.2.4.1.

3.3.2.5 1D Convolutional Neural Network (Multivariate)

Although Conv-MC-Model 1 and Conv-MC-Model 2 have same test set accuracy, Conv-MC-Model 1 is a better model because its training set accuracy is higher than that of Conv-MC-Model 2.

Conv-MC-Model 1 achieves classification accuracy of 0.5548 on the test set and 0.6125 on the training set. The details of this model are discussed in 3.2.5.1.

4. Mock Trading

The models summarized above will be applied in mock trading. The mock trading period is from April 2 to April 13 (10 business days). Assume we have enough money to invest in the first trade and assume we can go short in the stock market without virtually owning a stock. The decision to buy or sell is made before the market opens. The gains and loss will be calculated daily for each model. If the predicted price of today is higher than the real price on yesterday or the predicted label is 1, then the decision is to buy, otherwise to sell at the price of yesterday.

4.1 Regression Models

4.1.1 Linear Regression

Table 3.32 Mock trading results for linear regression model

Date	Prediction	Decision	Yesterday Price	Real Today Price	Gain/Loss
2-Apr	24128.46	Buy	24103.11	23644.19	-458.92
3-Apr	23677.28	Buy	23644.19	24033.36	389.17
4-Apr	24059.38	Buy	24033.36	24264.30	230.94
5-Apr	24286.11	Buy	24264.30	24505.22	240.92
6-Apr	24522.81	Buy	24505.22	23932.76	-572.46
9-Apr	23959.70	Buy	23932.76	23979.10	46.34
10-Apr	24004.37	Buy	23979.10	24408.00	428.90
11-Apr	24424.09	Buy	24408.00	24189.45	-218.55
12-Apr	24207.79	Buy	24189.45	24483.05	293.60
13-Apr	24495.83	Buy	24483.05	24360.14	-122.91
					257.03

The predictions by linear regression model are all higher than the real price of yesterday. As a result, the decision is to buy DJIA every day, which incurs an earning of $257.03.

4.1.2 Multilayer Perceptron (Regression)

Table 3.33 Mock trading results for MLP-R-Model 1

Date	Prediction	Decision	Yesterday Price	Real Today Price	Gain/Loss
2-Apr	24097.30	Sell	24103.11	23644.19	458.92
3-Apr	23647.17	Buy	23644.19	24033.36	389.17
4-Apr	24039.60	Buy	24033.36	24264.30	230.94
5-Apr	24268.02	Buy	24264.30	24505.22	240.92

6-Apr	24500.81	Sell	24505.22	23932.76	572.46
9-Apr	23948.38	Buy	23932.76	23979.10	46.34
10-Apr	23964.42	Sell	23979.10	24408.00	-428.90
11-Apr	24291.10	Sell	24408.00	24189.45	218.55
12-Apr	24197.79	Buy	24189.45	24483.05	293.60
13-Apr	24549.50	Buy	24483.05	24360.14	-122.91
					1899.09

The mock trading by MLP-R-Model 1 leads to a profit of $1899.09.

4.2 Classification Models

4.2.1 Logistic Regression

Table 3.34 Mock trading results for logistic regression model

Date	Prediction	Decision	Yesterday Price	Real Today Price	Gain/Loss
2-Apr	0	Sell	24103.11	23644.19	458.92
3-Apr	1	Buy	23644.19	24033.36	389.17
4-Apr	1	Buy	24033.36	24264.30	230.94
5-Apr	1	Buy	24264.30	24505.22	240.92
6-Apr	1	Buy	24505.22	23932.76	-572.46
9-Apr	1	Buy	23932.76	23979.10	46.34
10-Apr	0	Sell	23979.10	24408.00	-428.90
11-Apr	0	Sell	24408.00	24189.45	218.55
12-Apr	1	Buy	24189.45	24483.05	293.60
13-Apr	1	Buy	24483.05	24360.14	-122.91
					754.17

The logistic model leads to a profit of $754.17.

4.2.2 Support Vector Machine

Table 3.35 Mock trading results for SVM model

Date	Prediction	Decision	Yesterday Price	Real Today Price	Gain/Loss

2-Apr	0	Sell	24103.11	23644.19	458.92
3-Apr	0	Sell	23644.19	24033.36	-389.17
4-Apr	0	Sell	24033.36	24264.30	-230.94
5-Apr	1	Buy	24264.30	24505.22	240.92
6-Apr	1	Buy	24505.22	23932.76	-572.46
9-Apr	1	Buy	23932.76	23979.10	46.34
10-Apr	1	Buy	23979.10	24408.00	428.90
11-Apr	0	Sell	24408.00	24189.45	218.55
12-Apr	1	Buy	24189.45	24483.05	293.60
13-Apr	1	Buy	24483.05	24360.14	-122.91
					371.75

The SVM model makes a profit of $371.75.

4.2.3 Multilayer Perceptron (Classification)

Table 3.36 Mock trading results for MLP-C-Model 1

Date	Prediction	Decision	Yesterday Price	Real Today Price	Gain/Loss
2-Apr	0	Sell	24103.11	23644.19	458.92
3-Apr	0	Sell	23644.19	24033.36	-389.17
4-Apr	1	Buy	24033.36	24264.30	230.94
5-Apr	1	Buy	24264.30	24505.22	240.92
6-Apr	1	Buy	24505.22	23932.76	-572.46
9-Apr	1	Buy	23932.76	23979.10	46.34
10-Apr	1	Buy	23979.10	24408.00	428.90
11-Apr	0	Sell	24408.00	24189.45	218.55
12-Apr	1	Buy	24189.45	24483.05	293.60
13-Apr	1	Buy	24483.05	24360.14	-122.91
Total					833.63

We earn a profit of $833.63.

4.2.4 1D Convolutional Neural Network (Univariate)

Table 3.37 Mock trading results for Conv-UC-Model 1

Date	Prediction	Decision	Yesterday Price	Real Today Price	Gain/Loss
2-Apr	0	Sell	24103.11	23644.19	458.92
3-Apr	0	Sell	23644.19	24033.36	-389.17
4-Apr	1	Buy	24033.36	24264.30	230.94
5-Apr	1	Buy	24264.30	24505.22	240.92
6-Apr	1	Buy	24505.22	23932.76	-572.46
9-Apr	1	Buy	23932.76	23979.10	46.34
10-Apr	1	Buy	23979.10	24408.00	428.90
11-Apr	1	Buy	24408.00	24189.45	-218.55
12-Apr	1	Buy	24189.45	24483.05	293.60
13-Apr	0	Sell	24483.05	24360.14	122.91
					642.35

The mock trading result is a profit of $642.35.

4.2.5 1D Convolutional Neural Network (Multivariate)

Table 3.38 Mock trading results for Conv-MC-Model 1

Date	Prediction	Decision	Yesterday Price	Real Today Price	Gain/Loss
2-Apr	1	Buy	24103.11	23644.19	-458.92
3-Apr	1	Buy	23644.19	24033.36	389.17
4-Apr	1	Buy	24033.36	24264.30	230.94
5-Apr	1	Buy	24264.30	24505.22	240.92
6-Apr	1	Buy	24505.22	23932.76	-572.46
9-Apr	1	Buy	23932.76	23979.10	46.34
10-Apr	1	Buy	23979.10	24408.00	428.90
11-Apr	1	Buy	24408.00	24189.45	-218.55
12-Apr	1	Buy	24189.45	24483.05	293.60
13-Apr	1	Buy	24483.05	24360.14	-122.91
					257.03

The 1D convolutional (multivariate) model give predictions of 1 to all the samples in

our mock data set, resulting in a gain of $257.03.

4.3 Summary of Mock Trading Results

Table 3.39 Summary of mock trading results in descending order of gain

Model	Gain / Loss ($)
Multilayer perceptron (regression)	1899.09
Multilayer perceptron (classification)	833.63
Logistic regression	754.17
1D Convolutional neural network (Univariate)	642.35
Support Vector Machine	371.75
Linear regression	257.03
1D Convolutional neural network (Multivariate)	257.03

All models make earnings in mock trading. MLP (regression) model generates the highest return of $1899.09. MLP (classification) model ranks the second with a profit of $833.63, followed by the logistic regression model with earning of $754.17. Linear regression and 1D CNN (Multivariate) make the smallest earnings.

5. Discussions and Analysis

5.1 Evaluations and Explanations

From the model training results which contain the test set accuracy and test set MSE, we can first conclude that all the classification models have accomplished the target of obtaining accuracy above 0.5271. In addition, MLP (regression) achieves a lower MSE than linear regression. The model training results show that neural network models generally have more capacity than the traditional machine learning models in our project. However, the mock trading results show that the 1D CNN (Multivariate) model does not work well as expected while MLP models display good performance in both regression and classification.

It is also noteworthy that both the linear regression model and 1D CNN (Multivariate) model predict ‘buy’ in every trading day. It may imply the problem that these two models are very likely to predict positive case when they encounter new data. However, more data are required to further study these problems as 10 business days is a short time and these problems may occur only regarding the data of these 10 days.

5.2 Limitations and Difficulties

Although the dimension and the format of the financial data comply with the dimension of image data, the numerical values of financial data do not have the characteristics of pixel data. Normally, a pixel is associated with the surrounding pixels, showing information of shapes, colors etc. The configuration of financial data into a matrix in our project may fail to give appropriate graphical representation for CNN to learn.

Besides, data augmentation is a common and useful technique to deal with overfitting in terms of pattern recognition problem. Data augmentation can be accomplished by cropping, flipping, rotation etc. Nevertheless, we cannot utilize this powerful technique because our financial data are not image data.

The most substantial challenge lies in how to determine the optimal configuration of the hyperparameters of the neural networks. The number of hyperparameters to be determined by neural network models is much larger than linear regression, logistic regression etc. which makes grid search extremely time-consuming. In our project, the hyperparameters are tuned by hand-tuning, which may be inefficient and may fail to find out better configurations. The difficulty in finding the optimal settings explains why transfer learning with a pertained model is a popular practice.

6. Future Improvement and Recommendations

In our project, only 6 variables: open, high, low, close and percentage change of close prices are incorporated to train the models. In the financial market, there exist more meaningful or complicated financial indicators such as moving average, relative strength index (RSI), on-balance-volume (OBV) etc. for financial trading strategy

construction. In addition, the variables collected is on a daily basis scale. High frequency data with the scale of hours and minutes will not only substantially increase the data volume but also discloses more accurate short-term price variation.

Apart from increasing the amount and complexity of data, another possible improvement of performance of CNN is to use real images as input. Meaningful financial data can be plotted with refined settings of margins, color etc., following the rule to have more effective graphical representations of financial data.

Recurrent Neural Network (RNN) especially the Long Short-Term Memory (LSTM) may also be suitable for stock prediction as it stores "memory" about previous computations, which contributes to extract the pattern in time series data.

7. Conclusion

This report describes how we build different machine learning models including MLP and CNN models to predict the price or trend of Dow Jones Industrial Average on the 31st day based on the trading data of previous 30 days. The quantitative objective that all classification models have achieved accuracy above 52.71% has been satisfied. It is also observed that MLP can obtain a lower MSE than linear regression. The MSE and accuracy of the best models are reported and the mock trading results based on the constructed models for the period April 2 to April 13 have also been disclosed. The recommendations raised in the previous session can be our future direction. More data should be provided for mock trading to detect the prediction pattern of the models.

References

cs231n. (n.d.). CS231n Convolutional Neural Networks for Visual Recognition. Retrieved November 30, 2017, from http://cs231n.github.io/convolutional-networks/

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ketkar, N. (2017). Deep learning with Python: a hands-on introduction. Berkeley, CA: Apress. doi:10.1007/978-1-4842-2766-4

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research, 259*(2), 689-702.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324.

Leung, M. T., Daouk, H., & Chen, A. S. (2000). Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of Forecasting, 16*(2), 173-190.

S&P Dow Jones Indices. (2017, April). *Dow Jones Averages Methodology* [PDF]. Retrieved October 23, 2017, from https://us.spindices.com/indices/equity/dow-jones-industrial-average

Image Sources

Figure 2.3

CS231n. (n.d.). 3-layer neural network with three inputs, [Chart]. In *CS231n: Convolutional Neural Networks for Visual Recognition*. Retrieved October 24, 2017, from http://cs231n.github.io/neural-networks-1/

Figure 2.4

Brandon. (2016, August 18). *How do Convolutional Neural Networks work?* [Chart]. Retrieved October 24, 2017, from https://brohrer.github.io/how_convolutional_neural_networks_work.html

Figure 2.5

https://i.ytimg.com/vi/FTNNfba5CJw/hqdefault.jpg

Figure 2.6

https://www.google.com.hk/search?q=cnn+pooling+layer&safe=strict&source=lnms&tbm=isch&sa=X&ved=0ahUKEwirre6E2rbaAhVS6LwKHSFeDx4Q_AUICygC&biw=1133&bih=553#imgrc=_qdiQ0Wq2sQCiM: