Predicting High-frequency Stock Price Using Machine Learning Technique

Hyunwoo Roh
Department of Economics
University of Chicago

This dissertation is submitted for the degree of
Master of Arts

University of chicago 2020

I dedicate my dissertation work to my family and many friends.
A special feeling of gratitude to my loving parents, SoonRan and YoungHo.

Acknowledgements

I would like to express my gratitude to my thesis adviser Per Mykland whose expertise was invaluable in the formulating of the research topic and methodology in particular. His advice and guidance have been extremely helpful in terms of completing my thesis and for my own personal development as a graduate student. I would like to acknowledge Professor Christoper Roark as the second reader of this thesis, and I am gratefully indebted to his valuable comments on this thesis. I am profoundly grateful to Professor Niels Nygaard for his guidance of machine learning technique and his professional insights regarding its finance application. Discussions with Dr. Pham Ngoc Hoang Minh and Ziang Zhou have been very insightful. This accomplishment would not have been possible without them. All errors in this paper are mind alone.

Abstract

This paper addresses the problem of predicting the stock price using the high frequency data based on a machine learning approach. We study two things in this paper (1) comparison of the prediction performance among selected function classes with given look-back parameter in terms of the proposed evaluation measures in the process of finding the best in-sample empirical loss minimizer (2) the comparison of those results by changing the sampled frequency of financial time series data after obtaining an introduced set of high frequency data features extracted from the Trades and Quotes (TAQ) data. For the analysis of TAQ data, feature engineering involves the computation of 56 number of related features including market microstructure, statistical and technical indicator features. Re-estimation was done to improve the prediction accuracy for data models to obtain the predicted value every moving window. On the other hand, algorithmic models are used without re-estimation for the practical matter in that the time spent to train the model is often larger than the sampled frequency of the data. Moreover, the look-back parameter is introduced to cut off the irrelevant long past historical data.

Among the selected function class in the experiment, the results show that the PCA regression performs the best in terms of the mean directional accuracy and simple back testing for both NASDAQ100 index and TAQ data with given sampled frequencies (i.e., 3min, 5min, etc). Compared to previous studies using NASDAQ100, the results demonstrate that re-estimation and properly chosen look-back parameter improve the prediction performance in terms of proposed evaluation measures. When it comes to maximum draw down, which is a measure critical for risk management, DA-RNN rendered the smallest value and, thereby, was the best performing model for TAQ data for all time frequencies . We also provide DM statistics whose null hypothesis is that the accuracies of prediction values of any two given models will not be different. In case of TAQ data for all sampled frequencies, there is an evidence that we cannot reject the null hypothesis when comparing between PCA regression and DA-RNN model. Extensive experiments provide insights into properly evaluating the prediction performance of best in-sample empirical loss minimizer using the high frequency time series data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

When people or machines trade in the financial market, their order decisions (i.e., buy, sell or hold) are primarily based on certain types of valuable information (i.e., news data) or signals extracted from the stock price. Order decisions are made by many types of investors for their own purposes, and such diverse trading purposes are achieved through a wide spectrum of trading behaviors. Interestingly, even trading activity under identical goal ended up being realized in distinct manners because investors use different strategies subject to given constraint such as the amount of money they can invest, frequency of orders they can execute, types of data they can access to, and the tools they can implement. These differences bring conflicts on the view of price movement, which generates noisy and volatile price series.

Predicting stock prices, or time series forecasting in general, would have been much easier if investors were informed in prior of information that all other investors use, especially the type of data and strategy. However, there is no way to perfectly figure out which information or strategy is used but to observe the real time price fluctuations that reflect the aggregated and mixed opinion from all market participants. On top of this, ever-existing arbitrage forces in financial market make finding an alpha even more difficult, which leads to low signal-to-noise ratio [32]. As a result, time series prediction models are left to learn and memorize irrelevant information or random parts of the input data.

Moreover, as much as we worry about model learning noise, we need to worry about having a good result by chance. This is why financial time series prediction has been considered as one of the most challenging tasks. Nonetheless, researchers and practitioners still hope to find a pattern or signal that helps to predict the price series or its return series. The hope was reinforced by high-frequency data (e.g.,limit order book) and text data (e.g., news and financial statement) along with the advent of both higher computing power and state-of-the-art

machine learning algorithms.

In financial application, a widely-used type of high-frequency data set is Trades and Quotes (hereafter, TAQ) data, which is also called tick data or level 1 order book data. High frequency data often refers to any time series data which is recorded highly frequently such as in milliseconds or nanoseconds. Considering that first and last 30 minutes take up more than 50% of the total trades within a day, analyzing high frequency data now becomes almost inevitable to build an order strategy based on the careful consideration on the clear differences in terms of order arrival intensity throughout a given day.

Tick level data contains detailed information about every traceable public trading activity excluding the "dark pools" and, consequently, it is more useful in comparison to the most commonly used daily closing price. However, for the cost of additional information, we suffer from new challenges in data pre-processing and statistical analysis. As data appears much more frequently, high frequency data shows different characteristics and behaves very differently from the lower frequency time series. Those characteristics can be summarized with the following stylized facts: irregular time arrival, discreteness, diurnal patterns, and temporal dependence [16]. These facts present complications for analyses based on extant analysis tools.

To deal with such characteristics of high frequency data often summarized into a single term called market microstructure error, different approaches have been employed. Market micro-structure focuses on how prices adjust to new information and how the trading mechanism affects asset prices. [16] Statistical approach mainly deals with statistical difficulties regarding volatility estimation arising from discreteness, time endogeneity, market microstructure error and so on. Both approaches have successfully addressed market microstructure errors and at the same time brought deep understanding of TAQ data itself as well as structures underneath the market. In turn, they bring variety to the existing trading strategies and risk management tools. In this paper, we bring independently developed microstrcutrue approach and statistical approach together to introduce a set of features. The details of each feature will be discussed in Chapter 3.

Although some are not developed for the purpose of generating profitable trading strategy per se, they provide valuable information for predicting the price index and its movement. For the usage of those features in the prediction task, we introduce practical framework of time series prediction in Chapter 2. In the process of finding the best performance model, we compare different function classes including one of the state-of-the-art deep learning models for time series forecasting which is Dual Attention Recurrent Neural Network (DA-RNN). Such model is the variant of recurrent neural network (RNN) that is known to work well with sequential data such as text and time series. The list of the models used in the experiment will

be covered in Chapter 4.

This paper's contributions are as follows. The paper first walks through theoretical and practical issues on how to correctly perform time series prediction in machine learning context. The paper then introduces a feature set that incorporates various information in high-frequency research area ranging from market microstructure approach to statistical approach. Moreover, it is empirically worth trying various time frequencies to get a valid and meaningful result rather than just using single sampling frequency at a cost of time-consuming and laborious data cleaning work on TAQ data. Based on the result, we can observe how results change over different time frequencies, time look-back, and most importantly selected function class in terms of the proposed evaluation measures.

1.2 Motivation

Machine learning technique made a breakthrough on various applications from image classification to language translation. Among many, one of the most remarkable advancements was done by Google Deep Mind team as they conquered the complex board game 'Go' in the early 2016. The game of Go has been known for its enormous search space and difficulty of evaluating the board positions and moves. [43] Before AlphaGo beat the world strongest Go player 'Lee Sedol' at that time, no one could imagine that a machine could beat a world-class professional Go player.

People firmly believed that a machine cannot beat a world-class Go player's human intuition, creative mind, and reasonable guess based on experience. The only factor that seemed to favor machines, according to doubters, was their calculating ability. Alpha Go proved them wrong. It was shocking to see human professional players were incapacitated by artificial intelligent that combines Monte-Carlo simulation with value and policy networks to calculate a winning probability. [43] The shock was soon followed an exploding hope for machine learning to solve other problems in the world.

Unfortunately, machine learning techniques spread indiscriminately, even to areas where significant amount of domain knowledge based adjustment is required to properly apply machine learning. Typical examples include areas where training samples are not independent and identically distributed (hereafter, i.i.d). Therefore, without much consideration of characteristics underlying the data, using machine learning is as bad as not using it or even worse. Especially extreme care must be taken to use financial time series for the following reasons.

Stationarity (differentiation) and memory

Stationarity is an essential characteristics of the process and assumption that most of statistical models including machine learning rely on. The most widely used method of

transforming the process into stationary is by differentiating the process. Unfortunately, this can be harmful now that memory is partially removed during the stationary transformation. Due to such trade-off appearing in the financial time series where price series are believed to be non-stationary with memory while the returns series is stationary without memory, there are tremendous literature dealing with this issue [14]. Fractional differentiating, one of ways to get away from this issue, comes in handy to transform the process to be stationary with preserving the memory as much as possible. However, the question still remains as to how to define and quantify the memory as well as how to test non-stationarity of the process if possible.

Learning time and sampled time frequency

Under the i.i.d assumption, utilizing the data as many and recent as possible is ideal. However, it is not always the case that we can have both of them satisfied in time series forecasting unlike the usual machine learning setting. Although advanced computing power enables us to compute fast, at the same time, people started to use much larger data set on top of highly complicated models (e.g., deep learning), which makes learning and optimization take more than hours or even days. This becomes problematic for the real world operation. For example, predicting very near future such as one minute from now by utilizing up-to-date minutely data is practically not feasible if the time for learning goes beyond one minute. Together with look-ahead bias, this is another type of mistake easily made by researchers. Moreover, learning time is a critical issue for time series forecasting area where researchers may want to re-learn or re-estimate all model parameters for every forecasting to achieve the best possible predicted value or distribution.

Holding out a validation(test) set

In usual machine learning setup, we split the data into train, validation, and test set with user preferred weights to decide the each portion. Main purpose of this spit is to avoid the overfitting problem, which works well under the i.i.d assumption. This mechanism in time series forecasting, however, does not work as good as it should be now that we hold out the most recent data in learning process. In time series data, recent data points are believed to contain more valuable information than the long past. This issue becomes apparent when the process shows a strong non-stationarity where there is a strong time dependencies in sequence of data points. On the other hand, without validation set, it is easy to commit overfitting problem. Therefore, with or without keeping the most recent data out from the learning process, problem exists.

Performance of time series model itself is time series

Unlike the other ML tasks that are successfully addressed such as image classification, speech recognition, translation, etc., time series prediction is hard to guarantee the superiority of one model against others across time. For example, in natural language application, attention based transformer model excelled all other variants of RNNs in terms of accuracy and we firmly believe that it will be same for the future [31]. In other words, transformer model is believed to capture the time-invariant structure of natural language better than the others so that its superiority remains to be valid across time. Unfortunately, it is not applicable to the time series model. Performance of each model within the the large class of time series models changes over the time.

Best in-sample guess not an ability to foresee the future

Having an ability to foresee the future is fundamentally impossible unless we incorporate the elements from the future itself, which is often referred as look-ahead error in time series. Taleb explains this as a weaker form of law of iterated knowledge where it can be simply described by using tower rule (i.e., successive conditioning)[1]. Although Taleb is overly dismissive of prediction models as stated and also agreed by Aldous's review, many researchers and practitioners still want to seek the best in-sample model as a predictor to prepare the future using both classical financial econometrics methods and machine learning methods. All that matters is that we use time series prediction as the best in-sample guess and the predicted values from the chosen best model is further employed for other purposes such as trading system and risk managing tool.

These issues are all somewhat related to the following two fundamental questions in prediction task:

Can finite in-sample data represent the underlying process?

Do we expect future data points are (statistically) similar to observed training data?

These two questions are ultimately the essence of all kinds of statistical and machine learning analysis.

Firstly, if we do not assume that data represents the underlying process whatever that is, many statistical and machine learning analysis become meaningless. Such representability assumption is the minimal assumption to make. Without it, testability for the meaningful characteristics of the process (e.g., stationarity) is not guaranteed. Specifically, we are only able to test the degree of stationarity or non-stationarity under the explicit assumption for the

[1] $\mathbb{E}[\mathbb{E}[X \mid \mathscr{F}_{t+1}] \mid \mathscr{F}_t] = \mathbb{E}[X \mid \mathscr{F}_t]$ where $\mathscr{F}_t \subset \mathscr{F}_{t+1}$

class of diffusion (e.g., MA(1)) that the process belongs to. Secondly, for trained model or estimated model to be useful in conducting a prediction task, unseen future data should be similar to observed past data in terms of characteristic of the process that follows or distribution to be drawn from. This again reminds how financial time series has to be done more critically than it has been done so far, and this paper tries to discuss how it could have been done uncritically.

Chapter 2

General Framework of Financial Time-series Prediction

2.1 Background

In the real-world applications, we are faced with lots of prediction problems of which some are less troublesome than the others. Let's consider the following examples: weather forecast, number of COVID-19 infected, and stock prices. The first one, weather forecasting, is rather clearer about the way to do the forecasting than the other two now that there exists a physical process of atmosphere where the prediction model heavily relies on. Such underlying physical process remains stable across time, and it has been studied by meteorologists for a long time to describe the dynamics among factors affecting the weather. [48] Although there have been slight modifications on forecasting methods over a century from deterministic numerical weather prediction (NWP) to ensemble forecast methods up to now, physical model has been the main source of the prediction.

Such dynamics exist in the second example as well in a rather weak form where factors now become different types of agents (e.g., infected, susceptible, hospitalized, etc), which is also called as agents-based model. [1] Even with underlying physical process or dynamics among agents for both examples, however, there still exist a large uncertainties in prediction. The main concerns are initial condition and inadequate observations (e.g., measurement errors) as well as our limited understanding on the dynamics (e.g., fatality rate). [48] Due to the chaotic nature in the dynamics, controlling different set of model variables and parameters that govern the dynamics is crucial to make an accurate prediction based on simulation. Unfortunately, it is not always the case that such dynamics or relationships among factors (e.g., agents) exist.

Similarly, there have been previous attempts to explain the dynamics among different types of agents (e.g., institutional, individual and foreign investors) mostly focus on the impact and profitability of their trading behaviors using multivariate time series models (e.g., VAR) where the result is subject to changes across time and country. [2, 19] Stock price or its return series, however, does not have either underlying physical process or agent-based dynamics in general. Therefore, the examined dynamics in the finance literature is completely different cases of whether forecasting or COVID-19 for example in that dynamics in finance is mostly just a time dependent pattern that is much more vulnerable and chaotic than the latter.

What is left to be done is finding a pattern or signal from the past which we hope to be preserved over time, which is the main task of time series prediction. In other words, by carefully analyzing the past observations and its dependencies across time and variables, we develop a time series model describing the inherent structure of the time series process in which most of the cases rely on strong assumptions such as stationary and mixing. Unfortunately, these are often invalid assumptions let alone the $i.i.d$ and normality assumption on the noise terms. [23] Therefore, when using the time series model, we need to carefully look into which assumption the model depends on.

Regardless of different forecasting methods in various fields, eventually h-step forecasting is to obtain a series of predicted values for next h periods determined by research object. When we are interested in one-step forecasting, for example, h equals to one whereas h is larger than one for multiple-step forecasting. In a case where h equals to 30, it means we would like to acquire 30 number of predicted values. However, those values can refer to different things depending on the model setups, and the the sampling time frequency is one typical example. If the data is hourly temperature, the 30 number of predicted values usually refers to the temperatures for the next 30 hours. When we use the minutely stock returns data, it means those predicted values are next 30 minutes returns series. Other than the sampling frequency of the data, there are other factors worth taking a look at in the process of time series prediction.

Let's continue to assume that we obtained 30 predicted values. Those values can be interpreted differently according to various model setups utilized in the prediction process. First, we can obtain them by doing either one-step forecasting with 30 moving windows or 30-steps forecasting without moving window. Second, since we do not simply want to use the data of long past (e.g., 20 years ago) in predicting the next 30 minutes albeit the data is available, it is important to notify how much portion of past data was utilized among all the past available data. It directly affects on choosing the best prediction model. Third, for a selected forecasting model with a chosen subset of past data as an input, predicted values become different in terms of how often we update (i.e., re-estimate) the model settings such as parameters and

hyper-parameters especially under the moving window scheme. Therefore, we need to clarify those issues before starting to evaluate predicted values. In the following sections, we discuss previously introduced concepts in relation to a time series prediction framework including numerous prediction models and evaluation metrics.

2.2 Forecasting Horizons and Time Look-back

In this section, we discuss forecasting horizon and time look-backs. They are key factors to determine input and output dimension. Together with the moving window, forecasting horizon directly determines the number of predicted values, which determines an output dimension. Similarly, time look-back determines the number of data points used for the training set, which is an input dimension.

2.2.1 Output and Forecasting Horizon

Assume that we would like to do h-step forecast of the target variable y for the next m periods in rolling basis at time t. Then, the function, also called predictor, hypothesis, or a classifier provides us $m \times h$ number of different predicted values. In other words, we would obtain h different predicted target values for the each moving (or rolling) window where it slides from the time $t+1$ to time $t+m$. This particular setting is referred to as h-step forecast with m moving windows.

To start with the simplest case where $h = 1$ and $m = 1$ at time t, a single predicted value for time $t+1$ can be expressed as $\hat{y}_{t+1|t}$. This expression means that we forecast value of y_{t+1} given all the information up to time t where the information set refers to σ-algebra, which often expressed as Ω. If we extend the forecasting horizon from 1-step to h-steps with $m = 1$ at time t, we would obtain h different predicted values at time t as follows:

$$\hat{Y}_{h|t} = \left(\hat{y}_{t+1|t}, \hat{y}_{t+2|t}, ..., \hat{y}_{t+h|t}\right) \in \mathbb{R}^h$$

,which apparently does not need a moving window technique. These predicted values are what we often see in weather forecasting simulation (e.g., weather forecasting for the next 7 days). On the contrary, if we extend the number of moving windows with h fixed to one, we obtain each of m different predicted values at a time from t to $t+m-1$ as follows:

$$\hat{y}_{t+1|t}, \hat{y}_{t+2|t+1}, \cdots, \hat{y}_{t+m|t+m-1}$$

This is the result we obtain from one-step forecasting with m moving windows.

To combine the two, if we extend both h and m, we can summarize the predicted values as follows:

$$\begin{aligned}\hat{\mathbb{Y}}_{h,m,t} &= \left(\hat{Y}_{h|t}, \hat{Y}_{h|t+1}, \cdots, \hat{Y}_{h|t+m-1}\right)^T \in \mathbb{R}^{m\times h} \\ &= \begin{pmatrix} \hat{y}_{t+1|t}, \hat{y}_{t+2|t}, & \cdots & , \hat{y}_{t+h|t} \\ \hat{y}_{t+2|t+1}, \hat{y}_{t+3|t+1}, & \cdots & , \hat{y}_{t+1+h|t+1} \\ & \vdots & \\ \hat{y}_{t+m|t+m-1}, \hat{y}_{t+m+1|t+m-1}, & \cdots & , \hat{y}_{t+m+h|t+m-1} \end{pmatrix}\end{aligned} \tag{2.1}$$

This is an outcome matrix from h-step forecasting with m moving windows. We use this to test the model performance in comparison to the corresponding true values of target variable that will be realized in the future as $\mathbb{Y}$ in the following form:

$$\begin{aligned}\mathbb{Y}_{t,h,m} &= \left(Y_{h,t}, Y_{h,t+1}, \quad \cdots \quad , Y_{h,t+m}\right)^T \in \mathbb{R}^{m\times h} \\ &= \begin{pmatrix} y_{t+1}, y_{t+2}, & \cdots & , y_{t+h} \\ y_{t+2}, y_{t+3}, & \cdots & , y_{t+h+1} \\ & \vdots & \\ y_{t+m}, y_{t+m+1}, & \cdots & , y_{t+m+h} \end{pmatrix}\end{aligned} \tag{2.2}$$

, which is used to evaluate the predicted values from the best fitted model. Evaluation is done through the specificed loss function, l, as follows

$$l : F \times X \times Y \longrightarrow \mathbb{R}_+$$

for any set F (model) and some domain $X \times Y$, which eventually measures how different they are between actual values $\left(\mathbb{Y}_{t,h,m}\right)$ and predicted values $\left(\hat{\mathbb{Y}}_{h,m,t}\right)$. We will discuss more on this in later section.

2.2.2 Inputs and Time Look-back

Now, lets talk about the inputs for the model. Here, inputs are another name of independent variables and covariate. In forecasting process with m number of moving windows, accordingly we need m different set of inputs values X to train the model. As mentioned earlier, we do not want our model to be trained on the irrelevant data (i.e.,the very long past observations) so that it does not harm the performance. Therefore, for the sake of prediction accuracy, we

would like to restrict the number of past data points to be used if necessary by sacrificing the statistical power from the large sample size. We introduce a concept of time look-back that corresponds to the number of time lags allowing the past data points to be used. If we choose the look-back as 500 at current time t, for example, it means we only use a subset of data from time t to $t-499$ instead of whole available data.

To elaborate more, by letting the model analyze the only recent data points, we expect it to better capture a recent state of data generating process (i.e., complicated structure underneath the data). Considering that data generating process for time series data is believed to evolve over time and the value of data point vanishes with time, it is an appropriate to cut off the long past data and focus on recent observations instead. If the data has universal laws or structures over time in it, no matter how complex they are, we would like to use as much data as we can in order to capture them. In such case, we would expect to have better result with larger look-back by using all available data. If we choose to have a large time look-back with non-stationary time series, however, there is a high chance of ending up with accumulating and learning the noises. Since there is no theory for choosing optimal length of time look-back, we need to experiment with different look-backs, which remains as an empirical question.

Time look-back determines the dimension of input data. The most commonly used input data in time series forecasting takes a form of n variables consisting of independent variables and the history of target variable. Therefore we can write the input values X at time t as

$$
\begin{aligned}
X_t &= \left(\mathbf{x}_t^1, \mathbf{x}_t^2, \cdots, \mathbf{x}_t^{n-1}, \mathbf{y}_t\right) \in \mathbb{R}^{b\times n} \qquad (2.3)\\
\text{where each} \quad \mathbf{x}_t^1 &= \left(x_{t-b+1}^1, x_{t-b+2}^1, ..., x_t^1\right)^T \in \mathbb{R}^b\\
\mathbf{x}_t^2 &= \left(x_{t-b+1}^2, x_{t-b+2}^2, ..., x_t^2\right)^T \in \mathbb{R}^b\\
&\vdots\\
\mathbf{y}_t &= \left(y_{t-b+1}, y_{t-b+2}, \cdots, y_t\right)^T \in \mathbb{R}^b
\end{aligned}
$$

where b refers to time look-back, n refers to the number of features.

Since there are m number of rolling windows with l time look-back at time t, the first data point of each input set started to slide from time $t-b+1$ to time $t+m-b+1$. Therefore, we can express the input values for all moving windows as follows:

$$\mathbb{X} = (X_t, X_{t+1}, \cdots, X_{t+m}) \in \mathbb{R}^{b\times n\times m}$$

where $X_t, X_{t+1}, \cdots, X_{t+m}$ are from the equation (2.1) above. To avoid confusion from the notation below, we summarize the notations.

- Y : target variable, dependent variable, output

  - $y \in Y$: true value
    - Binary classification : $y = \{0, 1\}$
    - Regression problem : $y \in \mathbb{R}$
  - $\hat{y} \in Y$: predicted value

- X : features, regressors, independent variables, covariates, input
- S_n : Training set, in-sample observations
  - typical examples of training set are all available information up to time t: $((x_1, y_1), ...(x_t, y_t))$
  - (x_i, y_i) is a realization of random variables (X_i, Y_i)
- S_{out} : Test set, out-of-sample observations,
  - typical examples of test set are future data points : $((x_{t+1}, y_{t+1}), ...)$
- l : loss function
- F : function class or hypothesis class with realizability assumption
  - $f \in F$: model, mapping, algorithm, hypothesis
- $F_{\not\mathcal{A}}$: function class or hypothesis class without realizability assumption
- Φ : parameters, weights
- b : look-back, number of data points for each feature
- n : number of features
- m : number of moving (rolling) window, number of test points
- h : forecasting horizon and forecasting steps
- E : expectation over population
- $\mathbb{E}$: expectation over sample

2.3 Prediction Models or Algorithms

2.3.1 Data Model and Algorithmic Model

When conducting a forecasting task, there are many options in terms of choosing a model or mapping F. Among many existing models, we can group them into two: data model and algorithmic model as Breiman did in his paper. [10, 44] Each of them roughly corresponds to classical statistical models and machine learning models. To talk about data model first, it focuses on finding a latent data generating process governed by model parameters under the explicit assumption that data is a random process following a particular distribution. Parameters of the model are estimated by fitting them on data and the interpretation on selected set of parameters is valid only when assumptions hold. It means that if the underlying random process follows a known stochastic diffusion and the data generating process (DGP, hereafter) remains stable along time, data model will perform very well although such ideal data set rarely exists in a real world.

On the contrary, algorithmic approach cares less about underlying mechanism of DGP but rather concentrates on finding a way the algorithm generates the data. In other words, model parameters or weights are chosen to make the algorithm best explain the data unlike the parameters in data model governs the way data is generated from the given distribution assumption. Moreover, validation of model is done through calculating predictive accuracy. Such tendency to solely focus on predictive accuracy of the model is quite opposite to the data model whose validity is determined by goodness of fit and residual analysis where they play an important role on inference. Having little consideration on predictive accuracy does not necessarily mean that data model always has lower predictive accuracy in that there are certainly many situations where data models are necessary and ideal. [11]

Among many differences, it is noticeable to see the distinctive view points regarding a role of parameter for each model. Data modeling approach views a set of parameters as a way of controlling the data-generation process whereas the algorithmic approach views it as controlling how algorithm should learn from the data. They share same grounds in that both choose a set of parameters and weights that minimize the specified loss. However, algorithmic approach detects the structure in $\widehat{Y}$ whereas the data modeling approach produce an estimate of parameter $(\widehat{\beta})$ at least asymptotically good enough to explain the underlying relationship between X and Y, i.e., algorithmic approach belongs as a part of the toolbox marked $\widehat{Y}$ rather than the more familiar $\widehat{\Phi}$ compartment often represented by $\widehat{\beta}$ for the regression problem. [37]. Lastly, algorithmic approach works particularly well with 'high dimensional' or 'wide' data where there are more input variables than the data points in contrast to its performance with the 'long' data where the number of data points is greater than that of input variables.

[11, 35]

2.3.2 How to Choose a Best Prediction Model

The end goal of one-step prediction task in practice is to first find the best set of parameters associated with the best model and number of look-back given a specified loss function l using m number of moving window $i \in [1,m]$ from the past at time t as follows: [31, 35, 36]

$$f^*_{\Phi,t} = arg\min_{\Phi,b} \underbrace{\sum_{i=1}^{m} l\left(f\left(X_{t-i};\Phi\right),Y_{t-i}\right)}_{\text{expected in-sample loss}} \quad \text{over} \quad \underbrace{f \in F_{\mathscr{A}}}_{\text{function class}} \quad \text{subject to} \quad \underbrace{R(f) \leq c}_{\text{complexity restriction}} \tag{2.4}$$

where b is a time look-back parameter, $arg\min$ is the minimum within computational limit (often called computational minimum), $F_{\mathscr{A}}$ refers to a function class within an algorithmic limit, X and Y to the observed samples defined as in equation (2.1), l to a loss function, Φ to a parameter set according to different type of function, and $R(f)$ is related to the hyper-parameters specific to different types of function f. The function class can be thought of as a set of all kinds of predictor that can be implemented by software program (e.g. Python or R) written at most 10^9 bits of the code. [42] Before diving into the implementation, we need to understand the following three limitations that we face in order to have a better theoretical insight behind the machine learning. [35]

Limitations of Machine Learning

- Algorithmic limitation (realizability assumption)
  - We don't know that the true function class (if exists) is included in the function class that we search over. If there is no algorithmic limitation, i.e., realizability assumption holds, there exists a function $f^* \in F$ such that $l\left(f^*(X),Y\right) = 0$. where the training samples are drawn $i.i.d$ from the true underlying distribution. [42] With the algorithmic limit, however, we just hope that the chosen or well known function classes incorporate the true function class explaining the data. Therefore, for the further analysis, we separate the function class into two that are $F_{\mathscr{A}}$ and F representing the function class with or without algorithmic limitation respectively.
- Data knowledge limitation
  - We only have an access to the sample points with limited understanding about the true population. In other words, we cannot fully capture what the true underlying

data-generating joint distribution P_{XY} is over the (X,Y) through the sample points. Such data knowledge limitation is critical now that the learning guarantee of f^* depends on the relationship samples and population. [42] That is why we at least need to assume that samples are drawn $i.i.d$ from the P_{XY}. It brings us a better understanding about the relationship between the sample and population distribution as each independently sampled x from same P_{XY} is mapped to y with true function f if it exists. To deal with data knowledge limitation, however, this assumption does not hold in general so that data knowledge limitation becomes critical.

- Computational limitation
  - Seeking in-sample loss minimizing function is often computationally intractable. [34] It is necessary to consider a computational limit while finding the best parameter or weights especially when the typical machine learning involves three problems that are optimization, integration, and fixed-point computation [13]. When we implement prediction task using computer software, we just hope that a set of parameters within the computational limit is as close as to the one beyond the computational limit. We do not discuss such computational challenges here as it is central to a computer-science treatment of the subject.

Due to the limitations above, learning is simply not possible without making assumption to overcome the limitations. The following assumptions are the ones commonly used in machine learning area: [33]

1. We don't know the true joint probability distribution P_{XY} at the training stage.
2. No explicit assumption is made on the unknown joint probability distribution P_{XY} (Pop) or the underlying stochastic process of Y. Many existing statistical analyses rely on the explicitly made distributional assumption, which works well if it holds. Otherwise, we need to use statistical learning theory that works in an agnostic way.
3. Data points (x,y) must be independently and identically sampled from the Pop. It can be supplementary to the data knowledge limitation.
4. P_{XY} is static, i.e., it does not change over time so that the future data points will have properties similar to the ones already observed and used in the training stage.
   (a) Not suitable for problems involving time dependent data (e.g., time series data)
   (b) We will generalize this assumption for non-stationary or non-mixing data [citation]

Imagine that we are living in an ideal world without such limitation and we know the underling true joint probability from which the data points are *i.i.d* drawn. Then, the ideal version of finding in-sample minimizing function would be

$$f_{\Phi}^{*} = arg\min_{\Phi} \underbrace{E_{(X,Y)\sim Pop}\left[l\left(f\left(X_t;\Phi\right),Y_t\right)\right]}_{\text{expected out-of-sample loss}} \quad \text{over} \quad \underbrace{f \in F}_{\text{function class}} \quad \text{subject to} \quad \underbrace{R(f) \leq c}_{\text{complexity restriction}}$$

Again, the selected best function along with best parameter set achieves the computational minimum of expected in-sample loss within the chosen function class given the constraint on model complexity.

Once we have the best prediction model in our hand, prediction can be easily done by using the selected Φ and f^* as follows:

$$\begin{aligned}\hat{Y}_{\cdot|t} &= \left(\hat{y}_{t+1|t}, \hat{y}_{t+2|t}, ..., \hat{y}_{t+h|t}\right) \\ &= f^*(X_t; \Phi_i^*)\end{aligned}$$

where we employed h step forecasting at time t.

2.3.3 Different Types of Losses

We consider 4 different types of losses. Out of four, we already discussed the most important one above in equation 2.2, an expected empirical loss. This is only implementable loss and the other three are not calculable. There are three more and they are mostly used for the purpose of theoretical analysis. Loss is typically used to choose a proper function in a learning process. Within a class of existing prediction models, we want to choose one out of $F_{\mathscr{A}}$ using the training set S_n. Since the selected best function is dependent on the in-sample, we denote it as

$$f_n\left(x\right) = f\left(x; S_n\right)$$

where the subscript symbol n denotes the dependence on the training set size and look-back, which indicates that it is a function of training data set. [12, 35] Considering that f_n is a function of random sample, it again is a random quantity (random vector).

Then, the loss is calculated as

$$l\left(f_n\left(x\right), y\right)$$

where we wish it to become as small as possible. To remind the end goal of the prediction task, we want it to be small for the future data points so that our predicted value gets as close as to the real data points that will be realized in the future. Since we do not have access to

future data set on which the selected function performs the best, we can only choose one that minimizes the averaged in-sample loss. As we choose a function $f \in F$ using the training data to do so, notice that selected function f_n^* has to be in-sample dependent. To have better understanding on this, we discuss 4 losses in the following.

1. Population Loss of f

$$L(f) = E_{(x,y)\sim Pop}\left[l\left(f(x),y\right) \mid f\right]$$

This quantity is an expectation of instantaneous loss over joint population distribution $X \times Y$ and the function is dependent on random training sample. It measures how well, on average, the predictor sample dependent f performs with respect to the chosen loss function. Conditional statement is there to ensure that the definition is sensible even if f is a random quantity especially when the function is dependent on random training sample as in f_n. [12]

2. Empirical Loss of f

$$\widehat{L}(f) = \mathbb{E}_{(x,y)\sim Train}\left[l\left(f(x),y\right) \mid f\right] = \frac{1}{N}\sum_{(x,y)\in Train}\left[l\left(f(x),y\right) \mid f\right]$$

This quantity is a sample version of the population loss. Since we do not know the underlying distribution, we can only calculate this amount with the given training data. Consider the above two losses are random quantity when f is dependent on training sample (e.g., f_n). To deal with this randomness, we can simply take an average over all possible training data set and associated functions to get the other two non-random losses respectively in the following. When the function is not dependent on training sample (e.g., f^* or $f_{\mathscr{A}}^*$ below), however, there are no differences after taking an expectation so that $E\left[L(f)\right] = L(f)$.

3. Expected Population Loss of Training Sample Dependent f

We take expectations over both random sample and associated function and by the law of iterated expectation ($L.I.E$) can get

$$\begin{aligned} E\left[L(f)\right] &= E\left[E_{(x,y)\sim Pop}\left[l\left(f(x),y\right) \mid f\right]\right] \\ &= E_{(x,y)\sim Pop}\left[l\left(f(x),y\right)\right] \end{aligned}$$

This quantity is an out-of-sample evaluation of predictor f chosen from F using in-sample data for a given loss function. Compared to this, $L(f)$ is a finer measure since it takes into consideration the specific realization of the stochastic process and does not take an average over all possible training data set and associated functions.

4. Expected Empirical Loss of training sample dependent f

This is a sample version of the expected population loss of f, which is actually calculated when we implement with statistical package (e.g., Python or R) using a computer. Similarly to above, by taking an expectation and by the law of iterated expectation, we can get

$$\begin{aligned} E\left[\widehat{L}(f)\right] &= E\left[\mathbb{E}_{(x,y)\sim Train}\left[l\left(f(x),y\right) \mid f\right]\right] \\ &= \mathbb{E}_{(x,y)\sim Train}\left[l\left(f(x),y\right)\right] \end{aligned}$$

This is an in-sample evaluation of predictor f chosen from F.

2.3.4 Best Functions Respect to "Expected Population Loss" and "Expected Empirical Loss"

Using the last two defined above, we can specify the best function associated with each loss from both collection F and $F_{\mathscr{A}}$. Then, we ended up having 4 best functions as follows:

- $f^* = \arg\underbrace{\min}_{f\in F} E\left[L(f)\right] = arg\underbrace{\min}_{f\in F} E_{(x,y)\sim Pop}\left[l(f(x),y)\right]$
  - Best function with respect to expected population loss that is **beyond** algorithmic limit
  - This selected function is **independent** of training sample, S_n
- $f^*_{\mathscr{A}} = \arg\underbrace{\min}_{f\in F_{\mathscr{A}}} E\left[L(f)\right] = arg\underbrace{\min}_{f\in F_{\mathscr{A}}} E_{(x,y)\sim Pop}\left[l(f(x),y)\right]$
  - Best function with respect to expected population loss that is **within** algorithmic limit
  - This selected function is **independent** of training sample, S_n
- $f^*_n = \arg\underbrace{\min}_{f\in F} E\left[\widehat{L}(f)\right] = arg\underbrace{\min}_{f\in F} \mathbb{E}_{(x,y)\sim Train}\left[l(f(x),y)\right]$
  - Best function with respect to expected empirical loss that is **beyond** algorithmic limit
  - This selected function is **dependent** of training sample, S_n
- $f^*_{\mathscr{A},n} = \arg\underbrace{\min}_{f\in F_{\mathscr{A}}} E\left[\widehat{L}(f)\right] = arg\underbrace{\min}_{f\in F_{\mathscr{A}}} \mathbb{E}_{(x,y)\sim Train}\left[l(f(x),y)\right]$

  - Best function with respect to expected empirical loss that is **within** algorithmic limit
  - This selected function is **dependent** of training sample, S_n

2.3.5 Evaluations of 4 Best functions (Analysis Toolkit)

We have 4 specific best functions defined above. Now it is time to evaluate those 4 functions in two ways.

- Evaluate over population, which is not calculable (i.e., implementable). This is for the sake of theoretical analysis.
- Evaluate over sample, which is calculable.

We ended up having 8 different evaluations as is shown below.

2.3.5.1 Evaluation over Out-of-sample

We evaluate the four selected functions on the out-of-sample so that this quantity is not calculable in practice. However, it is useful to understand them in theoretical terms for the purpose of understanding the key concepts in machine learning (e.g., overfitting).

- $L(f^*) = E_{(x,y)\sim Pop}\left[l\left(f^*(x),y\right)\right] = E_{(x,y)\sim Pop}\left[l\left(f^*(x),y\right) \mid S_n\right]$: Irreducible population loss of f^*
  - Best out of sample function beyond algorithmic limit evaluated using out of sample
  - Second equality works because the function f^* is independent of training sample
  - Not calculable because we do not know the population distribution
- $L(f^*_{\mathscr{A}}) = E_{(x,y)\sim Pop}\left[l\left(f^*_{\mathscr{A}}(x),y\right)\right] = E_{(x,y)\sim Pop}\left[l\left(f^*_{\mathscr{A}}(x),y\right) \mid S_n\right]$: Irreducible population loss of $f^*_{\mathscr{A}}$
  - Best out-of-sample function within algorithmic limit evaluated using out-of-sample
  - Second equality works because the function $f^*_{\mathscr{A}}$ is independent of training sample
  - Not calculable and this is also independent of randomly drawn samples
- $L(f^*_n) = E_{(x,y)\sim Pop}\left[l\left(f^*_n(x),y\right) \mid S_n\right]$
  - Best in sample function beyond algorithmic limit evaluated using out of sample

  - We use conditional statement here because f_n^* depends on randomly drawn training data set.
  - Not calculable and best in sample function dependent on randomly drawn training data set

- $L(f_{\mathscr{A},n}^*) = E_{(x,y)\sim Pop}\left[l\left(f_{\mathscr{A},n}^*(x),y\right) \mid S_n\right]$
  - Best in-sample function within algorithmic limit evaluated using out-of-sample
  - We use conditional statement here because $f_{\mathscr{A},n}^*$ depends on randomly drawn training data set.
  - Not calculable and best in sample function dependent on randomly drawn training data set

By construction, the following inequality holds:

$$L(f^*) < L(f_{\mathscr{A}}^*) < \min\left(L(f_n^*), L(f_{\mathscr{A},n}^*)\right)$$

2.3.5.2 Evaluation over In-sample

We evaluate the four selected functions on the in-sample. The last quantity, $\widehat{L}(f_{\mathscr{A},n}^*)$, is only computable among all.

- $\widehat{L}(f^*) = \mathbb{E}_{(x,y)\sim Train}\left[l\left(f^*(x),y\right)\right] = \mathbb{E}_{(x,y)\sim Train}\left[l\left(f^*(x),y\right) \mid S_n\right]$
  - Best out of sample function beyond algorithmic limit evaluated using in-sample
  - Not computable
- $\widehat{L}(f_{\mathscr{A}}^*) = \mathbb{E}_{(x,y)\sim Train}\left[l\left(f_{\mathscr{A}}^*(x),y\right)\right] = \mathbb{E}_{(x,y)\sim Train}\left[l\left(f_{\mathscr{A}}^*(x),y\right) \mid S_n\right]$
  - Best out of sample function within algorithmic limit evaluated using in-sample
  - Not computable
- $\widehat{L}(f_n^*) = \mathbb{E}_{(x,y)\sim Train}\left[l\left(f_n^*(x),y\right) \mid S_n\right]$
  - Best in sample function beyond algorithmic limit evaluated using in-sample
  - Not computable
- $\widehat{L}(f_{\mathscr{A},n}^*) = \mathbb{E}_{(x,y)\sim Train}\left[l\left(f_{\mathscr{A},n}^*(x),y\right) \mid S_n\right]$
  - **Best in sample function within algorithmic limit evaluated using in-sample**

– **Computable**

By construction, the following inequality holds:

$$\widehat{L}(f_n^*) < \widehat{L}(f_{\mathscr{A},n}^*) < \min\left(\widehat{L}(f^*), \widehat{L}(f_{\mathscr{A}}^*)\right)$$

This is because when evaluating over in-sample, best in-sample functions (f_n^* and $f_{\mathscr{A},n}^*$) are better than the best out-of-sample functions (f^* and $f_{\mathscr{A}}^*$). Finally, we have analysis toolkit to understand the process underneath the prediction.

2.4 Understanding of Over-fitting

Back to the main equation, based on the data we have observed, we would like to make the following quantity as small as possible:

$$L(f_{\mathscr{A},n}^*) = E_{(x,y)\sim Pop}\left[l\left(f_{\mathscr{A},n}^*(x), y\right) \mid S_n\right]$$

which is an out-of-sample loss of best in-sample function. To deal with the randomness that we conditioned on, we take a expectation to get the following by law of iterated expectation:[12, 35]

$$\mathbb{E}_{(x,y)\sim Train}\left[L(f_{\mathscr{A},n}^*)\right] = \mathbb{E}_{(x,y)\sim Train}\left[E_{(x,y)\sim Pop}\left[l\left(f_{\mathscr{A},n}^*(x), y\right) \mid S_n\right]\right] = E_{(x,y)\sim Pop}\left[l\left(f_{\mathscr{A},n}^*(x), y\right)\right]$$

and this again can be decomposed into three parts as follows:

$$\begin{aligned}
\mathbb{E}_{(x,y)\sim Train}\left[L(f_{\mathscr{A},n}^*)\right] &= \mathbb{E}_{(x,y)\sim Train}\left[L\left(f_{\mathscr{A},n}^*\right) + L\left(f_{\mathscr{A}}^*\right) - L\left(f_{\mathscr{A}}^*\right) + L(f^*) - L(f^*)\right] \\
&= \mathbb{E}_{(x,y)\sim Train}\left[L\left(f_{\mathscr{A},n}^*\right) - L\left(f_{\mathscr{A}}^*\right)\right] + L\left(f_{\mathscr{A}}^*\right) + L(f^*) - L(f^*) \\
&= \underbrace{L(f^*)}_{\text{Irreducible Error, } \geq 0} + \underbrace{L\left(f_{\mathscr{A}}^*\right) - L(f^*)}_{\text{Approximation Error, } \geq 0} \\
&\quad + \underbrace{\mathbb{E}_{(x,y)\sim Train}\left[L\left(f_{\mathscr{A},n}^*\right)\right] - L\left(f_{\mathscr{A}}^*\right)}_{\text{Estimation Error, } \geq 0}
\end{aligned}$$

where there are irreducible error, estimation error, and the approximation error.

- Irreducible Error [18, 35]

  We cannot reduce this error. In statistical learning theory, this is equal to Bayes risk, i.e., minimum risk that one can ever achieve. It is related to excess risk whose value is obtained by subtracting this from $\mathbb{E}_{(x,y)\sim Train}\left[\widehat{L}(\hat{f}_{\mathscr{A},n}^*)\right]$, which has been studied to

find an upper bound that guarantees the learning (e.g. Probably Approximately Correct Learning). [9, 12, 39, 42]

- Approximation Error

  This error measures how well the selected best function within a function class $F_{\mathscr{A}}$ approximates the best function within function class F. In other words, it quantifies how much loss we have as we restrict ourselves to a specific function class, which means that the loss becomes zero under realizability assumption [42] It is noticeable that this quantity is not dependent on the sample size so that choosing the best in-sample function from the data has nothing to do with this term. Instead, approximation error is dependent on the functional class and underlying unknown population distribution of (X,Y), which makes the rate of convergence of the approximation error arbitrarily slow without explicit assumptions on target (e.g., Lipschitz smooth). [9, 12]

- Estimation Error

  The estimation error occurs because the empirical loss (i.e., training error) is only an estimate of the true loss. It implies that the predictor minimizing the empirical risk is only an estimate of the predictor minimizing the true risk. [42] Unlike approximation error, we can derive an upper bound for this error without explicit assumption about the distribution of training data using all sorts of concentration inequalities and convergence theorems. It is noticeable that this quantity is dependent on both training set size and the size of functional class (i.e., complexity).

2.4.1 More on Estimation Error

We further decompose the estimation error to have a better understanding about the overfit.

$$
\begin{aligned}
&\mathbb{E}_{(x,y)\sim Train}\left[L\left(f^*_{\mathscr{A},n}\right)-L\left(f^*_{\mathscr{A}}\right)\right] \\
=&\mathbb{E}_{(x,y)\sim Train}\left[L\left(f^*_{\mathscr{A},n}\right)-L\left(f^*_{\mathscr{A}}\right)+\widehat{L}\left(f^*_{\mathscr{A},n}\right)-\widehat{L}\left(f^*_{\mathscr{A},n}\right)+\widehat{L}\left(f^*_{\mathscr{A}}\right)-\widehat{L}\left(f^*_{\mathscr{A}}\right)\right] \\
=&\underbrace{\mathbb{E}_{(x,y)\sim Train}\left[\widehat{L}\left(f^*_{\mathscr{A},n}\right)-\widehat{L}\left(f^*_{\mathscr{A}}\right)\right]}_{\text{In-sample overfit, } \leq 0}+\underbrace{\mathbb{E}_{(x,y)\sim Train}\left[L\left(f^*_{\mathscr{A},n}\right)-\widehat{L}\left(f^*_{\mathscr{A},n}\right)\right]}_{\text{Unseen overfit, } \geq 0}+\underbrace{\widehat{L}\left(f^*_{\mathscr{A}}\right)-L\left(f^*_{\mathscr{A}}\right)}_{\text{Sample Size \& Non } i.i.d \text{ error, } \geq 0}
\end{aligned}
$$

Eventually, in an estimation error, what we would like to manage is the first two terms. However, there is a trade-off between the two. First, in-sample overfit becomes problematic because wrong function (i.e., best in-sample function) is supposed to look better than true function when evaluated on in-sample data. Unseen overfit is again erroneous when we fit well

on in-sample, some of that "fit" is overfit from the out-of-sample point of view although it is unseen:[35]

$$\underbrace{L\left(f^*_{\mathscr{A},n}\right)}_{\text{unobserved out-of-sample error}} = \overbrace{\widehat{L}(f^*_{\mathscr{A},n})}^{\text{observed in-sample error}} + \underbrace{\left(L\left(f^*_{\mathscr{A},n}\right) - \widehat{L}(f^*_{\mathscr{A},n})\right)}_{\text{Unseen overfit}}$$

Therefore, the calculable $\widehat{L}(f^*_{\mathscr{A},n})$ alone cannot control the overfit well enough even under the assumptions we made above.

"Here comes the regularization on model complexity"

Regularization on model complexity in equation 2.2, $R(f)$, takes a different form according to different models. Examples will be covered in the later section.

2.4.2 Trade-off

From above, we have seen how to decompose $\mathbb{E}_{(x,y)\sim Train}\left[L(f^*_{\mathscr{A},n})\right]$ into estimation error and approximation error. Then we further decompose the estimation error into three parts. We plug everything back to get

$$\mathbb{E}_{(x,y)\sim Train}\left[L(f^*_{\mathscr{A},n})\right] = \underbrace{L(f^*_{\mathscr{A}}) - L(f^*)}_{\text{Approximation Error}} + \underbrace{L(f^*)}_{\text{Irreducible Error}} + \underbrace{\widehat{L}(f^*_{\mathscr{A}}) - L(f^*_{\mathscr{A}})}_{\text{Sample Size \& Non } i.i.d \text{ error}}$$
$$+ \underbrace{\mathbb{E}_{(x,y)\sim Train}\left[\underbrace{\widehat{L}(f^*_{\mathscr{A},n}) - \widehat{L}(f^*_{\mathscr{A}})}_{\text{by construction, } \leq 0}\right]}_{\text{In-sample Overfit}} + \underbrace{\mathbb{E}_{(x,y)\sim Train}\left[\underbrace{L\left(f^*_{\mathscr{A},n}\right) - \widehat{L}(f^*_{\mathscr{A},n})}_{\text{by construction, } \geq 0}\right]}_{\text{Unseen Overfit}}$$

where we focus on the first term and the last two terms. This is because we cannot reduce the irreducible error as its name implies and the third term becomes smaller as sample size gets bigger under *i.i.d* assumption. For the rest three terms that are approximation error and estimation error, we can make each component smaller although there is a trade-off for reducing all of them simultaneously . Moreover, under the certain conditions, we can find an upper-bound for each term so that we can guarantee the learning [9, 12, 31, 39, 42]. To understand the trade-off, we consider the following two cases:

- Complexity constraint threshold of algorithm $\uparrow \equiv$ Enlargement of the set of function class (complicated)
  $\Longrightarrow$ Approximation error $\downarrow$, In-sample Error $\downarrow$, Unseen Overfit $\uparrow$

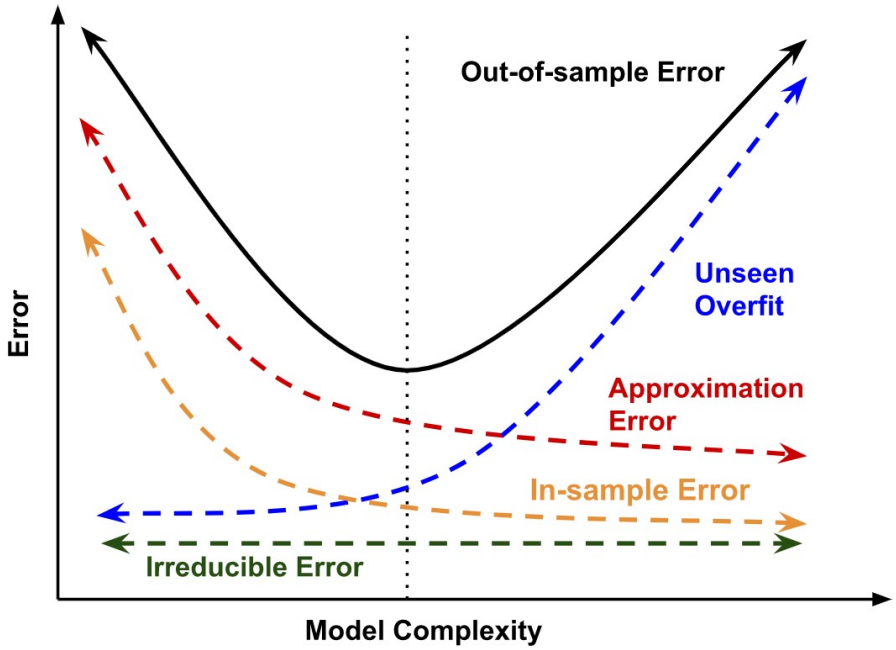

Figure 2.1: Bias-Complexity Trade-off

- Complexity constraint threshold of algorithm $\downarrow \equiv$ Reduction in the set of function class (simpler)
  $\Longrightarrow$ Approximation error $\uparrow$, In-sample Error $\uparrow$, Out-of-sample Overfit $\downarrow$

Therefore, we need a regularizer that controls the model complexity. The visualization of the trade-off is in Figure 2.1.

2.5 What Happens for the Non-stationary Financial Time series Data?

When the data is not stationary, there are a few extra concerns. First, from the theoretical point of view, it is not easy to achieve the learning guarantee as in the case under the *i.i.d* assumption. Under *i.i.d* assumption along with fixed (i.e., static) distribution, achieving an upper bound for estimation error or out-of-sample error in general has been studied for long time using concentration inequalities. Specifically, it covers many prediction problems including (multi-class) classification, regression, and density estimation in an agnostic way even with infinite function class. [34] Various attempts have been done to relax the *i.i.d* assumption

by assuming an underlying process as stationary or mixing. A discussion of those studies is well introduced in Kuznetsov and Mohri's paper and they further extend it to present data-dependent learning bounds for the general case of non-stationary and non-mixing processes. [23] Although they provide general theoretical learning guarantee for realistic settings by employing path-dependent generalization error and discrepancy measure, they only evaluate their predicted values in comparison to the most basic ARIMA model using mean squared error (MSE) alone.

Additionally, most of the existing statistical tests of stationarity rely on the additional assumption about the underling processes such as $AR(p)$. [41] Only when the assumption about the underlying process holds, the analysis becomes valid. Otherwise, we cannot construct the test statistics to use for the unit-root test (e.g. Dickey-Fuller). For a fortuitious case where the best in-sample function happens to be $AR(p)$, then the unit-root test is meaningful since the underlying process can be concluded as the stationary AR process. However, without an explicit assumption about the class of diffusion, we cannot test the stationarity of the process per se, let alone the non-stationarity. Non-stationarity is much harder to be concluded simply because we do not know in prior which assumption is violated when stationarity test fails. Rejecting the stationarity can imply the following three things starting from the worst: the finite training data is not long enough to represent the underlying diffusion, the underlying process of training data is different from what we assume, and the underlying process is non-stationary.

2.6 Function Classes and Its Parameters

For the time series prediction, we need to consider a large function class from simple to complex one. From the empirical results in [29, 30], it is shown that; 1) sophisticated or complex methods do not necessarily provide more accurate forecasts than the simpler ones, and 2) superiority of combination (ensemble predictor) is confirmed. Therefore, we consider large class of functions and in the Table 2.1 the incomplete summary of prediction models is introduced as in [35, 36].

2.6.1 Availability or Necessity of Re-estimation

Forecasting analysis has been done in two different manners. The main difference between the two is whether we re-estimate the model every time as we move forward in test sets.

- One-step forecast with re-estimation every rolling test window: the model parameters are re-estimated every time as we move forward.

Function class F	**Parameters** Φ	**Regularizer** $R(f)$
Global Predictors		
Linear $\beta^T X$ and g(linear)	$\beta \in \mathbb{R}^n$	$\| \beta \|_0 = \sum_{j=1}^n I\{\beta_j \neq 0\}$ $\| \beta \|_1 = \sum_{j=1}^n \| \beta_j \|$ $\| \beta \|_2 = \sum_{j=1}^n \| \beta_j^2 \|$
State Space Models (e.g., ARMA)	Associated parameter set (e.g., $\beta \in \mathbb{R}^{p+q}$)	Associated hyper parameter (e.g., Order of AR(p) and MA(q))
Local Predictors		
Decision tree	Threshold values at each node	Depth, minimal leaf size, number of nodes/leaves, information gain at split
Nearest Neighbors	Distant Metrics	Number of neighbors
Random forest	Parameters of multiple trees	Number of trees, complexity of trees number of features for each tree
Kernel Regression	Type of kernel	Kernel bandwidth (Smooth factor)
Mixed Predictors		
FFNN (Feed Forward)	$[W,b] \in \mathbb{R}^{g_1(n,N_1,N_2)}$	Number of hidden layer (N_1), Number of units for each hidden layer ($N_2 \in R^{N_1}$), Structure on neurons and layers (g_1)
RNN (Recurrent)	$[W,b] \in \mathbb{R}^{g_2(n,N_3,N_4)}$	Number of FFNN (N_3) , Number of hidden units (N_4), Structure on neurons, layers, and RNN cells (g_2)
CNN (Convolutional)	$[W,b] \in \mathbb{R}^{g_3(n,N_5,N_6)}$	Filter size ($N_5 \times N_6$), Structure on neurons, layers, and CNN cells (g_3)
Splines (Regression)	$\beta \in \mathbb{R}^{c+k+1}$	Number of knots(c), Order of spline(k) Location of knots, Choice of basis function
Ensembles		
Bagging	Parameters of weak (multiple) predictors & Weights on Boosting and Stacking	Number of draws, Size of bootstrap samples, Individual regularization hyper-parameter
Boosting		Learning rate, Number of iteration Individual regularization hyper-parameter
Stacking		Meta-model to decide how to combine weak predictors, set of hyper-parameters of chosen Meta-model

Table 2.1: Incomplete Summary of Model Classes

- One-step forecast without re-estimation every rolling test window: once model is trained for long time, it is used for the following moving window.

The problem is that to re-estimate a set of parameters using the most recent data point is not always practically implementable. This is simply because there are cases where the time it takes to estimate is longer than the time frequency of the data, especially in high-frequency data. For example, when we are interested in forecasting one minute ahead, we cannot re-train the model every time if it takes more than one minute to re-estimate the parameters. It goes the same to hyper-parameter tuning, which also takes a lot of time when model has lots of hyper-parameters. Despite of the trade-off between the training time and availability of re-estimation, in dealing with financial data especially with high frequency data, re-estimation is necessary.

2.7 Evaluation of Predicted Values

The forecasting performance of the models are evaluated using a number of widely used evaluation metrics. We introduce widely used conventional ones along with other evaluation metrics that are especially useful in evaluating the predicted values from the financial models. For the sake of simplicity, we consider the h step forecasting without moving windows. Then, we have the forecast error as follows:

$$\begin{aligned} error_{t,h} &= Y_{h,t} - \hat{Y}_{h|t} \in \mathbb{R}^h \\ &= (e_{t+1}, e_{t+2}, \cdots, e_{t+h}) \end{aligned} \tag{2.5}$$

2.7.1 Conventional Evaluation Measures

In time-series prediction, evaluating the predicted values is one of the most important tasks because after selecting the best in-sample model, we would like to test it to see whether it forecasts well on the new data set. The most widely used forecast error measures can be categorized into scale-dependent, percentage, and relative measures. [22] The two most popular measures are root mean square error (RMSE) and mean absolute error (MAE) where both are scale-dependent. Since they are dependent on scale that varies across different types of data, we cannot use it for the comparison across data sets unless their scales are unified. If we only evaluate different models on a single data set, scale dependency is not a big problem. To deal with this scale dependency issue, percentage-based measure and relative based measure are developed, which are scale-independent.

Such conventional measures, whether it be scale dependent or not, respect the view that forecast evaluation should concentrate on all large disturbances without much consideration on directional accuracy, although predicting price movement plays an important role on many situations including investment decision and economic policies. [5] Even when the utilized model is designed to be trained for a regression tasks instead of classification, it is still valuable to track the directional accuracy of the generated predicted values in comparison to other base models including classification model. Typically, consecutively accumulated small errors in a wrong direction might harm significantly in finance application (i.e., high-frequency trading) and thus the most commonly used evaluation metric in finance is back-testing and mean directional score (MDS) since they consider both direction and magnitude of the forecasting error.

2.7.2 Directional Accuracy Measure Revisited

To consider the directional error, this paper applies a standard directional accuracy measure and its variants [5, 8]. We first have to define the forecasting direction (FD) and actual direction (AD). Using the notation defined above in equation , we can express FD and AD as follows:

$$\begin{aligned} \mathrm{FD}(i) &= \operatorname{sign}\left(\hat{Y}_{h|t} - Y_{h,t-1}\right) \in \mathbb{R}^h \\ \mathrm{AD}(i) &= \operatorname{sign}\left(Y_{h,t} - Y_{h,t-1}\right) \in \mathbb{R}^h \end{aligned}$$

where i refers to the i-th component and sign refers to the sign function defined as follows:

$$\operatorname{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

And using the indicator function I, we can calculate a mean directional accuracy (MDA) as follows:

$$\mathrm{MDA} = \frac{1}{h}\sum_{i=1}^{h} I\{AD(i) = FD(i)\}$$

$$\text{where for each i, } I\{AD = FD\} = \begin{cases} 1 & \text{correct direction} \\ 0 & \text{wrong direction} \end{cases}$$

which is exactly same as the F-score in confusion matrix.

Furthermore, we can assign arbitrary positives a and b values instead of assigning 1 and 0

for correct and incorrect prediction. We can interpret a as a reward and b as a penalty in this case [5]. Then, we would have a generalized mean directional accuracy (GMDA) as follows

$$\text{GMDA} = \frac{1}{h}\left[\sum_{i=1}^{h} a \cdot I\{AD(i) = FD(i)\} - \sum_{i=1}^{h} b \cdot I\{AD(i) \neq FD(i)\}\right]$$

and when $a = 1$ and $b = 0$, it is same as MDA.

Mean directional accuracy actually answers to the following question: When the actual value went up or down, how often did the model predict it correctly? However, we often want to know more than just a rate of correctly predicted direction. We would like to know specifically how good the model can predict in different situations. Here, we consider the four most simplest cases as each of them corresponds to a two-by-two confusion matrix.

- When the actual value went up, how often did the model predict it to go up?
- When the actual value went down, how often did the model predict it go down?
- When model predicted the value to go up, how often did it actually go up?
- When model predicted the value to go down, how often did it actually go down?

These are all valuable questions to answer. Researchers care some particular questions more than others according to the purpose and the usage of prediction model. To express the question, it is helpful to draw a confusion matrix as Table 2.2. below. We use the following notation

- Actual Up/Down: Up and Down
- Predict Up/Down: $\widetilde{\text{Up}}$ and $\widetilde{\text{Down}}$

Now we are ready to answer the questions using the notation in the table 2.2.. Using the commonly used terms in confusion matrix, we introduce more intuitive and straightforward names for an evaluation of the predicted values in terms of directional accuracy as follows.

- When the actual value went up, how often did model predict it to go up?

$$P\left(\widetilde{\text{Up}} \mid \text{Up}\right) = \frac{P\left(\text{Up} \cap \widetilde{\text{Up}}\right)}{P(\text{Up})} = \frac{\#\left[\text{Up} \cap \widetilde{\text{Up}}\right]}{\#\text{Up}}$$

  which is known as recall, sensitivity, or true positive rate. We call it as Mean Positive Directional Rate (PDR)

	Predict Up	Predict Down	
Actual Up	$\widetilde{\text{UU}} = \#\left[\text{Up} \cap \widetilde{\text{Up}}\right]$	$\widetilde{\text{UD}} = \#\left[\text{Up} \cap \widetilde{\text{Down}}\right]$	$\widetilde{\text{UU}}$+$\widetilde{\text{UD}}$= Up
Actual Down	$\widetilde{\text{DU}} = \#\left[\text{Down} \cap \widetilde{\text{Up}}\right]$	$\widetilde{\text{DD}} = \#\left[\text{Down} \cap \widetilde{\text{Down}}\right]$	DU+$\widetilde{\text{DD}}$= Down
	$\widetilde{\text{UU}}$+$\widetilde{\text{DU}} = \widetilde{\text{Up}}$	UD+$\widetilde{\text{DD}} = \widetilde{\text{Down}}$	

Divide by $N = \widetilde{\text{UU}} + \widetilde{\text{DU}} + \text{UD} + \widetilde{\text{DD}}$ to get

	Predict Up	Predict Down	
Actual Up	$P\left(\text{Up} \cap \widetilde{\text{Up}}\right)$	$P\left(\text{Up} \cap \widetilde{\text{Down}}\right)$	$P(\text{Up})$
Actual Down	$P\left(\text{Down} \cap \widetilde{\text{Up}}\right)$	$P\left(\text{Down} \cap \widetilde{\text{Down}}\right)$	$P(\text{Down})$
	$P\left(\widetilde{\text{Up}}\right)$	$P\left(\widetilde{\text{Down}}\right)$	

Table 2.2: Confusion Matrix for Directional Accuracy

- When model predicted the value to go up, how often did it actually go up?

$$P\left(\text{Up} \mid \widetilde{\text{Up}}\right) = \frac{P\left(\text{Up} \cap \widetilde{\text{Up}}\right)}{P(\widetilde{\text{Up}})} = \frac{\#\left[\text{Up} \cap \widetilde{\text{Up}}\right]}{\#\widetilde{\text{Up}}}$$

  which is known as precision or positive predicted value. We call it as Mean Positive Directional Precision (PDP)

- When the actual value went down, how often did model predict it go down?

$$P\left(\widetilde{\text{Down}} \mid \text{Down}\right) = \frac{P\left(\text{Down} \cap \widetilde{\text{Down}}\right)}{P(\text{Down})} = \frac{\#\left[\text{Down} \cap \widetilde{\text{Down}}\right]}{\#\text{Down}}$$

  which is known as selectivity, specificity or true negative rate. We call it as Mean Negative Directional Rate (NDR)

- When model predicted the value to go down, how often did it actually go down?

$$P\left(\text{Down} \mid \widetilde{\text{Down}}\right) = \frac{P\left(\text{Down} \cap \widetilde{\text{Down}}\right)}{P(\widetilde{\text{Down}})} = \frac{\#\left[\text{Down} \cap \widetilde{\text{Down}}\right]}{\#\widetilde{\text{Down}}}$$

  which is known as negative predictive value. We call it as Mean Negative Directional

Precision (NDP)

In terms of positive direction, high positive directional rate with low positive directional precision means the following: when the price goes up, most of the time it is correctly predicted but there are many cases where the price actually does not go up whenever it is predicted to go up. On the other hand, low positive directional rate with high positive directional precision indicates the following: when the price goes up, most of the time it is not wrongly predicted but there are many cases where the price actually goes up whenever it is predicted to go up. For negative direction, high negative directional rate with low negative directional precision indicates the following: when the price goes down, most of the time it is correctly predicted but there are many cases where the price actually does not go down whenever it is predicted to go down. As an opposite case with low negative directional rate with high negative directional precision, most of the time it is wrongly predicted when the price goes down, but there are many cases where the price actually goes down whenever it is predicted to go down.

To sum up, directional accuracy measures are meaningful in that they capture the directional error which was not seriously considered by traditional evaluation measures such as RMSE and its variants. At the same time, however, by looking at directional error alone with little thought on magnitude is as problematic as focusing on the size of error alone. This is mainly because investors eventually want to have large profits, which can only be realized with right direction with small size of error. It goes same to loss in that one error with large size would offset the accumulated profits at once. Therefore, in order to take into account both direction and size of error, we can simply modify the GMDA as introduced below.

2.7.3 Directional Score (MDS)

We propose a directional score that captures both the directional error and size of prediction error. In the previously defined GMDA, each constant a and b plays a role as reward and penalty when the direction was correct and wrong respectively. Instead of using user-defined value a and b, it is natural to replace them by using inverse of forecast error and forecast error itself as reward and penalty respectively. Using the previously defined error vector in equation (5), we replace a and b to get

$$\frac{1}{h}\sum_{i=1}^{h}\left[\frac{1}{\mid error_{t,h}(i)\mid}\cdot I\{AD(i)=FD(i)\}-\mid error_{t,h}(i)\mid\cdot I\{AD(i)\neq FD(i)\}\right]$$

To compare the different models, larger MDS means more accurately predicted values in terms of its direction and the size of error.

2.7.4 Simple Back Testing

Simple back testing is a natural extension of GMDA in a financial industry. Investors focus on money metric which it tells whether there is a profit or not rather than the forecast error. Unlike the mean directional score, we consider the price vector instead of error vector when implementing simple back testing. We penalize the incorrect prediction by the difference between prices at a time of buy and sell instead of the error vector defined in equation 5. We can write the most simplest back testing strategy that we buy and sell each time based on the predicted direction with initial capital of 1.

$$\frac{1}{h}\left[\sum_{i=1}^{h} \mid Y_{h,t} - Y_{h,t-1} \mid \cdot (I\{AD(i) = FD(i)\} - I\{AD(i) \neq FD(i)\})\right]$$

To compare the profitability of different models, higher value indicates that we earn more money from the predicted values.

2.7.5 Diebold-Mariano (DM) Test

During prediction tasks, we often face the situations where we want to evaluate and compare the predicted values from different models. For most cases, after choosing the best in-sample model that minimizes the specified loss in a training, we want to test whether the predicted values from the best model actually performs better than the predicted values from the other models in terms of evaluation metrics. DM test is developed to compare those predicted values.[15] Assume that we have the following two forecast vectors:

$$\hat{Y}^1_{h|t} \text{ and } \hat{Y}^2_{h|t}$$

where each forecast vector is from the different time series model. We define the forecast errors for each forecast vector as

$$error^1_{t,h} = Y_{h,t} - \hat{Y}^1_{h|t} \in \mathbb{R}^h$$
$$error^2_{t,h} = Y_{h,t} - \hat{Y}^2_{h|t} \in \mathbb{R}^h$$

which is same as in equation (5).

We are interested in the loss differential between the two forecasts as follows:

$$d_t = L\left(error^1_{t,h}\right) - L\left(error^2_{t,h}\right)$$

where some examples of loss function g are $L(x) = x^2$, $L(x) = \mid x \mid$, and $L(x) = \exp(\lambda x) -$

$1-\lambda x$ where λ is a positive constant. What we would ultimately like to test is the following hypothesis

$$H_0 : E[d_t] = 0 \qquad \forall t$$
$$H_1 : E[d_t] \neq 0$$

The null hypothesis indicates that there is no difference in the accuracy of the two predicted values. To test the null hypothesis for the predicted values with $h \geq 1$, Diebold-Mariano (1995) devised the following statistics

$$\text{DM} = \frac{\bar{d}}{\sqrt{\frac{\hat{\gamma}_d(0)+2\sum_{k=1}^{h-1}\hat{\gamma}_d(k)}{T}}}$$

where the auto-covariance function $\hat{\gamma}(k)$ is given by

$$\hat{\gamma}_d(k) = \frac{1}{T}\sum_{t=|k|+1}^{T}\left(d_t-\bar{d}\right)\left(d_{t-|k|}-\bar{d}\right)$$

There is a concern that the DM test can have the wrong size, rejecting the null hypothesis too often, depending on the degree of serial correlation among the forecast errors and the sample size T. [46] To overcome this issue, adjusted test statistics is suggested by Harvey, Leybourne, and Newbold (1997), which is

$$\text{HLN-DM} = \sqrt{\frac{T+1-2h+h(h-1)}{T}}\cdot\text{DM}$$

Modified statistic makes a bias correction to the DM test statistic and compare the corrected statistic with a student t distribution with $(T-1)$ degrees of freedom, rather than the standard normal. [21]

Chapter 3

High-frequency Data Features

Investors believe that high frequency data has higher level of predictability than low frequency data. One reason for this is because trades and quotes (hereafter, TAQ) data contains rich information about every trading activity represented by the best bid and ask prices and their sizes. Such tick data is resource intensive so that we expect to generate distinct features from what we can obtain using low frequency data (e.g., daily or monthly return). Although TAQ data is not as extensive as whole Limit Order Book (LOB) data, it is still rich enough to understand the certain level of market microstructure of price dynamics and to analyze its statistical characteristics to accomplish a prediction task. Unfortunately, previous researches [20, 38, 45] focusing on applying deep learning framework on time series prediction using moderately frequent data (e.g., 1 min) have not tried to exploit those features extracted from TAQ. In this chapter, we will discuss a set of features extracted from TAQ data as follows:

- Basic Feature
- Market-Microstructure Feature
- Statistical Feature

We hope these set of features are helpful for capturing complicated patterns underneath the data and extract information that well summarizes trading activities happened within a time interval $[t, t+\Delta t)$ of our interest. To generate them, we use two types data set. One is matched data set where trades and quotes are matched according to Lee and Ready algorithm. The other one type is trades and quotes data itself without matching them together. In terms of the exchange (e.g., NYSE), we do not differentiate different exchanges for notational simplicity. Also, for the sake of simplicity, we consider the time interval of $[t, t+\Delta t)$ instead of i-th time interval $[t+(i-1)\cdot\Delta t\,,\, t+i\cdot\Delta t)$ where Δt refers to the sampled frequency that many existing features depend on.

3.1 Basic Feature

This feature set contains the most essential and basic information that we should include in the input.

3.1.1 Time Features

Time is the most basic feature for time series data. Typical time effects are induced from the cyclical property of data. It is well known and empirically proved that there exist opening time and closing time effect as well as Monday and Friday effect. We can capture such weekly and hourly effect by simply including the following time features. We characterize the time feature as follows:

- Hour: $9-16$
- Minute: $1-60$
- Week day (Mon - Fri): $1-5$
- Month (Jan - Dec): $1-12$
- Month day: $1-30\,(31)$
- Year day: $1-365$

3.1.2 Price Features

In general, we can not simply use entire trades and quotes data as an input while training the models due to its massive size and irregular time arrival. Instead of not using full data points, we sample them according to a certain time frequency of our interest such as 1 min or 5 min. For example, when we set our sampled frequency as 5 min $(=\Delta t)$, the most commonly used price features are the first, last, smallest, and highest one, which corresponds to open, close, low, and high price series also known as OHLC. For TAQ data, we can use either traded price series or quoted price series. Suppose we have the observed price series that are $P_t^1, P_t^2, \cdots, P_t^k$ and accordingly volume series that are $V_t^1, V_t^2, \cdots, V_t^k$ for a given time interval $[t, t+\Delta t)$. For the bid price (respectively ask price), we write it as

$$\text{Bid Price} = P_{BID,t}^1, P_{BID,t}^2, \cdots, P_{BID,t}^n \tag{3.1}$$

$$\text{Ask Price} = P_{ASKt}^1, P_{ASKt}^2, \cdots, P_{ASKt}^m \tag{3.2}$$

for a given time interval $[t, t+\Delta t)$.

- $\text{Open}_t = P_t^1$
- $\text{Close}_t = P_t^k$
- $\text{High}_t = \max\left\{P_t^1, P_t^2, ...P_t^k\right\}$
- $\text{Low}_t = \min\left\{P_t^1, P_t^2, ...P_t^k\right\}$
- Averaged Price $(\text{AP})_t = \frac{1}{k}\sum_{i=1}^{k} P_t^i$
- $\text{Return}_t = \frac{P_t^i - P_{t-1}^i}{P_{t-1}^i} \qquad i \in \{1, 2, \cdots, n\}$
- Activity Weighted Return $(\text{AWR})_t = \frac{AP_t - AP_{t-1}}{AP_{t-1}}$
- Volume Weighted Averaged Price (VWAP) = $\frac{\sum_{i=1}^{n} P_i \cdot V_i}{\sum_{i=1}^{n} V_i}$

We use the term "activity weighted" since we are in fact basing the weight on the frequency at which a particular price appears in trades. Unlike return that captures the change at a specific time point, the AWR captures the trading intensity. Specifically, for the consecutive time interval $[t, t+\Delta t)$ and $[t+\Delta t, t+2\Delta t)$, if the trading activity is centered around the same price point, then AWR could be similar to zero. Traditional return is heavily dependent on the time points at which the data is sampled. [47]

3.1.3 Volume Related Features

There are some other basic features that are essential for high-frequency data analysis. Suppose we have a series of observed bid size and ask size series as follows for a given time interval $[t, t+\Delta t)$:

$$BIDSIZE_t = \left\{V_{BID,t}^1, V_{BID,t}^2, \cdots, V_{BID,t}^n\right\} \tag{3.3}$$

$$ASKSIZE_t = \left\{V_{ASK,t}^1, V_{ASK,t}^2, \cdots, V_{ASK,t}^m\right\} \tag{3.4}$$

for a given time interval $[t, t+\Delta t)$.

- Number of Ticks: Number of ticks traded during the time interval. This can be used as one of the liquidity measures.

$$\begin{aligned}\text{Number of Ticks}_t &= \#\left(BIDSIZE_t\right) + \#\left(ASKSIZE_t\right) \\ &= n + m\end{aligned}$$

- BID and ASK Size: Bid size refers to the number of shares a buyer is willing to buy at the bid price. The ask size refers to the number of shares a seller is willing to sell at the ask price. We can interpret bid and ask size as representing the demand and supply relationship for certain stocks. If there exists a large discrepancy between bid and ask size, execution ceases until a liquidity provider steps into market.

$$\text{Bid Size}_t = \#(BIDSIZE) = n$$
$$\text{Ask Size}_t = \#(ASKSIZE) = m$$

- BID/ASK Imbalance: difference between Bid size and Ask size

$$n - m$$

- Number of Trades: Total number of trade executed during the time, which is volume.

$$\text{Number of Trades}_t = \#(Volume) = k$$

3.2 Market Microstructure Feature

Features in this section are related to specific theory in market microstructure field.

3.2.1 Indicator Feature

For every executed trading, there exist buyer and seller. We would like to know who initiates the deal in that trades cannot happen at the middle price between bid and ask in stock market, which divides every trade into two: either buy-side or sell-side. Buyers stand in the BID side and sellers stand in the ASK side as each of them wants to buy and sell at a lower and higher price respectively. To see from the seller's perspective, nothing happens until seller believes that the price will keep going down and wants to sell it no later than the price dropped more. By then, seller will sell at the price where buyers are waiting. This price is lower than the ASK price but should be higher than the price to the degree where the seller predicted to be dropped.

However, raw TAQ data does not contain such information so that we should employ trade direction algorithms to classify trades as being either buyer or seller motivated. Here we use the most well known algorithm called Lee and Ready (hereafter, LR). Simply, LR algorithm combines tick and quote by comparing the previous price and midpoint quotes respectively

[27]. Please refer to Lee and Ready (1990) paper for the detail. Using Lee & Ready algorithm, we obtain the directional indicator $D_t^1, D_t^2, \cdots, D_t^k$ where each indicates the following:

- $D_t = +1$: Buyer-initiated trade. So the traded size is classified as Buy amount
- $D_t = -1$: Seller-initiated trade. So the traded size is classified as Sell amount
- AR(1) coefficient of D_t: Serial correlation of the trade direction as follows:

$$D_t = \alpha D_{t-1} + \varepsilon_t$$

  where $\alpha = \frac{\mathbb{E}[D_t D_{t-1}]}{\text{Var}[D_{t-1}]}$ that minimizes the sum of squared errors. The high coefficient means that the buy order comes after buy order and sell order comes after sell order. Therefore, it becomes easier to predict whether the next order will be from sell-side or buy-side.

- Depth Imbalance:

$$\frac{D_t \cdot (\text{Ask Size}_t - \text{Bid Size}_t)}{\text{Ask Size}_t + \text{Bid Size}_t}$$

  where D_t denotes the indicator feature introduced. This is another measure of imbalance between ask and bid size.

3.2.2 Spread Feature

Bid-Ask spreads measure trade execution costs and reflect the price concessions necessary to complete transactions quickly. [6] Many investors view Bid-Ask spread as an important indicators of market quality.[7] When spread is high, cost of executing the trade at market price is high. When spread is low, the cost of executing the trade at market price is low. We set ASK_t and BID_t as the sampled value from Equation (6) and (7) where it usually refers to the last data point within the time interval $[t, t+\Delta t)$.

- Quotes Spread : $ASK_t - BID_t$
  - The difference between the best ASK price and the best BID price.
- Effective Spread : $2 \cdot (P_t - MID_t) \cdot D_t = 2 \cdot |P_t - MID_t|$
  - $MID_t = \frac{ASK_t + BID_t}{2}$
  - The effective spread is based on deviation between the execution price and the true underlying value of the security. [7] Since we do not observe the true price, we replace it with mid price. This is an estimate of the execution cost of trader and at the same time it is a benefit for the liquidity provider.

The spread defined above describes the differences at the specific time point which is sampled without reflecting the dynamics inside the time interval. To overcome this, Ye and Florescu [47] introduced activity weighted spread. For the BID prices and sizes (respectively ASK prices and sizes) in Equation (6) and (8) above, we consider the unique value of each series and label it as follows

$$BP_1, BP_2, \cdots, BP_h \tag{3.5}$$

where $h < n$ for a given time interval $[t, t+\Delta t)$. For a particular price BP_i, assume that we observe a sequence of consecutive sizes denoted as $\{BS_1, BS_2, \cdots BS_{h_i}\}$. Note that this sequence is for a particular price BP_i and please check examples in [47] for the details. Using this, we define the consolidated bid size for a specific price BP_i as follows

- Consolidated BID Size (CBS) : Within a time interval $[t, t+\Delta t)$, a series of consolidated BID size $(CBS_1, CBS_2, \cdots, CBS_h)$ is calculated as

$$CBS_i = BS_1 + \sum_{j=1}^{h_i} \mid BS_{j+1} - BS_j \mid, \qquad \text{for } i \in \{1, 2, ..., h\}$$

  The consolidated bid size cumulates the dynamics of the limit bid orders at the specific price BP_i so that we use this value as a weight factor to define the activity price. [47]

- Total Consolidated BID Size (TCBS)

$$TCBS_t = \sum_{i=1}^{n} CBS_i$$

- Weighted BID Price (WBP)

$$WBP_t = \sum_{i=1}^{h} BP_i \cdot \frac{CBS_i}{TCBS_t}$$

- Weighted ASK Price (WBP)

$$WAP_t = \sum_{i=1}^{h'} AP_i \cdot \frac{CAS_i}{TCAS_t}$$

  where n' denotes the number of observed ask prices within the interval $[t, t+\Delta t)$.

- Activity Weighted Spread (AWS)

$$AWS_t = WAP_t - WBP_t$$

It is not hard to see why this spread contains more information than traditional spread measures. Differences become evident when market exhibits demand pressure on one side (e.g., buy) while the other side (e.g., sell) orders keep pace but only by posting new offers at a relatively small sizes. In such cases, the WAP_t becomes lower than the WBP_t so that AWS_t can be negative as the expected bid based on activity is a higher value than the expected ask price. [47] Such activity weighted spread can be used to detect the anomalous situation by simply assuming that it follows a certain distribution (e.g., normally distributed) and concluded anomalous when it falls outside of the $1-\alpha\%$ confidence interval.

3.2.3 Size of Buy and Sell

The classification of trades is a fundamental information in market microstructure (hereafter, MM) as it helps to calculate the order imbalance, the price impact of large trades, and many other related issues in MM [27]. Specifically, ability to classify the order as buy or sell enables researches to build inventory model and information asymmetry model. Imbalance in buy-sell orders is used to measure the market response to an information event. [24] After matching the trades and quotes data using Lee and Ready algorithm, we know whether the volume of trade should be classified as buy size or sell size depending on the sign of indicator.

- Buy Size: Volume traded when the trading was buy-initiated with positive sign of indicator
- Sell Size: Volume traded when the trading was sell-initiated with negative sign of indicator
- Buy/Sell ratio:
$$r_{BS} = \frac{Buy\,Size}{Sell\,Size}$$
which shows the imbalance of buy and sell.
  - If the $r_{BS} > 1$, more buy orders were executed. Otherwise, more sell orders were executed.
- Buy/Sell Imbalance: the difference between buy size and sell size

3.2.4 MRR Features

In the market microstructure, researchers have tried to understand the the bid-ask spread and its components. Trade indicator model is one essential class of models and the idea is influenced

by 1) Roll (1998) model suggesting that prices contains information on the size of the spread and 2) Glosten and Milgrom (1985) model suggesting that suppliers of liquidity will adjust their bid and ask prices in response to the information content of trades they observe. [17] To estimate a trade indicator model, we need to have a series of executed prices as well as a trade indicator (direction) variable. Specifically, Madhavan, Richardson and Roomans (MRR) decomposes the variance of price change to the following four different parts:

$$Var(\Delta P_t) = \underbrace{\underbrace{2(1-\rho)\phi^2}_{\text{trading cost}} + \underbrace{(1-\rho^2)\theta^2}_{\text{info asymmetry}} + \underbrace{2(1-\rho^2)\phi\theta}_{\text{interaction}}}_{\text{Trading friction}} + \underbrace{\sigma_\varepsilon^2}_{\text{public news shock}}$$

where each component has its own meaning. The first three parts are from the trading friction inducing volatility change. The last part is the leftover that is not explained by trading activity so that we can interpret it as the volatility portion of news shocks. This is based on the price formulation equation

$$p_t = p_{t-1} + (\phi + \theta) \cdot D_t - (\phi + \rho\theta) \cdot D_{t-1} + \varepsilon_t$$

where the current price depends on the previous price, the cost of market maker (ϕ),level of market friction (θ), the degree of continuity of order flow (ρ) and the public information shock (ε). [28]. We used generalized method of moments (GMM) to estimate the coefficients. After getting the estimated coefficients, we can calculate how much public news shock takes portion for the price change volatility for every given time frequency such as 5 min. We believe that an information of changing portion of the public new shock and trading friction for every 5 min is valuable for prediction task.

3.2.5 Bid-Ask Bouncing Rate

Unlike daily data of closing price, TAQ data carries additional supply and demand information in the form of bid and ask prices. The bid-ask bounce refers to a situation that the price of a stock bounces between bid and ask. Bid-Ask bounce itself is often misunderstood as a perceived volatility because the price changed from bid to ask or vice versa. However, it is not actual oscillation because the range lies within the bid-ask spread, which means the mid price remains unchanged. In other words, not every executed tick changes the mid-price. Therefore, our matters of interest are

1. How many ticks were executed before the mid-price changed
2. How often the mid-price changed

For example, for a time interval $[t, t+\Delta t)$, we count the number of ticks executed before the mid-price changes. If every executed tick changes the mid-price, then the number of ticks executed before the mid-price change becomes 1. If the number of ticks executed before the mid-price change is 10, then it means that 9 executed ticks have bounced within the same bid-ask range. Therefore, frequently changing mid-price indicates that bid-ask bouncing rate is low.

This information is critical for market makers whose profits depend on the bid-ask bouncing rate. For a given time interval, suppose we have a series of values that refers to the number of ticks (NoT) executed before changing the mid-price that is $NoT_t = \{NoT_1, NoT_2, \cdots, NoT_n\}$. We are interested in using the following features related to bid-ask bouncing.

- Average number of ticks executed before changing the mid price

$$\frac{1}{n}\sum_{i=1}^{n} NoT_i$$

- Maximum number of ticks executed before changing the mid price

$$\max\{NoT_1, NoT_2, \cdots, NoT_n\}$$

- Mid-price change ratio

$$\frac{\#\{NoT_t\}}{\sum_{i=1}^{n} \mathrm{No}T_i}$$

3.3 Statistical Approach Features

3.3.1 Realized Volatility

By the formula, we know that realized volatility is squared root of realized variance and realized variance is a sum of squared intra-day log returns for a particular day. We learn from the elementary statistics course that the volatility characterized by variance or standard deviation from the underlying true distribution is not observable. On the other hand, realized volatility is empirically observable measurement of volatility over a specific time. This is simply because we use observed log stock price $X_{t_i} (= \log S_{t_i})$ to calculate the realized volatility. Now, you can simply think of this as another estimator for variance.

The notation X_{t_i} refers to i-th log price of an asset on $N+1$ regularly spaced times in the period of $[0,T]$ for the case of equidistance time interval. If we write $t_i = \Delta \cdot i, \quad i = 0,1,...,N$ and $\Delta = \frac{T}{N}$. We can think of a 24-trading-hours foreign exchange market where prices are

sampled every 30 minutes. The length of trading hours in one day can be normalized to $[0,1]$ and $[0,T]$ can be interpreted as length of trading hours for T days. When we set $T = 1$ and $N = 48$, this means we sample every 30 minutes for 24 hours on a particular day. If we define the X_{t_i}as a logarithm of price S_{t_i}, we can calculate the return as

$$\begin{aligned} R_{t_i} &= X_{t_i} - X_{t_{i-1}} \\ &= \log\left(\frac{S_{t_i}}{S_{t_{i-1}}}\right) \end{aligned}$$

When assuming that the log price follows the Geometric Brownian Motion, the quadratic variation of it over the interval $[0,T]$ is $\sigma^2 \cdot T$. We can use realized variance to get a consistent estimator $\hat{\sigma^2}$ as follows:

$$\begin{aligned} \hat{\sigma^2} &= \frac{1}{T}\sum_{i=1}^{N} R_{t_i}^2 \\ &= \frac{1}{T}\sum_{i=1}^{N} \left(X_{t_i} - X_{t_{i-1}}\right)^2 \end{aligned}$$

Such realized volatility exploits the fact that high frequency data will provide accurate estimate of variance of a process.

3.3.2 Two-Scales Realized Volatility

Two scales realized volatility (TSRV) estimator introduced by Zhang, Mykland, and Ait-Sahalia (2005) assumes that the price is measured with the noise from the high frequency data. The paper incorporates noise as follows:

$$Y_{t_i} = X_{t_i} + \varepsilon_{t_i}$$

where X_{t_i} is a true price and Y_{t_i} is an observed price with noise ε_{t_i} represented by market microstructure errors (e.g., spread and liquidity). The first-best estimator that combines the two time scales (all) and (avg), $\widehat{\langle X,X\rangle}_T$, is defined as [49]

$$\begin{aligned} \widehat{\langle X,X\rangle}_T &= [Y,Y]_T^{(avg)} - \frac{\bar{n}}{n}[Y,Y]_T^{(all)} \\ &= \frac{1}{K}\sum_{k=1}^{K}[Y,Y]_T^{(k)} - \frac{n-K+1}{n\cdot K}[Y,Y]_T^{(all)} \end{aligned}$$

where averaging across k number of subsamples. Other types of realized volatility estimators and their comparison can be found in [26].

3.3.3 Merton Jump Diffusion (MJD) Model

Jump diffusion model was designed to address the issue of fat tails, which is often observed in financial asset returns. Unlike the classical Black-Sholes model which assumes that the behavior of stock price is determined as

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t$$

where W_t is the Wiener process, μ and σ are constant draft and diffusion model, the MJD model assumes that the underlying process follows

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t + d\left(\sum_{j=0}^{N_t} (V_j - 1)\right)$$

The discontinuities of the price process are described by the Poisson process N_t with intensity parameter λ that controls the average number of jumps per unit time. The jump size is assumed to be log normally distributed with

$$\log V_j \sim N\left(\theta, \delta^2\right)$$

The set of parameters that we calibrate for this model is

$$\Theta = \{\mu, \sigma, \theta, \delta, \lambda\}$$

and we follow the [3] and set θ to be 0 meaning symmetric jumps.

3.3.4 Hawkes Process

We use hawkes process on the trades data to estimate the intensity of order arrival. For the sake of simplicity, we use exponential kernels and assume that the price moves by 1 tick only. We first consider the case where the buy orders and sell orders are independent of each other. This means we only care a self-exciting effect among the same type. Later, we can extend it to consider the cross-exciting process with bi-variate hawkes process model. This paper follows the notation from [40]. The point process is called Hawkes process if the conditional intensity

function $\lambda(t|H_t)$ takes the form:

$$\lambda(t|\mathscr{H}_t) = \lambda_0(t) + \sum_{i=t>T_i} \phi(t-T_i)$$

where $T_i < t$ are all event time having occurred before current time t, which contributes to the event intensity at time t. In other words, a counting process having random intensity function is hawkes process. As in the equation above, the intensity rate at time t is a random variable whose value is determined by $\mathscr{H}_t$. Furthermore, the following is the axiom of poisson process in hawkess process version:

$$\begin{aligned}
&\mathbb{P}(N(t+h) - N_t(t) = 1 \mid \mathscr{H}_t) = \lambda(t|H_t)\,h + o(h) \\
&\mathbb{P}(N(t+h) - N_t(t) \geq 2 \mid \mathscr{H}_t) = o(h)
\end{aligned}$$

Note that $\lambda(t|H_t) = \lambda(t \mid N(s), s \leq t)$. We can rephrase this in a more straight forward way as follows. For the first one,

$$\begin{aligned}
&\mathbb{P}(\text{an event in}[t,t+h] \mid \mathscr{H}_t) = \lambda(t \mid \mathscr{H}_t)h + o(h) \\
\Longleftrightarrow &\lim_{h\to 0} \frac{\mathbb{P}(\text{an event in}[t,t+h] \mid \mathscr{H}_t)}{h} = \lambda(t \mid \mathscr{H}_t)
\end{aligned}$$

For the second one,

$$\begin{aligned}
&\mathbb{P}(\text{two or more events in}[t,t+h] \mid \mathscr{H}_t) = o(h) \\
\Longleftrightarrow &\lim_{h\to 0} \frac{\mathbb{P}(\text{two or more events in}[t,t+h] \mid \mathscr{H}_t)}{h} = 0
\end{aligned}$$

Back to the conditional intensity function $\lambda(t|H_t)$, $\lambda_0(t) : \mathbb{R} \to \mathbb{R}_+$ is a deterministic base intensity function, and $\phi : \mathbb{R} \to \mathbb{R}_+$ is called the memory kernel. We observe that the hawkes process is a particular case of non-homogeneous Poisson process, in which the intensity is stochastic and explicitly depends on previous events through the kernel function. The reason why the hawkes process is called self-exciting is that every time an event occurs, the intensity increases.

3.4 Technical Indicators Feature

Classical technical indicators are going to be used in the experiment. All of them are calculated through the "TTR" package in R software.

RSI Relative Strength Index calculates a ratio of the recent upward price movements to the absolute price movements

BBands Bollinger Band is momentum indicator describing the two bands that are two standard deviation above and below a simple moving average.

Aroon Aroon is used to identify trend changes in the price of an asset as well as the strength of that trend. Up (Dn) trends are indicated when the aroonUp (Dn) is between 70 and 100. Strength is measured by the difference between the two.

SAR The parabolic SAR attempts to determine trend direction and potential reversals in price.

MOM Momentum measure the change or price, which can be used to capture a trend direction.

ChaikinAD Chaikin Accumulation Distribution is a volume-based indicator that measures the money flowing into or out of a security.

EMA Exponential Moving Average is a type of moving average that gives more weighting to recent price data.

ADX The average directional index is one of the directional movements indicators developed by Welles Wilder. IT measures the strength of the trend whether it be positive or negative direction.

MACD Moving Average Convergence Divergence is a trend following momentum indicator. If MACD goes up (down), then the stock price also goes up (down).

Chapter 4

Experiments

To find the best in-sample empirical loss minimizer as in (2.4), experiment is done using a set of selected function classes from the Table 2.1 on the NASDAQ100 index and TAQ data of IBM stock. For IBM TAQ data, different sampled time frequencies (e.g., 3, 5, 10, 15 Minutes) are used. We compare the performance of prediction for different function classes in terms of the selected evaluation measure. In this chapter, we provide the best experimental result for each model and data set. The detailed result table is provided in the Appendix B.

4.1 Datasets

We use the two datasets to test and compare the performance of different function classes in time series prediction.

NASDAQ100 This data set is first used in [38] and it contains price value of 81 major corporations listed in NASDAQ and the index value of NASDAQ100. Sampled frequency of this data set is one minute. The time series data covers the period from July 26, 2016 to December 22, 2016. The last 2,730 data points are used as the test set out of 40,560 data points.

TAQ This dataset contains the trades and quotes data of Trades and Quotes data set acquired from Wharton Research Data Service (WRDS). The time series data covers the period from January 01, 2020 to June 31, 2020, which includes the several extreme data points due to COVID-19. Among three well known technology companies listed in NYSE, we look into IBM whose number of observations are larger than DELL and smaller than MSFT so that it does not take too much time to clean and generate features. The number of data points for each month before cleaning the data is provided in the Table 4.1. The last 500 data points are used as the test set regardless of sampled frequency. We choose

	MSFT		IBM		DELL	
	Quotes	Trades	Quotes	Trades	Quotes	Trades
Jan	68.9M	4.1M	6.1M	1.0M	2.05M	382K
Feb	67.1M	7.4M	8.6M	1.1M	2.45M	448K
Mar	95.0M	**14.1M**	**21.1M**	**2.0M**	4.00M	**771K**
Apr	107.4M	8.3M	12.6M	1.3M	3.74M	416K
May	116.6M	5.7M	12.3M	0.9M	2.73M	366K
Jun	**132.4M**	6.6M	14.3M	1.2M	**5.20M**	627K

Table 4.1: Monthly TAQ Number of Observations in 2020

	Trades and Quotes Matched Data for IBM					
	Quotes	Trades	Quotes	Trades	Quotes	Trades
	Trading Hours		Same Time Stamps		Zero Values	
Jan	127K	23K	5.3M	762K	228K	0
Feb	74K	9K	8.0M	926K	136K	0
Mar	228K	21K	20.2M	1.7M	176K	0
Apr	106K	29K	11.9M	990K	118K	0
May	78K	10K	11.7M	716K	83K	0
Jun	93K	14K	13.5M	946K	174K	0

Table 4.2: Deleted Number of Observations to Match Trades and Quotes

4 sampled frequencies that are 3, 5, 10, and 15 minutes and the corresponding number of data points for IBM stock are

It is not surprising that for all three stocks in Table 4.1 shows the largest number of trades on March where the market crashes as COVID-19 spread unexpectedly fast globally and especially in New York at that time. Table 4.1 also shows that the number of Quotes are much larger than the number of Trades. This is why it takes so much time to clean the Quotes data and match it with Trades data to generate the related features introduced in Chapter 3. The way to clean the raw TAQ data is provided in the Appendix A. We modify and use the functions in R package called "highfrequency" to prepare the data for the experiment. The Table 4.3 shows the number of deleted trades and quotes data for IBM stock in the process of data cleaning.

4.1.1 Summary of Selected Feature Set

Using the TAQ data of IBM stock, we generate the 4 different types of feature set for different time frequencies as in Chapter 3. Those are basic, market microstructure, statistical and technical indicator features. For each of selected features, we provide the summary statistics including minimum, median, mean, maximum, and standard deviation (SD) as well as

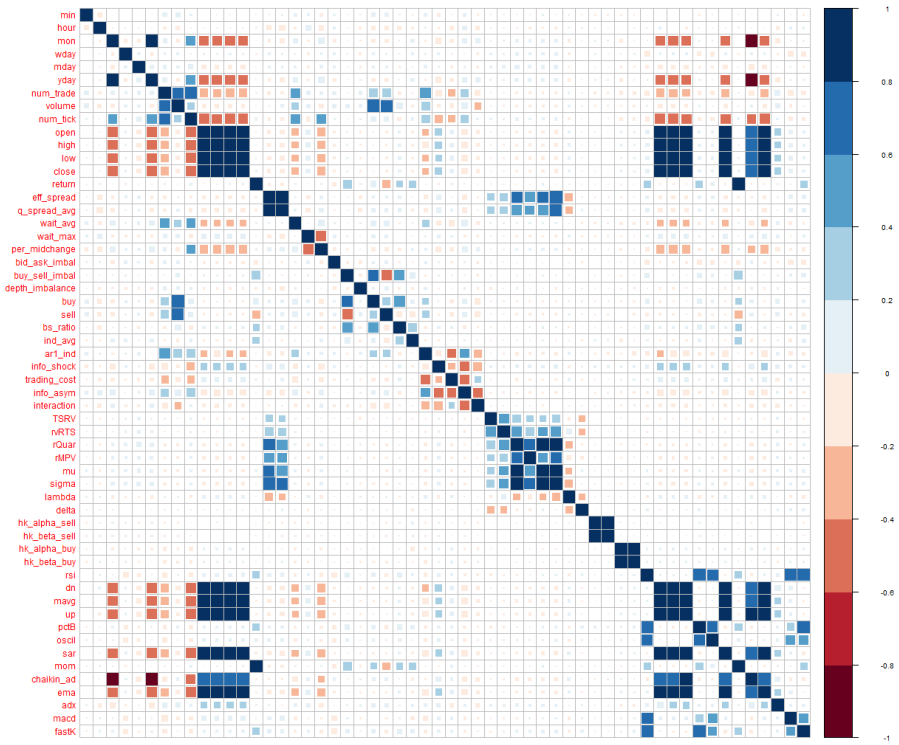

Figure 4.1: Correlation Plot of Features from IBM 5 Minute

the correlation plot as in Figure 4.1. From Table 4.3 to Table 4.6, we provide the summary statistics of selected features for IBM with sampled frequency of 5 minute from 2020.01.01 to 2020.06.31. For the IBM 5 min dataset, the whole number of data points are 9701 and the number of selected features is 56.

4.2 Deep Learning Architecture of Time-series Models

The basic building block of the deep learning architecture for time series forecasting is from the encoder-decoder framework. Useful relationships in data set is learnt through a series of non-linear layers to construct the intermediate feature representation. [4] In time series context, encoder plays a role as encoding the relevant historical information into a latent variable z_t and decoder use this to generate the predicted values.

$$\hat{y}_{t+1|t} = g_{dec}(z_t) = f(X_t, s) = f\left(\mathbf{x}_t^1, \mathbf{x}_t^2, \cdots, \mathbf{x}_t^{n-1}, \mathbf{y}_t\right)$$
$$z_t = g_{enc}(X_t, s)$$

where X_t is input values as in Equation (2.3), s is a static metadata associated with the entity, and g_{dec} and $g_{enc}$are encoder and decoder functions. [25] These are the basic building blocks

IBM 5min ($n = 9701$) from 2020.01.01 to 2020.06.31					
Feature	Min	Median	Mean	Max	SD
Minute	0	30	28.65	55	17.23
Hour	9	12	12.24	15	1.89
Month	1	4	3.539	6	1.71
Day in Weak	1	3	3.029	5	1.40
Day in Month	1	16	15.81	31	8.75
Day in Year	2	92	92.81	182	52.02
Number of Trade	39	159	165.7	300	46.68
Total Volume	7401	45238	66872	2.11M	82K
Number of Tick	44	285	273.7	300	28.54
Open	91.12	123.88	126.74	157.65	13.91
High	91.45	124.13	126.98	158.75	13.83
Low	90.59	123.71	126.5	157.24	13.99
Close	91.04	123.89	126.73	157.66	13.91
Return	-0.115	0	0	0.060	0.00

Table 4.3: Summary Statistics of Selected Basic Features

IBM 5min ($n = 9701$) from 2020.01.01 to 2020.06.31					
Feature	Min	Median	Mean	Max	SD
Eff Spread	0.016	0.123	0.208	6.384	0.311
Avg Spread	0.025	0.178	0.268	7.524	0.325
Lambda	0.937	0.993	0.992	0.997	0.005
Max Wait	2.000	4.000	3.953	22.000	1.156
% Mid Chang	0.473	0.779	0.775	0.931	0.050
Bid/Ask Imbal	-69621	145	201	655037	9820
Buy/Sell Imbal	-1.96M	-138	-72.9	1.76M	48645
Depth Imbal	-0.159	-0.004	-0.005	0.163	0.023
Avg Buy Size	24	139	180	8260	204
Avg Sell Size	20	141	180	9036	207
Buy/Sell Ratio	0.030	0.992	1.158	29.563	0.904
Avg D_t	-0.552	0.020	0.020	0.560	0.138
AR(1) D_t	-0.179	0.236	0.235	0.597	0.106
Info Shock	0.394	0.747	0.747	1.000	0.108
Trading Cost	0.000	0.031	0.047	0.555	0.052
Info Asym	0.000	0.082	0.106	0.835	0.096
Interaction	-0.757	0.060	0.050	0.205	0.068

Table 4.4: Summary Statistics of Selected Microstructure Features

IBM 5min ($n = 9701$) from 2020.01.01 to 2020.06.31					
Feature	Min	Median	Mean	Max	SD
rTSCov	-0.0872	0.00001	0.00015	0.16510	0.0042
rvRTS	-0.1385	0.00000	0.00064	0.54008	0.0148
rQuar	0.0000	0.00006	0.08034	16.1458	0.6169
rMPV	0.0000	0.00002	0.00203	0.42797	0.0150
μ	-0.0005	0.00000	0.00004	0.01957	0.0003
σ	0.0000	0.00001	0.00008	0.01062	0.0004
λ	-0.2548	0.00053	-0.00034	0.40644	0.0095
δ	-0.0998	0.00130	0.00129	0.11316	0.0044
α Sell	0.0000	0.00000	0.01244	39.0088	0.4463
β Sell	0.00250	1.55930	1.90180	1815.83	20.629
α Buy	0.00000	0.00000	0.00486	5.28578	0.0618
β Buy	0.00195	1.56037	1.60662	220.795	2.4179

Table 4.5: Summary Statistics of Selected Statistical Features

IBM 5min ($n = 9701$) from 2020.01.01 to 2020.06.31					
Feature	Min	Median	Mean	Max	SD
RSI	8.273	50.456	50.127	90.707	11.29
Bbands Dn	89.09	123.01	125.68	156.89	14.27
Mavg	92.35	123.76	126.75	157.11	13.89
Bbands up	94.09	125.19	127.82	158.74	13.57
ΔBbands	-0.582	0.513	0.501	1.584	0.33
Aroon Osc	-100	0	0.2824	100	56.50
SAR	90.59	124.05	126.76	158.75	13.91
MOM	-11.79	0	-0.0014	5.9346	0.45
Chaikin AD	-9.88M	-4.13M	-2.25M	6.95M	4.44M
EMA	92.18	123.75	126.75	157.11	13.88
ADX	5.576	23.103	24.569	63.508	10.56
MACD	-2.622	0.0072	-0.01	1.827	0.36
FastK	0	0.486	0.4984	1	0.32

Table 4.6: Summary Statistics of Selected Technical Features

of the deep learning architecture of time-series model of the following deep learning models for time series prediction.

The deep learning model employed in this paper is modified version of dual-stage attention-based recurrent neural network (DA-RNN). This model belong to the general class of non-linear Autoregressive Exogenous (NARX) models, which predicts the future value of a time series based on historical values of this series plus the historical values of multiple exogenous time series. [38] More importantly, two stages attention mechanism is under the encoder-decoder framework.

In encoder, g_{enc}, exogenous time series variables were used as input sequence and RNN encodes this price series into a feature representation with input attention mechanism where it helps adaptively select the relevant driving series. [38] Encoder is simply a non-linear activation function that could be any variant of RNN. In decoder, g_{dec}, we use another LSTM unit and attention layer to decode the encoded input information with temporal attention layers. The temporal attention layer decides the weights representing the temporal importance of each encoder hidden state for the prediction. Just like in encoder part, we choose LSTM unit as a non-linear activation function. The graphical illustration of the model is provided in the Figure 4.2. The number of look-back is selected from the [3,5,10,15,20]. The size of hidden states for the encoder and decoder is selected as 128 that achieve the best performance over the validation set.

4.3 Baseline Prediction Models

NAIVE is the most naive methods that uses the current value as a predicted value. In other words, naive forecasting method just repeats previous sample.

ARIMA is the auto-regressive integrated moving average model with regression terms on exogenous variables. Vector auto-regression model is an extension on multi-variable case. We choose the most simplest form which is AR(1). This is the most classical time series methods classified as global predictor in Table 2.1. Best parameter set is selected through re-estimation for every step.

PCA is the regression model with exogenous variables that are transformed based principal component analysis using factor loadings analogous to coefficients in a multiple linear regression. PCA loadings refer to the weights of the linear combination of each of input variables $\mathbf{x}_t^1, \mathbf{x}_t^2, \cdots, \mathbf{x}_t^{n-1}, \mathbf{y}_t$ as in Equation (2.3).The number of principal components is selected from [2,3,5,10,20] through cross validation. The best parameter set is updated as we re-estimate the model for every moving window to generate a sequence of

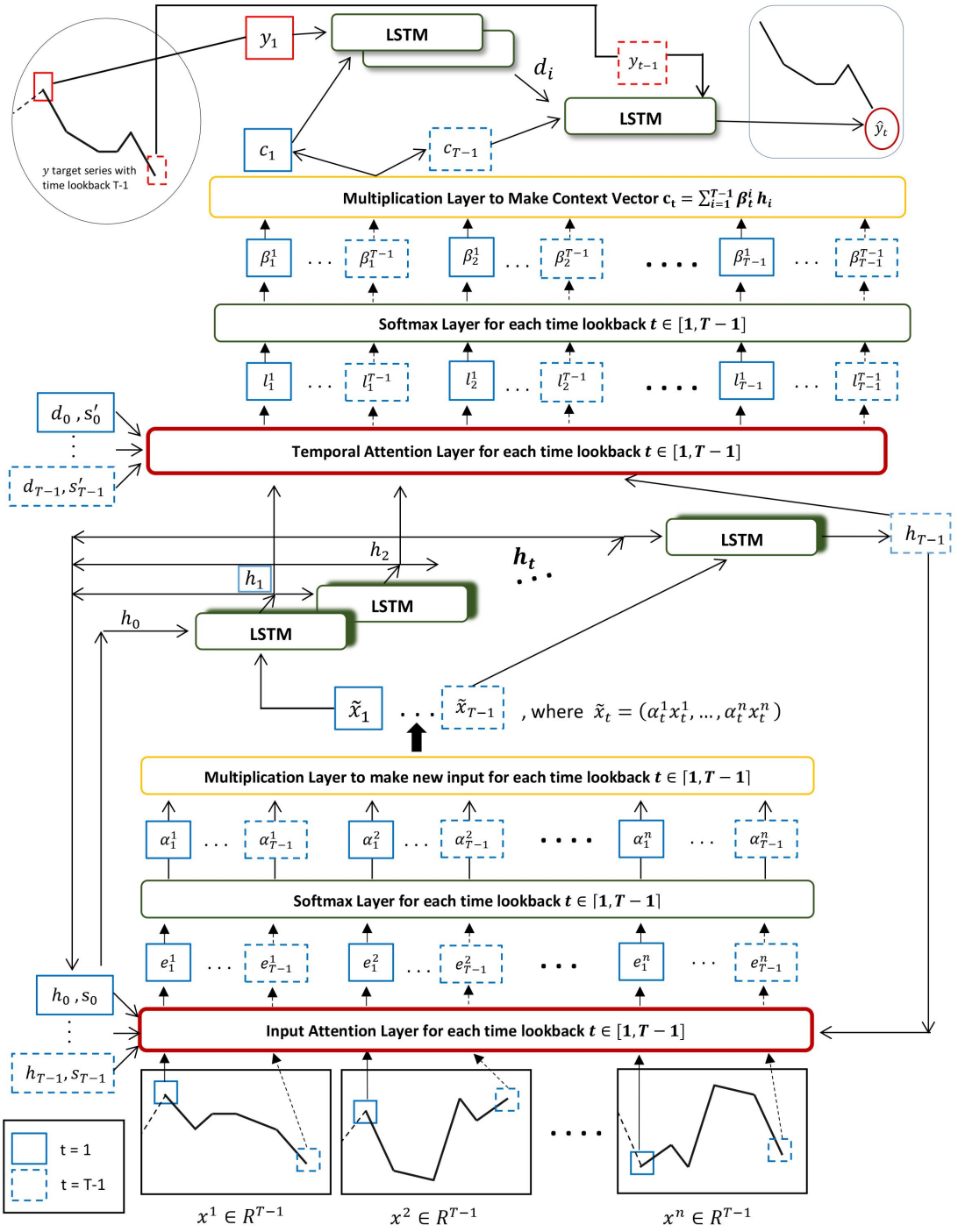

Figure 4.2: Graphical Illustration of DA-RNN Model

predicted values.

PLS is the partial least squared regression. It is similar to PCA in that both are used for dimension reduction and extract principal components. Unlike PCA that transforms the input X without considering Y, however, principle components in PLS focus on maximizing correlation between X and Y. The best parameter set is updated as we re-estimate the model for every moving window to generate a sequence of predicted values.

RR is the regularized regression model that contains ridge and lasso regression. We optimize the penalty parameter through grid searching within [0.005,0.01,0.05,0.1,0.5,1,5,10,20,50] for every estimation. The best parameter set is updated as we re-estimate the model for every moving window to generate a sequence of predicted values.

RF is the random forest regression. This is an ensemble model that uses bagging technique to average the predictions of each decision tree. It can be classified both local predictors and ensemble predictors in Table 2.1. In terms of hyper-parameter, we optimize the maximum number of levels in each decision tree, number of trees in the forest, the minimum number of data points allowed in a leaf node, and the maximum leaf nodes. Each of these is chosen from the range [5,20], [10,30], [1,10] and [5,15] respectively via grid search.

XGB is the extreme gradient boosting regression. This is another ensemble model based on decision trees, which produces a prediction model in the form of an ensemble of weak prediction models of decision trees. In terms of hyper-parameter, we optimize the number of estimator, minimum child weight, gamma, portion of subsample and maximum depth. Each of these is chosen from the range [30, 50], [1,5, 10, 50], [0.5,1,2,5], [0.6, 0.8, 1], [3,5] respectively via grid search.

4.4 Experimental Results and Discussion

To evaluate the performance of the proposed models along with baseline model, we use the following evaluation metrics of which most of them are introduced in Chapter 2: Root Mean Square (RMSE), Mean Directional Accuracy (MDA), Maximum Draw Down (MDD) Simple Back-testing Return, MPDP, MNDP, and Maximum Draw Down (MDD). The purpose of experiment is to compare the performance of different function classes with different time frequencies and different set of parameters. The first phase of the experiment uses the NAS-DAQ100 data using a set of selected function classe introduced above and the second phase

of the experiment uses the TAQ data of IBM stock with 4 different sampled frequencies that are 3, 5, 10, and 15 minutes. The chosen function classes from the Table 2.1 are ARIMA, Principle Component Regression, Lasso regression, Ridge regression, Partial Least Square regression, Regression Forest DA-RNN model. According to the different type of model and length of data, we choose the look-back parameter.

4.4.1 NASDAQ100 Data (n = 40560)

Table 4.7 shows the best selected look-back (LB) for each model along with the various evaluation metrics. The result shows that the best prediction model for NASDAQ100 data is PCA regression with the highest mean directional accuracy along with the profit although the AR model obtained the lowest RMSE with the lowest directional accuracy and profit. Note that for RFR, XGB and DA-RNN model, we did not re-train every time for the practical matter because it takes more than 1 minute to train the model. For other models, we retrain the model every moving window to update the model parameters. In terms of look-back, [50, 100, 500, 1000, 5000, 10000, 12000, 15000] are considered for the first five models. For RFR and XGB, we consider look-backs from [5000, 10000, 20000, 30000]. Look-back parameter of DA-RNN model is chosen from [3, 5, 10, 15, 20].

From the Table 4.7, It is notable that the PCA model renders over 92% directional accuracy followed by the 76% which is the second highest among the chosen function classes, whereas the RSME was 1.345, which is not the smallest number. The RMSE of DA-RNN is similar to the result reported in [45]. Moreover, since PCA is the best model, we compare the predicted values from it to other models using DM statistics whose null hypothesis states that there is no difference in the accuracy of the two predicted values. According to DM statistics in a 5% significance level, there is no difference between PCA model and the following models: DA-RNN, LR and RFR.

4.4.2 IBM TAQ Data

From Table 4.9 to 4.16, each table shows the best selected look-back (LB) for selected model and time frequency along with the various evaluation metrics and DM statistics. In terms of look-back for the sampled frequency of 3 minutes, [100, 500, 1000, 3000, 5000, 10000] is considered for the first five models. In terms of look-back for the sampled frequency of 5 minutes and 10 minutes, [100, 300, 500, 1000, 3000] is used for the first five models. In terms of the sampled frequency of 15 minutes, [100, 300, 500, 1000, 2000] is used. For RFR and XGB, we consider look-backs from [500, 1000, 2000, 3000] in case of sampled frequency of 3, 5, and 10 minutes. In terms of sampled frequency of 15 minutes, [500, 1000, 1500, 2000]

Model	LB	RMSE	MDA	MDD	Profit	PDR	NDR	PDP	NDP
AR	5000	**1.307**	0.656	-0.32	28.77	0.68	0.63	0.64	0.67
PCA	15000	1.345	**0.926**	**-0.04**	**58.49**	**0.93**	**0.92**	**0.92**	**0.93**
LR	5000	1.343	0.723	-0.19	38.88	0.88	0.57	0.66	0.83
RR	5000	1.308	0.741	-0.07	44.25	0.84	0.65	0.70	0.81
PLS	15000	1.356	0.718	-0.19	38.09	0.74	0.69	0.70	0.73
RFR	30000	1.365	0.767	-0.21	44.90	0.83	0.70	0.73	0.81
XGB	30000	1.362	0.764	-0.31	39.32	0.92	0.61	0.69	0.88
DA-RNN	15	1.412	0.749	-0.08	48.54	0.74	0.80	0.78	0.76

Table 4.7: The Best Selected Results of NASDAQ100 Index

DM Statistics on NASDAQ100	
Compared to PCA	P-value
AR	0.022
LR	**0.781**
RR	0.000
RFR	**0.064**
DA-RNN	**0.212**

Table 4.8: DM Statistics for Different Models Compared to PCA

is used. Look-back parameter of DA-RNN model is chosen from [2, 3, 5, 10, 15] for all time frequencies. Experimental results show that the RMSE are MDD are increasing with time frequency while the MDA remains almost same. In terms of RMSE, MDA, and Profit, PCA model shows the best result. However, DA-RNN would be considered to be optimal with the smallest value of maximum draw down in that investors want to manage the risk while market fluctuate a lot. Especially during the time when the market price fluctuates rapidly, one big loss would be critical. It is notable to see that MDD of DA-RNN is one third of most of the other models.

4.4.2.1 3 minutes (n = 16190)

Model	LB	RMSE	MDA	MDD	Profit	PDR	NDR	PDP	NDP
AR	3000	0.135	0.62	-0.34	20.13	0.65	0.59	0.61	0.63
PCA	3000	**0.129**	**0.92**	-0.57	**49.12**	**0.92**	**0.93**	**0.93**	**0.92**
LR	3000	0.135	0.77	-0.57	36.73	0.66	0.88	0.84	0.73
RR	3000	0.132	0.73	-0.57	35.35	0.70	0.77	0.74	0.72
PLS	3000	0.136	0.59	-0.57	14.88	0.57	0.61	0.59	0.59
RFR	500	0.141	0.54	-0.59	6.48	0.44	0.54	0.54	0.64
XGB	2000	0.134	0.55	-0.31	10.73	0.18	0.90	0.65	0.53
DA-RNN	10	0.146	0.72	**-0.19**	35.70	0.70	0.75	0.74	0.71

Table 4.9: The Best Selected Results of IBM 3 Minute

DM Statistics on IBM 3 Minute	
Compared to PCA	P-value
AR	0.000
LR	0.000
RR	**0.082**
RFR	0.000
DA-RNN	**0.203**

Table 4.10: DM Statistics for Different Models Compared to PCA

4.4.2.2 IBM 5 minutes (n = 9701)

Model	LB	RMSE	MDA	MDD	Profit	PDR	NDR	PDP	NDP
AR	300	0.219	0.56	-0.87	12.99	0.49	0.64	0.59	0.55
PCA	2000	**0.199**	**0.89**	-0.64	**76.47**	**0.88**	**0.90**	**0.90**	**0.88**
LR	3000	0.218	0.79	-0.64	63.38	0.73	0.85	0.84	0.76
RR	300	0.217	0.54	-1.42	9.60	0.40	0.68	0.57	0.52
PLS	2000	0.220	0.59	-0.67	17.67	0.59	0.60	0.60	0.59
RFR	500	0.206	0.64	-1.41	25.17	0.61	0.67	0.66	0.62
XGB	1000	0.225	0.57	-1.41	17.63	0.20	0.95	0.82	0.54
DA-RNN	10	0.228	0.70	**-0.24**	50.16	0.62	0.78	0.74	0.68

Table 4.11: Best Selected Results of IBM 5 Minute

DM Statistics on IBM 5 Minute	
Compared to PCA	P-value
AR	0.000
LR	0.005
RR	0.000
RFR	**0.349**
DA-RNN	**0.341**

Table 4.12: DM Statistics for Different Models Compared to PCA

4.4.2.3 IBM 10 minutes (n = 4832)

Model	LB	RMSE	MDA	MDD	Profit	PDR	NDR	PDP	NDP
AR	500	0.409	0.68	-1.03	76.55	0.65	0.70	0.70	0.66
PCA	1000	**0.392**	**0.89**	-0.69	**164.28**	**0.87**	**0.91**	**0.91**	**0.87**
LR	1000	0.405	0.76	-1.16	111.40	0.75	0.78	0.78	0.74
RR	1000	0.400	0.78	-0.98	137.05	0.79	0.76	0.78	0.78
PLS	500	0.406	0.66	-3.71	66.48	0.61	0.70	0.68	0.63
RFR	500	0.426	0.61	-3.70	42.88	0.64	0.58	0.62	0.60
XGB	1000	0.432	0.49	-3.70	1.75	0.46	0.52	0.50	0.47
DA-RNN	5	0.448	0.76	**-0.42**	115.8	0.69	0.82	0.80	0.73

Table 4.13: Best Selected Results of IBM 10 Minute

DM Statistics on IBM 10 Minute	
Compared to PCA	P-value
AR	**0.172**
LR	0.000
RR	**0.383**
RFR	0.000
DA-RNN	**0.657**

Table 4.14: DM Statistics for Different Models Compared to PCA

4.4.2.4 IBM 15 minutes (n = 3210)

Model	LB	RMSE	MDA	MDD	Profit	PDR	NDR	PDP	NDP
AR	500	0.625	0.63	-4.06	52.10	0.67	0.60	0.61	0.67
PCA	500	**0.560**	**0.90**	-3.22	**219.4**	**0.88**	**0.91**	**0.90**	**0.89**
LR	1000	0.611	0.74	-2.13	146.8	0.77	0.71	0.71	0.77
RR	500	0.599	0.74	-4.06	112.7	0.76	0.72	0.71	0.76
PLS	1000	0.621	0.65	-4.06	67.89	0.68	0.62	0.62	0.68
RFR	500	0.598	0.56	-1.37	65.3	0.63	0.50	0.54	0.59
XGB	500	0.594	0.54	-3.91	48.9	0.60	0.5	0.53	0.57
DA-RNN	2	0.689	0.73	**-0.53**	166.7	0.77	0.69	0.70	0.76

Table 4.15: Best Selected Results of IBM 15 Minute

DM Statistics on IBM 15 Minute	
Compared to PCA	P-value
AR	0.041
LR	0.016
RR	**0.125**
RFR	0.004
DA-RNN	**0.240**

Table 4.16: DM Statistics for Different Models Compared to PCA

Chapter 5

Conclusion & Future Work

In this paper, we explore machine learning approach to find the best in-sample empirical loss minimizer when conducting stock price prediction using a high-frequency data with certain sampled time frequencies (e.g., 1min, 5min, and 10min). Among many time series prediction models including both data model and algorithmic model, the following are selected and compared in terms of a set of proposed evaluation measures: ARIMA, PCA regression, Lasso and Ridge regression, Partial Least Squares regression, Random Forest regression, XGBoost regression, and Dual-stage Attention-based Recurrent Neural Network. The experiment focuses on one-step ahead forecasting with re-training the models for each moving window if available. If not available for practical reasons, we optimize the set of model parameters once and employ it to generate the sequence of predicted values in a rolling basis.

Unlike the complicated structured machine learning and deep learning model, simple data models can be re-estimated at every moving window now that re-estimation of those models can be done very quickly. Consequently, wherever applicable, re-estimation is highly recommended to improve the prediction performance especially for the non-stationary time series data whose underlying structures are time-varying. Another way to improve prediction is to introduce the look-back parameter to cut off the irrelevant long past historical data. In this setting, each model is optimized to select the most relevant time period of historical data. Compared to previous studies using NASDAQ100 [38, 45], the results demonstrate that re-estimation and properly chosen look-back parameter improve the prediction performance in terms of proposed evaluation measures.

For each time look-back, comparison of different models is done through a number of proposed evaluation metrics through which we were able to conclude that selected model with the smallest RMSE does not always have the highest directional accuracy and back testing result (i.e., profit) based on the results tables in Appendix B. Experiments with NASDAQ100 data, for example, show that the PCA regression with time look back of 15000 performs

the best in terms of all evaluation measures except RMSE. It is notable that the PCA model renders over 92% directional accuracy followed by a 76% directional accuracy by RFR, which is the second highest among the chosen function classes. At the same time, PCA's RMSE was 1.345, which was not the smallest number. Moreover, we can observe that the selected best look-back parameter changes over different types of function classes, which brings a large impact on prediction performance.

Experiments with TAQ data of IBM stock involves the calculation of a set of 56 selected features. Such feature set is believed to capture meaningful information that in turn generates profitable signals. We sampled the data every 3, 5, 10, and 15 minutes and extract the features from the TAQ raw data downloaded from WRDS. For each sampled time frequency, we evaluate the prediction performance and confirm that the RMSE and MDD are getting larger with each increasing time frequency increment, while the MDA remains similar. It is remarkable to see that the MDA remains around 90 percent for PCA throughout all the sampled frequencies and also over 70 percent for other models such as DA-RNN and Lasso Regression. Moreover, in terms of maximum draw down, DA-RNN is the best model for all sampled frequencies. Having smallest MDD is critical for risk management, especially when it comes to the year of 2020 when COVID-19 started to spread unexpectedly fast throughout the whole world.

In the future, we intend to apply the current analysis on other stock data and different type of financial products (e.g., futures and options). While our empirical application focuses on one-step ahead prediction, the method can be also extended to long term multi-step ahead prediction. Regarding the parameter set for each model (e.g., attention weights on DA-RNN and regression coefficient for PCA regression), deeper analysis needs to be done later to have better interpretation on the experimental results as to variable importance or temporal importance.

References

[1] Cleo Anastassopoulou, Lucia Russo, Athanasios Tsakris, and Constantinos Siettos. Data-based analysis, modelling and forecasting of the covid-19 outbreak. *PLOS ONE*, 15(3):e0230405, 2020.

[2] Richards Anthony. The dynamics of institutional and individual trading. *The Journal of Financial and Quantitative Analysis*, 40(1):1–27, 2005.

[3] Stan Becker. A note on estimating the parameters of the diffusion-jump model of stock returns. *Journal of Financial and Quantitative Analysis*, 16(1):127–140, 1981.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new persepctives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[5] Christoph Bergmeir, Mauro Costantini, and Jose M. BenÃtez. On the usefulness of cross-validation for directional forecast evaluation. *Computational Statistics and Data Analysis*, 76:132–143, 2014.

[6] Hendrik Bessembinder and Kumar Venkataraman. *Bid-ask spreads: measuring trade execution costs in financial markets*. John Wiley & Sons, New York, 2010.

[7] Hendrik Bessembinder and Kumar Venkataraman. Bid-ask spreads: Measuring trade execution costs in financial markets. *Encyclopedia of Quantitative Finance*, Wiley:184–190, 2010.

[8] Oliver Blaskowitz and Helmut Herwartz. On economic evaluation of directional forescasts. *International Journal of Forecasting*, 27(4):1058–1065, 2011.

[9] Olivier Bousquet, Stephane Boucheron, and Gabor Lugosi. Introduction to statistical learning theory. *Lectures Notes in Artificial Intelligence*, 3176:169–207, 2004.

[10] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16:199–231, 2001.

[11] Danilo Bzdok, Naomi Altman, and Martin Krzywinski. Statistics versus machine learning. *Nature Methods*, 15:233–234, 2018.

[12] Rui Manuel Castro. Statistical learning theory. 2019. Lecture Notes, Eindhoven University of Technology.

[13] Maddison Christopher. Alphago, hamiltonian descent, and the computational challenges. 2019. Statistics Seminar, Stanford Univeristy.

[14] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley, New York, 2018.

[15] Francis X. Diebold. Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of diebold-mariano tests. *Journal of Business and Economic Statistics*, 33:1, 2015.

[16] Roberf F. Engle and Jeffrey R. Russell. *Analysis of high frequency data*. North Hollan, Amsterdam, 2004.

[17] Theissen Erik and Zehnder Lars Simon. Estimation of trading costs: Trade indicator models revisited. *Center for Financial Research 14-09*, Working Paper, 2014.

[18] Drew Fudenberg, Jon Kleinberg, Annie Lian, and Sendhil Mullainathan. Measuring the completness of theories. *PIER Working Paper*, No. 18-010, 2020.

[19] John M. Griffin, Jeffrey H. Harris, and Sellim Tapaloglu. The dynamics of institutional and individual trading. *The Journal of Finance*, 58:2285–2320, 2003.

[20] Tian Guo, Tao Lin, and Nino Antulov Fantulin. Exploring interpretable lstm neural networks over multi-variable data. *Proceedings of the 36th International Conference on Machine Learning*, ICML, 2019.

[21] David Harvey, Stephen Leybourne, and Paul Newbold. Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13:281–91, 1997.

[22] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[23] Vitaly Kuznetsov and Mehryar Mohri. Discrepancy-based theory and algorithms for forecasting non-stationary time series. *Annals of Mathematics and Artificial Intelligence*, 86(1):1–33, 2020.

[24] Charles M. C. Lee. Information dissemination and the small trader: An intraday analysis of the small trader response to announcements of corporate earnings and changes in dividend policy. Ph.D. dissertation, Cornell University, 1990.

[25] Bryan Lim and Stefan Zohren. Time series forecasting with deep laerning: A survey. *arXiv preprint arXiv:2004.13408*.

[26] Liy Y. Liu, Andrew J. Patton, and Kevin Sheppard. Does anything beat 5-minute rv? a comparison of realized measures across multiple asset classes. *Journal of Econometrics*, 187:293–311, 2015.

[27] Yang Cheng Lu and Yu Chen Wei. Classification of trade direction for an equity market with price limit and order match: evidence from the taiwan stock market. *Journal of Investment Management and Financial Innovations*, 6, 2009.

[28] Ananth Madhavan, Matthew Richardson, and Mark Roomans. Why do security prices change? a transaction-level analysis of nyse stocks. *Center for Financial Research 14-09*, 10:1035–1064, 1997.

[29] Spyros Makridakis and Michele Hibon. The m3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16:451–476, 2000.

[30] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[31] David McAllester. Information theory: The fundamental equations of deep learning. 2019. Lecture Notes, University of Chicago.

[32] R. David Mclean and Jeffrey Pontiff. Does academic research destroy stock return predicrtability. *Journal of Finance*, 71(1):5–32, 2016.

[33] Rodrigo F Mello and Moacir Antonelli Ponti. *Machine Learning: A Practical Approach on the Statistical Learning Theory*. Springer, New York, 2018.

[34] Mehyar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT Press. Second Edition, Cambridge, 2018.

[35] Sendhil Mullainathan. Applications of machine learning to the empirical sciences. 2019. Lecture Notes, University of Chicago.

[36] Sendhil Mullainathan and Jann Spiess. Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31:87–106, 2017.

[37] Sendhil Mullainathan and Jann Spiess. Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106, 2017.

[38] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI' 17: 2627–2633, 2017.

[39] Alexander Rakhlin and Karthik Sridharan. Statistical learning theory and sequential prediction. 2014. Lecture Notes, Massachusetts Institute of Technology.

[40] Marian-Andrei Rizoiu, Young Lee, Swapnil Mishra, and Lexing Xie. A tutorial on hawkes processes for events in social media, 2017.

[41] Yves-Laurent Kom Samo and Dieter Hendricks. What makes an asset useful? *arXiv:1806.08444*, 86(1):https://arxiv.org/abs/1806.08444, 2018.

[42] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, Cambridge, 2014.

[43] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[44] Athey Susan and Guido Imbens. Machine learning methods economists should know about, 2019.

[45] Yunzhe Tao, Lin Ma, Weizhong Zhang, Jian Liu, Wei Liu, and Qiang Du. Hierarchical attention-based recurrent highway networks for time series prediction. *arXiv preprint*, arXiv:1806.00685, 2018.

[46] Umberto Triacca. Time series analysis. 2012. Lecture Notes, University of Sant'Anna.

[47] Ziwen Ye and Ionut Florescu. Extracting information from the limit order book: New measures to evaluate equity data flow. *High Frequency*, 2(1):37–47, 2019.

[48] Hailing Zhang and Zhaoxia Pu. Beating the uncertainties: Ensemble forecasting and ensemble-based data assimilation in modern numerical weather prediction. *Advances in Meteorology*, 2010, 432160.

[49] Lan Zhang, Per A. Mykland, and Yacine Ait-Sahalia. A tail of two time scales: Determining integrated volatility with noisy high-frequency data. *J. Amer. Statist. Assoc.*, 100: 1394–1411, 2005.

Appendix A

Highfrequency Data Cleaning

For this appendix, we explain how to get and clean up the trades and quotes data from the WRDS database through R program so that the same input data can be reproducible. The existing 'highfrequency' package in R does not contain all the functions we need to clean and extract the feature sets. We need to modify some of them to accomplish the certain task of researchers interest. Most importantly, to begin with, one has to have an access to the WRDS database.

1. Download the specific stock of interest with all variables checked for both quotes and trades data. Each quote identifies the time, exchange, security, bid/ask volumes, bid/ask prices, NBBO indicator, and more. Each trade identifies the time, exchange, security, volume, price, sale condition and more. We choose all the possible variables provided in WRDS for both quotes and trades data.

2. If you choose multiple stocks, we need to first split the data by ticker symbol (i.e. T for AT&T and MSFT for Microsoft). Unless you use the strong server that can handle whole dataset at the same time, it is recommended to separate by ticker first because for the case of AT&T stock 4 months of quotes data itself is already 8 GB. In case of IBM, 4 months of quotes data takes up 3.5 GB approximately.

3. To use the built-in functions in highfrequency package in R, we need to first transform the data properly. To do that, we rename the column names of the quotes and trades data respectively. The appropriate column names for the Quotes are "DATE", "TIME", "EX", "SYMBOL", "SYM_SUFFIX", "BID", "BIDSIZ", "OFR", "OFRSIZ", "MODE". The appropriate column names for the Trades are "SYMBOL", "DATE", "TIME", "PRICE", "SIZE", "G127", "CORR", "COND", "EX". We can now use the built-in function called *convert* in highfrequency package to convert saved csv file to xts timeseries object in

R. We have to convert trades and quotes data separately. One thing to notice is that we should use *%OS* to preserve the time stamp recorded as microseconds which is upto 6 digits.

4. Once we complete this, we can easily upload the data by using built-in function called *TAQLoad* in highfrequency package. We have to check whether the time index was sorted properly or not beforehand. After we load the TAQ data into R, we are ready to clean it. Among many built-in functions inside of package, we are going to use few of them that are necessary for analysis to extract the feature set of our interest.

   (a) Eliminate all the observations out of the trading hours which is from 9:30 to 4:00.

   (b) Get rid of observations whose price, quote, and bid are zero.

   (c) Replace multiple price and quote entries that have same time stamp by a single one whose the price and quotes come with maximum volume. This is to make the data set have unique time stamps only. One thing to notice is that if we use the built-in function for old version of the package, you have to manually change some codes. Otherwise, it will automatically merge same time stamp upto seconds not miliseconds so that we lose lots of data points. In the latest version of the code, it works properly. For additional tip, code will run much faster if we manually comment out the function that checks whether it is proper quotes and trades data that takes up time.

5. Now we are ready to match the trades and quotes data on the same unique time stamps. A key purpose of matching those two is to get an indicator variable that indicates whether it was buy-side order or sell-side order.

Appendix B

Additional Result

The more detail information about the experiment results are provided here. There are five data sets that are NASDAQ100, IBM sampled at 3, 5, 10, and 15 minutes.

B.1 NASDAQ100

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
AR	50	4.581	0.515	5.68	0.51	0.52	0.51	0.52	-0.25
	100	3.399	0.514	4.88	0.51	0.52	0.50	0.52	-0.21
	500	1.387	0.580	15.81	0.58	0.58	0.57	0.59	-0.20
	1000	**1.290**	0.604	22.69	0.59	0.61	0.60	0.61	**-0.16**
	5000	1.307	**0.656**	**28.77**	**0.68**	0.63	**0.64**	**0.67**	-0.32
	10000	1.334	0.650	28.09	0.65	**0.65**	**0.64**	0.66	-0.21
	12000	1.383	0.627	21.49	0.62	0.64	0.62	0.63	-0.32
	15000	1.371	0.621	21.38	0.62	0.62	0.61	0.63	-0.32

Table B.1: Results of AR on NASDAQ100

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PCA	50	**1.119**	0.811	38.83	0.81	0.81	0.81	0.82	-0.21
	100	1.142	0.837	43.59	0.84	0.83	0.83	0.84	-0.21
	500	1.155	0.842	50.49	0.83	0.86	0.85	0.84	-0.21
	1000	1.166	0.869	54.18	0.86	0.88	0.87	0.87	-0.07
	5000	1.298	0.860	55.05	0.91	0.81	0.82	0.90	-0.07
	10000	1.331	0.905	55.76	0.91	0.91	0.90	0.91	-0.11
	12000	1.342	0.875	50.71	0.89	0.86	0.86	0.89	-0.20
	15000	1.345	**0.926**	**58.49**	**0.93**	**0.92**	**0.92**	**0.93**	**-0.04**

Table B.2: Results of PCA Regression on NASDAQ100

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
LR	50	**1.305**	0.596	18.54	0.59	0.60	0.59	0.60	-0.20
	100	1.311	0.602	20.34	0.59	**0.62**	0.60	0.61	**-0.16**
	500	1.358	0.614	21.10	0.71	0.52	0.59	0.65	-0.21
	1000	1.355	0.606	20.80	0.78	0.44	0.57	0.68	-0.32
	5000	1.343	**0.723**	**38.88**	0.88	0.57	**0.66**	0.83	-0.19
	10000	1.391	0.595	17.33	0.86	0.34	0.56	0.72	-0.22
	12000	1.411	0.526	7.55	0.94	0.12	0.51	0.70	-0.20
	15000	1.414	0.498	0.28	**0.99**	0.02	0.49	**0.86**	-0.22

Table B.3: Results of Lasso Regression on NASDAQ100

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RR	50	1.279	0.613	20.52	0.61	0.61	0.60	0.62	-0.21
	100	1.259	0.653	27.48	0.66	**0.65**	0.64	0.66	-0.21
	500	**1.247**	0.681	34.12	0.73	0.63	0.66	0.71	-0.32
	1000	1.253	0.694	37.26	0.77	0.63	0.66	0.74	-0.32
	5000	1.308	**0.741**	**44.25**	**0.84**	**0.65**	**0.70**	**0.81**	**-0.07**
	10000	1.358	0.662	28.86	0.74	0.59	0.63	0.70	-0.21
	12000	1.377	0.624	21.24	0.68	0.57	0.60	0.65	-0.21
	15000	1.372	0.636	24.02	0.66	0.61	0.62	0.65	-0.21

Table B.4: Results of Ridge Regression on NASDAQ100

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PLS	50	1.486	0.567	12.50	0.55	0.58	0.55	0.57	-0.24
	100	1.397	0.574	11.02	0.56	0.57	0.56	0.58	-0.248
	500	1.277	0.620	22.45	0.62	0.61	0.60	0.63	-0.21
	1000	**1.243**	0.650	28.64	0.67	0.62	0.63	0.66	-0.31
	5000	1.294	0.685	34.57	0.74	0.63	0.65	0.71	-0.31
	10000	1.348	0.679	33.73	0.74	0.61	0.65	0.71	**-0.16**
	12000	1.360	0.702	34.53	**0.76**	0.63	0.67	**0.74**	-0.19
	15000	1.356	**0.718**	**38.09**	0.74	**0.69**	**0.70**	0.73	-0.19

Table B.5: Results of PLS Regression on NASDAQ100

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RFR	5000	**1.348**	0.721	30.83	0.72	**0.71**	0.71	0.73	-0.31
	10000	1.358	0.679	33.40	**0.92**	0.44	0.61	**0.85**	**-0.18**
	20000	1.357	0.732	39.15	0.80	0.65	0.69	0.78	-0.31
	30000	1.365	**0.767**	**44.90**	0.83	0.70	**0.73**	0.81	-0.21
XGB	5000	1.370	0.594	13.55	0.80	0.39	0.56	0.67	**-0.21**
	10000	**1.383**	0.640	20.59	0.87	0.41	0.59	0.77	**-0.21**
	20000	1.360	0.741	**40.26**	0.91	0.57	0.67	**0.88**	-0.31
	30000	1.362	**0.764**	39.32	**0.92**	**0.61**	**0.69**	**0.88**	-0.31

Table B.6: Results of RFR and XGB Regression on NASDAQ100

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
DA RNN	3	1.469	0.736	47.33	0.60	**0.86**	**0.80**	0.69	**-0.07**
	5	1.436	0.758	48.21	0.67	0.84	**0.80**	0.72	**-0.07**
	10	1.443	0.765	47.97	**0.81**	0.72	0.73	**0.79**	**-0.07**
	15	**1.412**	0.749	**48.54**	0.74	0.80	0.78	0.76	-0.08
	20	1.417	**0.774**	47.49	0.72	0.81	0.79	0.75	-0.08

Table B.7: Results of DA-RNN Regression on NASDAQ100

B.2 IBM 3 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
AR	100	764.5	0.50	-2.60	0.50	0.49	0.49	0.50	-0.57
	500	0.261	0.46	-5.72	0.38	0.53	0.45	0.47	-0.59
	1000	0.139	0.53	7.33	0.45	**0.61**	0.53	0.53	-0.35
	3000	**0.135**	**0.62**	20.13	0.65	0.59	**0.61**	0.63	**-0.34**
	5000	0.137	**0.62**	**21.09**	**0.67**	0.57	**0.61**	**0.64**	-0.57
	10000	0.144	0.46	-1.71	0.46	0.46	0.45	0.47	-0.57

Table B.8: Results of AR on IBM 3 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
LR	100	0.173	0.51	3.71	0.59	0.42	0.51	0.50	-0.59
	500	0.140	0.55	5.71	0.34	0.74	0.57	0.54	-0.59
	1000	0.138	0.58	12.11	0.21	**0.92**	0.71	0.55	-0.57
	3000	**0.135**	**0.77**	**36.73**	**0.66**	0.88	**0.84**	**0.73**	**-0.57**
	5000	0.138	0.73	31.61	0.65	0.81	0.76	0.70	-0.57
	10000	0.149	0.23	-27.09	0.17	0.28	0.19	0.26	-0.59

Table B.9: Results of LR on IBM 3 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RR	100	0.675	0.49	0.00	0.52	0.46	0.49	0.49	-0.59
	500	0.148	0.49	-0.16	0.38	0.60	0.48	0.50	**-0.57**
	1000	0.136	0.61	16.94	0.45	**0.78**	0.66	0.59	**-0.57**
	3000	**0.132**	**0.73**	**35.35**	**0.70**	0.77	**0.74**	**0.72**	**-0.57**
	5000	0.136	0.66	29.41	0.64	0.68	0.66	0.66	**-0.57**
	10000	0.141	0.51	3.43	0.52	0.49	0.50	0.51	**-0.57**

Table B.10: Results of RR on IBM 3 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PLS	100	770.5	0.48	-2.94	0.52	0.44	0.48	0.48	-0.59
	500	0.218	0.46	-4.21	0.42	0.49	0.45	0.46	**-0.57**
	1000	0.142	0.52	5.49	0.43	**0.61**	0.52	0.53	**-0.57**
	3000	**0.136**	**0.59**	**14.88**	**0.57**	**0.61**	**0.59**	**0.59**	**-0.57**
	5000	0.139	0.54	8.82	0.55	0.53	0.54	0.54	**-0.57**
	10000	0.143	0.52	4.63	0.51	0.53	0.51	0.53	**-0.57**

Table B.11: Results of PLS on IBM 3 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PCA	100	0.137	0.77	26.95	0.78	0.76	0.76	0.78	-0.59
	500	0.132	0.77	27.89	0.71	0.83	0.81	0.75	-0.59
	1000	0.131	0.83	38.39	0.76	0.91	0.89	0.79	**-0.57**
	3000	**0.129**	**0.92**	**49.12**	**0.92**	**0.93**	**0.93**	**0.92**	**-0.57**
	5000	0.132	0.91	48.71	0.91	0.91	0.91	0.91	**-0.57**
	10000	0.141	0.62	8.71	0.65	0.60	0.61	0.63	-0.59

Table B.12: Results of PCA on IBM 3 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RFR	500	**0.141**	**0.54**	**6.48**	**0.44**	0.54	**0.54**	0.64	-0.59
	1000	0.144	0.50	5.50	0.34	0.65	0.48	**0.65**	**-0.57**
	1500	0.147	0.48	-1.65	0.11	**0.83**	0.40	0.49	**-0.57**
	2000	0.151	0.44	-5.27	0.20	0.66	0.37	0.46	**-0.57**
XGB	500	0.142	0.49	-2.78	0.15	0.82	0.44	0.50	-0.59
	1000	0.153	0.51	2.40	0.05	0.94	0.48	0.51	-0.57
	1500	0.144	0.50	0.47	0.03	**0.95**	0.40	0.50	-0.57
	2000	**0.134**	**0.55**	**10.73**	**0.18**	0.90	**0.65**	**0.53**	**-0.31**

Table B.13: Results of RFR and XGB on IBM 3 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
DA RNN	2	0.146	0.73	45.45	0.71	0.76	0.76	0.71	-0.35
	3	0.154	**0.74**	**46.70**	0.69	**0.79**	**0.78**	**0.79**	-0.30
	5	0.147	**0.74**	39.67	**0.72**	0.77	0.76	0.73	-0.33
	10	0.146	0.72	35.70	0.70	0.75	0.74	0.71	**-0.19**
	15	**0.144**	0.72	32.72	0.65	0.80	0.77	0.69	-0.28

Table B.14: Results of LR on IBM 3 Minute

B.3 IBM 5 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
AR	100	404.7	0.48	-4.66	0.43	0.54	0.49	0.48	-1.42
	300	**0.219**	**0.56**	**12.99**	0.49	**0.64**	**0.59**	**0.55**	-0.87
	500	0.248	0.57	11.50	**0.51**	0.62	**0.59**	**0.55**	-1.42
	1000	0.221	**0.56**	10.04	**0.51**	0.61	0.58	**0.55**	**-0.68**
	2000	0.228	0.53	-1.78	**0.51**	0.56	0.54	0.52	-1.42
	3000	0.226	0.51	-4.91	0.50	0.52	0.52	0.51	-0.82

Table B.15: Results of AR on IBM 5 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PCA	100	0.205	0.69	23.41	0.69	0.69	0.70	0.68	-0.68
	300	0.192	0.73	42.53	0.72	0.74	0.74	0.72	-0.64
	500	**0.175**	0.85	72.42	0.81	0.89	0.89	0.82	**-0.23**
	1000	0.198	0.81	60.53	0.68	**0.95**	**0.93**	0.74	-0.35
	2000	0.199	**0.89**	**76.47**	**0.88**	0.90	0.90	**0.88**	-0.64
	3000	0.205	**0.89**	76.26	0.87	0.91	0.91	0.87	-0.64

Table B.16: Results of PCA on IBM 5 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
LR	100	0.342	0.44	-9.17	0.45	0.42	0.45	0.43	-1.42
	300	0.237	0.54	15.07	0.39	0.69	0.56	0.53	-0.67
	500	**0.204**	0.62	30.53	0.38	**0.86**	0.73	0.58	-0.67
	1000	0.217	0.55	18.79	0.28	0.83	0.63	0.53	**-0.64**
	2000	0.207	0.78	61.08	**0.73**	0.82	0.81	0.75	**-0.64**
	3000	0.218	**0.79**	**63.38**	**0.73**	0.85	0.84	**0.76**	**-0.64**

Table B.17: Results of LR on IBM 5 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RR	100	0.288	0.51	-3.72	**0.50**	0.51	0.51	0.50	-1.42
	300	**0.217**	0.54	**9.60**	0.40	**0.68**	**0.57**	**0.52**	-1.42
	500	0.229	0.53	7.71	0.39	**0.68**	0.56	0.52	-1.42
	1000	0.226	0.50	-0.60	0.37	0.64	0.52	0.50	-1.42
	2000	0.227	0.46	-7.13	0.46	0.46	0.47	0.46	**-0.87**
	3000	0.231	0.39	-20.00	0.34	0.44	0.38	0.39	**-0.87**

Table B.18: Results of RR on IBM 5 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PLS	100	183.9	0.48	-2.67	0.51	0.45	0.49	0.47	-1.42
	300	0.241	0.54	3.64	0.47	0.61	0.56	0.53	-1.42
	500	0.229	0.57	15.00	0.51	**0.64**	0.59	0.56	-1.42
	1000	0.227	0.58	13.34	0.52	**0.64**	**0.60**	0.57	-1.42
	2000	**0.220**	**0.59**	**17.67**	**0.59**	0.60	**0.60**	**0.59**	**-0.67**
	3000	0.231	0.55	8.12	0.53	0.56	0.55	0.54	**-0.67**

Table B.19: Results of PLS on IBM 5 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
DA RNN	2	0.229	**0.73**	**66.9**	**0.68**	0.77	**0.75**	**0.71**	-0.32
	3	0.232	0.71	61.8	0.63	0.78	0.73	0.69	-0.30
	5	0.229	0.71	59.08	0.65	0.77	0.74	0.69	-0.30
	10	**0.228**	0.70	50.16	0.62	0.78	0.74	0.68	**-0.24**
	15	0.235	0.69	47.76	0.60	**0.79**	0.73	0.67	-0.32

Table B.20: Results of LR on IBM 5 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RFR	500	**0.206**	**0.64**	**25.17**	0.61	0.67	**0.66**	**0.62**	-1.41
	1000	0.228	0.48	-1.57	0.45	0.52	0.49	0.47	-1.41
	2000	0.207	0.50	7.28	0.13	**0.87**	0.51	0.49	**-0.67**
	3000	0.216	0.49	-3.9	**0.68**	0.28	0.5	0.47	-0.86
XGB	500	0.225	**0.57**	**17.63**	0.20	0.95	**0.82**	**0.54**	-1.41
	1000	0.227	0.50	4.91	0.02	**0.97**	0.58	0.49	**-0.69**
	2000	**0.220**	0.50	-0.01	**0.95**	0.03	0.50	0.4	-0.86
	3000	0.221	0.49	2.54	0.08	0.89	0.46	0.49	-1.41

Table B.21: Results of RFR and XGB on IBM 5 Minute

B.4 IBM 10 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PCA	100	0.461	0.78	91.82	0.76	0.79	0.79	0.76	-1.32
	300	0.377	0.85	142.55	0.80	0.90	0.89	0.81	-0.66
	500	0.392	0.82	129.99	0.74	**0.90**	**0.89**	0.77	**-1.91**
	1000	**0.392**	**0.89**	**164.28**	**0.87**	0.91	0.91	**0.87**	-0.69
	2000	0.401	0.80	110.49	0.76	0.84	0.83	0.77	-1.12
	3000	0.455	0.47	-25.12	0.50	0.44	0.48	0.45	-3.71

Table B.22: Results of PCA on IBM 10 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
LR	100	0.696	0.53	14.42	0.46	0.61	0.55	0.52	-1.91
	300	0.408	0.61	55.60	0.35	**0.89**	0.77	0.57	-1.32
	500	0.411	0.67	74.85	0.49	0.86	**0.78**	0.62	-1.91
	1000	**0.405**	**0.76**	**111.40**	**0.75**	0.78	**0.78**	**0.74**	**-1.16**
	2000	0.413	0.63	48.73	0.66	0.59	0.63	0.63	-1.39
	3000	0.488	0.24	-50.30	0.23	0.26	0.25	0.24	-3.71

Table B.23: Results of LR on IBM 10 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RR	100	0.489	0.52	18.45	0.47	0.58	0.54	0.51	-1.91
	300	0.399	0.65	81.60	0.53	0.77	0.71	0.61	-3.71
	500	**0.395**	0.73	113.31	0.64	**0.83**	**0.80**	0.68	-3.71
	1000	0.400	**0.78**	**137.05**	**0.79**	0.76	0.78	**0.78**	**-0.98**
	2000	0.425	0.70	57.35	0.74	0.65	0.69	0.71	-3.71
	3000	0.444	0.61	18.51	0.62	0.60	0.62	0.60	-3.71

Table B.24: Results of RR on IBM 10 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PLS	100	3847	0.56	16.71	0.56	0.55	0.57	0.54	-1.91
	300	2395	0.57	45.16	0.50	0.65	0.60	0.55	-3.71
	500	**0.406**	**0.66**	66.48	**0.61**	**0.70**	**0.68**	**0.63**	-3.71
	1000	0.409	0.62	**73.80**	**0.61**	0.63	0.64	0.61	-1.64
	2000	0.419	0.56	26.25	0.60	0.51	0.57	0.55	**-1.39**
	3000	0.466	0.58	11.54	**0.61**	0.54	0.58	0.56	-3.71

Table B.25: Results of PLS on IBM 10 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
DA RNN	2	0.462	0.755	143.1	0.71	0.79	0.76	0.74	-0.52
	3	0.465	0.733	126.7	0.70	0.76	0.74	0.72	-0.45
	5	**0.448**	**0.761**	115.8	0.69	**0.82**	**0.80**	0.73	**-0.42**
	10	0.450	0.741	116.3	0.69	0.79	0.79	0.70	-0.43
	15	0.448	0.741	121.3	0.69	0.78	0.77	0.71	-0.42

Table B.26: Results of DARNN on IBM 10 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RFR	500	**0.426**	**0.61**	**42.88**	**0.64**	**0.58**	**0.62**	**0.60**	**-3.70**
	1000	0.444	0.49	0.19	0.51	0.47	0.50	0.48	**-3.70**
	2000	0.473	0.28	-40.06	0.26	0.29	0.28	0.27	**-3.70**
	3000	0.517	0.48	-16.19	0.83	0.10	0.49	0.38	**-3.70**
XGB	500	0.435	0.51	-0.54	**0.99**	0	0.51	0	**-1.91**
	1000	**0.432**	0.49	**1.75**	0.46	**0.52**	0.50	0.47	-3.70
	2000	0.442	**0.52**	-0.85	0.77	0.26	**0.52**	**0.52**	-3.70
	3000	0.471	0.49	-12.68	0.87	0.09	0.50	0.4	-3.70

Table B.27: Results of RFR and XGB on IBM 10 Minute

B.5 IBM 15 Min (n = 3210)

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
AR	100	13902	0.52	-8.18	0.54	0.51	0.50	0.54	**-4.06**
	300	0.652	0.61	45.91	0.60	**0.62**	0.60	0.63	**-4.06**
	500	**0.625**	**0.63**	52.10	**0.67**	0.60	**0.61**	**0.67**	**-4.06**
	1000	0.627	0.61	**55.24**	0.59	**0.62**	0.59	0.62	**-4.06**
	1500	0.635	0.60	26.23	0.61	0.59	0.58	0.62	**-4.06**
	2000	0.672	0.49	-19.38	0.49	0.49	0.47	0.51	**-4.06**

Table B.28: Results of AR on IBM 15 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PCA	100	**0.505**	0.76	139.8	0.71	0.81	0.77	0.75	**-1.47**
	300	0.561	0.81	183.4	0.81	0.82	0.81	0.82	-1.47
	500	0.560	**0.90**	**219.4**	0.88	**0.91**	**0.90**	0.89	-3.22
	1000	0.594	0.88	217.5	**0.91**	0.86	0.85	**0.91**	-2.13
	1500	0.61	0.76	116.5	0.73	0.79	0.76	0.76	-3.92
	2000	0.642	0.51	-18.9	0.52	0.50	0.49	0.53	-3.92

Table B.29: Results of PCA on IBM 15 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
LR	100	0.627	0.56	32.0	0.50	0.60	0.54	0.57	-3.22
	300	0.594	0.61	66.4	0.49	**0.72**	0.62	0.61	-3.22
	500	**0.589**	0.70	114.0	0.70	0.70	0.68	0.71	-3.22
	1000	0.611	**0.74**	**146.8**	**0.77**	0.71	**0.71**	**0.77**	**-2.13**
	1500	0.624	0.57	32.4	0.73	0.43	0.54	0.63	-3.92
	2000	0.664	0.28	-50.2	0.23	0.33	0.24	0.31	-3.92

Table B.30: Results of LR on IBM 15 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RR	100	**0.586**	0.63	97.2	0.64	0.62	0.61	0.65	**-1.47**
	300	0.604	0.69	**115.6**	0.69	0.69	0.67	0.71	-4.06
	500	0.599	**0.74**	112.7	**0.76**	**0.72**	**0.71**	**0.76**	-4.06
	1000	0.622	0.71	85.1	0.74	0.68	0.68	0.74	-4.06
	1500	0.620	0.62	54.5	0.71	0.54	0.59	0.67	-4.06
	2000	0.650	0.54	-1.26	0.54	0.54	0.52	0.56	-4.06

Table B.31: Results of RR on IBM 15 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
PLS	100	9424	0.56	30.84	0.58	0.54	0.54	0.59	-4.06
	300	0.621	0.60	56.31	0.63	0.58	0.58	0.63	-3.92
	500	**0.602**	0.64	**71.05**	0.67	0.61	0.61	0.67	**-3.81**
	1000	0.621	**0.65**	67.89	**0.68**	**0.62**	**0.62**	**0.68**	-4.06
	1500	0.628	0.56	28.76	0.64	0.47	0.53	0.59	-4.06
	2000	0.679	0.46	-22.47	0.44	0.47	0.44	0.48	-4.06

Table B.32: Results of PLS on IBM 15 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
DA RNN	2	0.689	**0.73**	**166.7**	0.77	**0.69**	**0.70**	0.76	**-0.53**
	3	0.731	0.71	148.2	0.78	0.65	0.66	0.77	-0.71
	5	0.745	0.70	163.1	0.79	0.62	0.66	0.76	-0.51
	10	0.883	0.66	111.8	0.79	0.53	0.62	0.73	-1.42
	15	**0.686**	**0.73**	140.7	**0.79**	0.67	0.68	**0.79**	-0.54

Table B.33: Results of DARNN on IBM 15 Minute

	LB	RMSE	MDA	Profit	PDR	NDR	PDP	NDP	MDD
RFR	500	**0.598**	0.56	**65.3**	0.63	**0.50**	**0.54**	**0.59**	**-1.37**
	1000	0.606	0.51	25.0	0.74	0.29	0.49	0.55	**-1.37**
	1500	0.664	0.51	1.87	**0.81**	0.23	0.49	0.57	-3.91
	2000	0.697	**0.49**	1.82	0.63	0.36	0.48	0.52	-3.91
XGB	500	**0.594**	**0.54**	**48.9**	0.60	**0.5**	**0.53**	**0.57**	**-3.91**
	1000	0.631	0.47	-6.63	**0.99**	0	0.48	0	-4.05
	1500	0.635	0.50	-3.7	0.85	0.17	0.49	0.55	**-3.91**
	2000	0.631	0.53	7.42	0.6	0.47	0.51	0.56	**-3.91**

Table B.34: Results of RFR and XGB on IBM 15 Minute