Quantifying Heterogeneity in Financial Time Series for Improved Prediction

Yao, Shibo* sy372@njit.edu

Yu, Dantong dtyu@njit.edu

Abstract

In this paper, we apply multiple machine learning models to predict whether the machine learning enabled stock portfolio management outperforms the market benchmark. We first quantify the heterogeneity of time series data by using a Kullback-Leibler divergence approach to measure the similarity between different time points in financial markets. The different time points contain the historical data that demonstrate the same market pattern and stock distributions and constitute a "time regime" for which we train individual machine learning models. During the testing phase and the future market validation, we first evaluate its closest time regime and then choose the designated model in the same time regime for inference and evaluation. The new approach yields better performance than those based on the sliding window approach.

Keywords: Financial time series, data shift, heterogeneity.

1 Introduction

Prediction in finance has been difficult due to its stochastic nature, nonlinearity and heterogeneity. People deal with the first two problems using signal processing methods and nonlinear models, which have been proved effective along with their fast development and wide application in many fields including finance[8][10][2]. Heterogeneity describes the property that time series data have different statistical characteristics at different time points or within different time segments. Data shift in machine learning [17] is another name that is highly related with heterogeneity. This could lead to significant violations in the initial model assumptions and therefore weak ability of model generalization. One existing approach is sliding window[9] shown in figure 1 which assumes that adjacent time points are homogeneous. This corresponds to the fact in finance that many financial participants prefer to look at the most recent financial information[12] as they think the most recent information contains most value and, the market environment doesn't change much within a certain period.

*Presented at 6th Applied Financial Modeling Conference, New York

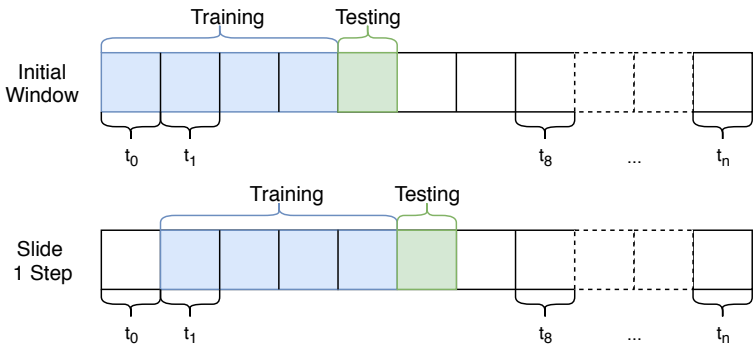

Figure 1: Sliding Window Approach

In this paper, we would like to contribute to the problem that given the financial reports of public-listed companies, how can we predict the stock price movement within the next period, to be more specific, whether the stocks can outperform the market benchmark or not. Several nonlinear machine learning models are applied for classification. We propose to use Kullback-Leibler divergence[11] to measure the similarity between time segment pairs, and find out the most similar historical time segments and use them as the training set given a new time segment as testing set. This is inspired by time series segmentation and clustering[6][9].

A subproblem of such prediction is to quantify the heterogeneity. To be more specific, given a testing set with a timestamp, choosing which portions of the historical data can help to quantify the heterogeneity in time and yield better performance in prediction? This corresponds to the problem in practice: given the current financial reports, if an investor wants to make predictions on the stock performances over the benchmark, which historical financial reports should he or she look at to build the model? Such concerns also have explanation in finance. People could have different behavior patterns (e.g. risk aversion-loving) in different market settings. Thus, it seems necessary to train different models under different market settings, where each model should be based on the set of most similar time series segments.

We address the problem of data shift in machine learning taking financial time series prediction as an example. Data shift has been considered a major obstacle in machine learning and data mining as it leads to violations of model assumptions on data distribution. There are three types of data shift, the prior shift, the covariate shift and the concept shift.

In this paper, we propose to use Kullback-Leibler divergence[11] to measure the similarity between time segment pairs, and find out the most similar historical time segments. During the training step, we create one model for each new time segment by training with the data from all of its similar time segments first. During the testing phase, we first identify the time segment and its associated model that was already pre-trained, and use it for inference.

Section 2 details the problem of data shift in machine learning and section 3 proposes a KL-divergence based similarity approach. We review the principles of Naive Bayes, Logistic Regression, Support Vector Machine (SVM), Random Forest and Adaptive Boosting Tree in section 4 and describe the experiments and data in Section 5. After that, we offer the results, conclusion and the future works.

2 Data Shift

Data shift is a class of problems in machine learning where the distribution of training, testing and real data varies significantly, which renders the trained model is irrelevant for testing and production deployment. It is already confirmed in [13][17] that direct inference by the model is problematic. Traditional statistical models usually make strict assumptions on data distribution, for example, the data should be independent and identically distributed (i.i.d.). Although machine learning models make less strict assumptions on data, the distribution can still significantly affect model performance. Data shift can be categorized into three types, Prior Probability Shift, Covariate Shift and Concept Shift.

2.1 Prior Probability Shift

Prior Probability Shift refers to the situation where the target variable distribution varies in training set and testing set, $p(y)_{train} \neq p(y)_{test}$. In the most simple case, binary classification, the target variable falls into a Bernoulli distribution. However, it is possible that the training set is balanced and the testing set is unbalanced, which means there are approximately equal number of positive instances and negative instances in training set yet in testing set there are much more of one class than the other, or the other way around. The model trained on a balanced set can have unrealistic performance given a new unbalanced testing set.

In this paper, we construct labels based on excess returns of stocks. Hence the data is always balanced and the prior distribution stays unchanged.

2.2 Covariate Shift

Covariate Shift refers to the situation where feature value distribution varies, $p(\vec{x}_{train}) \neq p(\vec{x}_{test})$. This is another common problem in machine learning. For example, peers from computer vision always need to take care of the illumination changes. Normalization and standardization can be applied to quantify covariate shift.

In this paper, we first convert feature values into ordinal and then apply normalization.

2.3 Concept Shift

Concept shift refers to the situation where the joint distribution of target variable and features vary in training set and testing set. $p(\vec{x}, y)_{train} \neq p(\vec{x}, y)_{test}$ This can be caused by environment changes, for example, a financial participator can have different opinions toward similar financial reports in bull and bear markets respectively. Prior shift and covariate shift are quantifiable over data transformation whereas concept shift is much more difficult to process.

The main idea of quantifying the concept shift is to find out the most similar historical data regarding the probabilistic distribution and proceed to learn with the retrieved historical data. Some people refer to this approach as "active learning" [3][15] that means actively selecting relevant data to improve the quality of a model. Here we use the Kullback-Leibler Divergence to measure the similarity between two distributions at two different time segments. The smaller divergence, the more similarity between them. We depict how to use the KL-Divergence to mitigate the concept drift in the following section.

3 The Kullback-Leibler Divergence Based Similarity

The KL-divergence measures how much one statistical distribution diverges from another one[11]. Therefore, the larger the divergence is, the less similarity there would be. In continuous cases, the KL-divergence is given by

$$D_{KL}(P||Q) = -\int_{-\infty}^{\infty} p(x) log \frac{p(x)}{q(x)} dx$$

where P and Q denote two probability distributions and $p(x)$ and $q(x)$ are their probability density functions. In information theory, $D_{KL}(P||Q)$ is called the KL-divergence from Q to P where P is the true probability distribution and Q is an approximated distribution. One good property of KL-divergence is that it is a one-way measurement, usually $D_{KL}(P||Q) \neq D_{KL}(Q||P)$, which complies with the property of time series. Thus, in our case P represents the distribution of historical time points for training and Q represents the new incoming time points for testing.

We cannot apply KL-divergence to measure the joint distribution distance between training set and testing set as the ground truth of testing would not be available in practice. Thus we propose to use market benchmark daily trading data before financial report publishing days to model the market similarity. Such market similarity can be an indicator of the joint distribution similarity. In other words, the financial participators' behaviour pattern toward financial reports can be influenced by the market environment. And such market environment is a strong reason for data shift. One model assumption is necessary, that is the 4-price daily returns of market benchmark fit multivariate-Gaussian,

$$f(x) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu)\right)$$

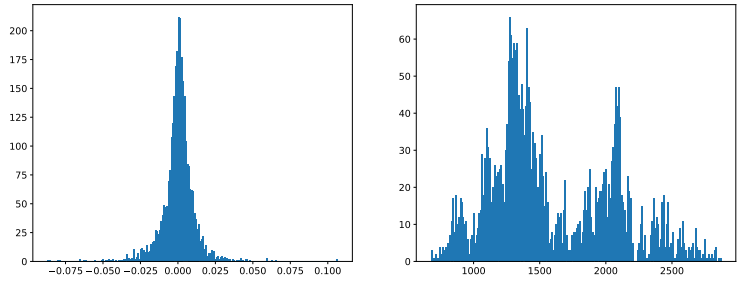

Figure 2: Benchmark Daily Return Distribution v.s. Benchmark Daily Price Distribution

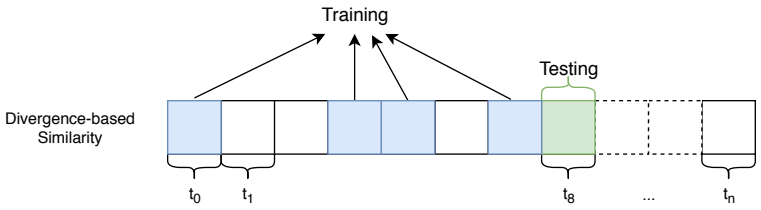

Figure 3: KL-divergence Similarity Approach

where $x \in R^4$ as the 4-price (open, high, low, close) data are used for modeling the similarity. The reason daily returns instead of original daily prices are employed for measuring market similarity is that return is more Gaussian-like as shown in figure 2.

Since we aim at training different models under different market settings, where each model should be based on the set of most similar time series segments, the objective function can be formed as

$$\begin{aligned} \arg\min_{J} \quad & \sum_{j \in J} D_{KL}(p_j(x) || p_t(x)) \\ \text{s.t.} \quad & j < t \qquad \forall j \in J, \\ & |J| \leq C, \\ & p_j \sim \mathcal{N}(\mu_j, \Sigma_j), \\ & p_t \sim \mathcal{N}(\mu_t, \Sigma_t) \end{aligned} \tag{1}$$

where $p_t(x)$ is the probability distribution at the testing timestamp, $p_j(x)$ denotes the timestamps where we would like to form a dataset for training, J is the result set, $|J|$ is the cardinality of the set and C is a hyperparameter that restrict the cardinality of the result set, the constraint $j < t$ enforces that only historical data can be used for training and prevents future information leak.

Given two mutivariate-Gaussian distributions with same dimensionality, $\mathcal{N}_0(\mu_0, \Sigma_0)$ and $\mathcal{N}_1(\mu_1, \Sigma_1)$, the KL-divergence from $\mathcal{N}_1$ to $\mathcal{N}_0$[4] is described by

$$D_{KL}(\mathcal{N}_0||\mathcal{N}_1) = \frac{1}{2}(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) - k + ln(\frac{|\Sigma_1|}{|\Sigma_0|})) \tag{2}$$

Since we would also like to compare the results given by KL-divergence similarity and sliding window, the sliding window can be embedded in the objective. Recall that sliding window always encourages the most recent adjacent time segments fall into the same training set. The exponential decay function

$$I(t_2) = I(t_1)e^{-\lambda(t_2 - t_1)}$$

can be used for describing this process. The mixed objective function is

$$\begin{aligned} \arg\min_{J} \quad & \sum_{j \in J} D_{KL}(p_j(x)||p_t(x)) - \beta e^{-\lambda(t-j)} \\ \text{s.t.} \quad & j < t \forall j \in J, \\ & |J| \leq C, \\ & p_j \sim \mathcal{N}(\mu_j, \Sigma_j), \\ & p_t \sim \mathcal{N}(\mu_t, \Sigma_t), \\ & \beta > 0, \\ & \lambda > 0 \end{aligned} \tag{3}$$

where λ controls the speed at which information decays, β controls the trade-off between KL-divergence similarity and sliding window. When β approaches very large, the objective function describes sliding window process; when β approaches zero, the objective describes KL-divergence similarity process; when β is in the middle, the objective describes a mixed approach. We can fix other hyperparameters and vary β to see how well the proposed method performs over sliding window.

4 Classification Models

The prediction is eventually a binary classification problem.

$$\vec{x} = (x_1, x_2, ..., x_n)^T \rightarrow Y \ ,$$

where $\vec{x}$ represents the financial factors that are represented by a scalar value and retrieved from financial reports, n is the number of the factors and Y is the label that denotes whether a specific stock outperforms the market benchmark, for example S&P index. We list some representative financial factors and their meaning in Table 1.

Table 1: Representative Financial Factors

Bloomberg Code	**Description**
EPS_GROWTH	earnings per share report-to-report growth
EARN_FOR_COMMON	net income available to common shareholders
HISTORICAL_MARKET_CAP	historical market capitalization
SALES_REV_TURN	sales-revenue turnover rate
CF_CASH_FROM_OPER	cashflow from operating activities
FNCL_LVRG	financial leverage
VOLATILITY_30D	30-day volatility

4.1 Naive Bayes

Based on Bayes Theorem and the assumption of conditional independence, Naive Bayes uses each feature as evidence to maximize the posterior probability. Bayes Theorem tells us that

$$p(A|B) \cdot p(B) = p(B|A) \cdot p(A)$$

, which can be written as

$$p(A|B) = \frac{p(B|A) \cdot p(B)}{p(A)}$$

. In the classification case, it can be written as

$$p(C_k|\vec{x}) = \frac{p(C_k) \cdot p(\vec{x}|C_k)}{p(\vec{x})}$$

where C_k is class label k, $\vec{x} = (x_1, x_2, ..., x_n)$ is the feature vector. We can apply the chain rule to expand this formula,

$$p(C_k|\vec{x}) = \frac{p(C_k) \cdot p(x_1, x_2, ..., x_n|C_k)}{p(\vec{x})}$$

Naive Bayes makes conditional independence assumption on class label and feature values, which means

$$p(x_1, x_2, ..., x_n|C_k) = \prod_{i=1}^{n} p(x_i|C_k)$$

Therefore

$$p(C_k|\vec{x}) = \frac{p(C_k) \cdot \prod_{i=1}^{n} p(x_i|C_k)}{p(\vec{x})}.$$

Since $p(\vec{x})$ is fixed given the input feature vector, the Naive Bayes method can be represented as

$$p(C_k|\vec{x}) \propto p(C_k) \cdot \prod_{i=1}^{n} p(x_i|C_k).$$

And we would like to maximize the posterior to construct the classifier,

$$\hat{y} = argmax_{k \in 1,2,...,K} p(C_k) \prod_{i=1}^{n} p(x_i|C_k).$$

In practice, Naive Bayes works quite well in predicting market trend.

4.2 Logistic Regression

Logistic Regression originates from linear regression and regresses on the logarithm of the odd being true or false. It uses the sigmoid function to convert the regression result to the probability of a binary label. The sigmoid function maps a real-number input to a value between zero and one.

$$\sigma(t) = \frac{1}{1 + e^{-1}}$$

In simple linear regression, we have the model assumption

$$y = \beta_0 + \beta_1 \cdot x_1 + ... + \beta_n \cdot x_n + \epsilon$$

. If we replace the t in the denominator part in the sigmoid function with a linear combination of different attributes, we have the form of logistic regression,

$$y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot x_1 + ... + \beta_n \cdot x_n)}}$$

.

4.3 Support Vector Machine (SVM)

The SVM is a classification method that obtains a hyperplane to separate the data points of different labels[7] with the largest margin. The tuning process is to maximize the hyperplane margin, and at the same time to have as few mis-classifications as possible. The objective function for SVM is described as follows:

$$\begin{aligned} \underset{w,\beta}{\arg\min} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & \xi_i \geq 0, \\ & y_i(x_i^T w + \beta) \geq 1 - \xi_i \ \forall i \end{aligned}$$

where w is the correlation coefficient vector, β is the intercept term, C is a hyper-parameter that controls the tradeoff between fitting the data and maximizing the margin, N denotes the number of samples for training, x_i is the feature vector, y_i is the corresponding label, ξ_i is the slack variable for constructing the soft margin.

To deal with nonlinear situations, kernel SVM can be employed. The trick is to find out a kernel that can map the data points from the original space into a high-dimensional feature space so that the inseparable cases become separable. The kernel employed in this report is (Gaussian) Radius Basis Function (RBF) kernel,

$$K(x, x^{'}) = exp(-\frac{||x - x^{'}||^2}{2\sigma^2}) \ ,$$

where x and $x^{'}$ are two feature vectors in the original space, $|| \cdot ||$ denotes l_2 norm, σ is a hyper parameter.

4.4 Random Forest

Random Forest[16] is an ensemble machine learning algorithm based on decision tree. The mechanism of decision tree classifier follows the top-down approach to separate the data points into different subsets until all data points in each subsets are correctly labeled or certain criteria is reached. Each node in the tree corresponds to a feature from the original data and each leaf at the bottom of the tree corresponds to a class. In a node m, representing a region R_m with N_m observations, let

$$p_{mk} = \frac{1}{N_m} \sum_{x_k \in R_m} I(y_i = k) \ ,$$

the portion of class k observations in node m. The observations in node m are classified to class

$$k(m) = argmax_k p_{mk} \ ,$$

the majority class in node m. In this report, Gini Impurity, $\sum_{k=1}^{K} p_{mk}(1-p_{mk})$, is used for growing the tree. Random Forest is such a classifier that consists of many trees where each tree is built on a randomly-selected portion of the training data. Given a new instance, the predicted label would be the majority vote of the trees. And since this report focuses on two-class problem, $K = 2$. The key of Random Forests is to ensemble multiple learners and vote for the final learning result. It has the advantage of minimal overfitting and excellent generalization capability in the new dataset.

4.5 Adaptive Boosting Tree

Boosting is an another ensemble model based on a series of weak learners and of the form $F_T(x) = \sum_{t=1}^{T} f_t(x)$, where F_T is the final classifier, f_t is the classifier built at the t step and T denotes there are T steps in total. The weak learners in Boosting Tree are shallow trees or leaves.

Adaptive Boosting Tree is a specific type of Boosting Tree method which has the form [5]

$$F_t(x) = F_{t-1}(x) + \alpha_t h_t(x),$$

where $F_t(x)$ is the classifier at t step, $F_{t-1}(x)$ is the classifier at $t-1$ step, $h_t(x)$ is the weak learner that minimizes ϵ_t, the weighted sum error of misclassifications

$$\epsilon_t = \sum_{i=1, h_t(x_i) \neq y_i}^{n} w_{i,t}$$

and

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t} \ .$$

The weight for each data point is updated

$$w_{i,t+1} = w_{i,t} e^{-y_i \alpha_i h_t(x_i)}$$

and renormalized in each iteration such that they sum up to 1. All these models are well developed and already implemented in Scikit-learn[14]. Adaptive Boosting Tree fits with the general practice of risk-aversion where investors identify those error-prone difficult instances and increase their weights so that these stances will get special attention during the next round optimization.

5 Experiment Settings

For classification, quarterly financial reports issued by 87 public-listed companies which are also S&P500 components are retrieved from Bloomberg terminal. The period spreads from 2012-08-01 to 2017-05-01. There are 83 financial factors used as features and they remain the same for all stocks and all timestamps. The stocks, features and period are decided jointly to avoid missing values.

The dataset is cross-sectional formatted. Let m be the number of stocks, T be the number of time points and n be the number of factors, we can use

$$x_{i,t,j} \quad i = 1, 2, ..., m; t = 1, 2, ..., T; j = 1, 2, ..., n$$

to represent the scalar value stored in the element i, t, j in the 3-dimensional array.

Data preprocessing steps are crucial to the subsequent machine learning algorithms. Firstly, we modify the extreme values in the factor vectors in the stock dimension. For a given factor at a certain time point, we denote $\widetilde{x_{i,t}}$ as the median value across all financial factors $(x_{i,t,1}, x_{i,t,2}, ..., x_{i,t,n})$. We normalize the financial factors by subtracting the median value from the original vector and attain a new median vector $(x_{i,t,1} - \widetilde{x_{i,t}}, x_{i,t,2} - \widetilde{x_{i,t}}, ..., x_{i,t,n} - \widetilde{x_{i,t}})$. The final preprocessing value depends on a hyper-parameter, named d. We use empirical method to get the optimal value of the hyper-parameter. We use the following equation to replace the extreme values:

$$x_{i,t,j}^{(1)} = \begin{cases} \widetilde{x_{i,t}} - d \cdot \widetilde{x_{i,t,j} - x_{i,t}}, & \text{if } x_{i,t,j} < \widetilde{x_{i,t}} - d \cdot \widetilde{x_{i,t,j} - x_{i,t}} \\ \widetilde{x_{i,t}} + d \cdot \widetilde{x_{i,t,j} - x_{i,t}}, & \text{if } x_{i,t,j} > \widetilde{x_{i,t}} + d \cdot \widetilde{x_{i,t,j} - x_{i,t}} \end{cases}$$

Secondly, we take out the market sector signal and market capitalization influences on the factors, by regressing factors against market sector dummy variables $S = (s_1, s_2, ..., s_k)^T, k = 9$, and logarithm of stock market values $log(v_{i,t})$. Taking logarithm of large-scale variables is a commonly-used transformation for rescaling financial data.

$$x^{(1)}_{i,t,j} = \beta_{0_i} + B_i^T \cdot S_i + \beta_{k+1_i} \cdot log(v_{i,t}) + \epsilon_{i,t,j}$$

where $B_i = (\beta_1, \beta_2, ..., \beta_k)^T$ is the coefficient vector of the dummy variables.

Now we take the residuals $\epsilon_{i,t,j}$ as the processed factor values to carry on.

$$x^{(2)}_{i,t,j} = \epsilon_{i,t,j}$$

Thirdly, the factor values at each time point are rescaled to a uniform distribution on $(0, 1]$ by reordering them in an increasing manner ($x^{(2)}_{[i],t,j}$, where $[i]$ is the new order) and dividing their orders by m which is the total number of stocks. This can help to quantify covariate shift in data.

$$x^{(3)}_{i,t,j} = \frac{[i]}{m}$$

As for constructing labels for the patterns, we first calculate the rate of return in such way:

$$r_{i,t} = \frac{\Delta P_{C_{i,t}}}{P_{Ci,t}}$$

where $r_{i,t}$ is the rate of return of stock i within in period t, $P_{Ci,t}$ is the closing price of stock i at the beginning of time period t, $\Delta P_{Ci,t}$ is the closing price change of stock i within time period t.

Then, we take the excess return $r^{'}_{i,t}$, which is the difference between stock price and mean return of selected stock pool,

$$r^{'}_{i,t} = r_{i,t} - \frac{1}{m}\sum_{i=1}^{m} r_{i,t}$$

and give each stock at each time point a label $L_{i,t}$, which is analogous to Y,

$$L(x_{i,t}) = \begin{cases} 1, & \text{if } r^{'}_{i,t} \geq 0 \\ 0, & \text{if } r^{'}_{i,t} < 0 \end{cases}$$

. Note that label $L(x_{i,t})$ corresponds to pattern vector $x_{i,t} = (x_{i,t,1}, x_{i,t,2}, ..., x_{i,t,n})^T$. By using the label based on excess return, we can quantify the prior shift in data.

As we use excess stock returns as target, this unavoidably enforces that the targets must be constructed within the same periods. In other words, the stock returns need to be compared with each other within the same periods for calculating the excess returns. Some flexibility in terms of target construction is sacrificed in exchange of solving prior shift. Therefore, we need to determine

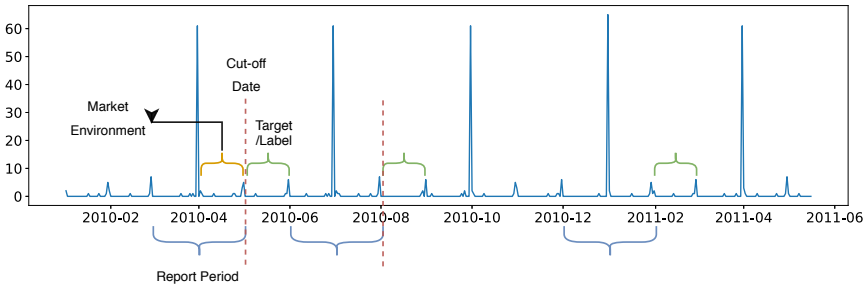

Figure 4: Financial Reports Density

some cut-off dates based on the density of financial reports. According to the data, most of the publicly listed companies disclose their financial reports four times a year, in December or January for the first one, in March or April for the second one, in June or July for the third one and in September or October for the forth one respectively. Thus, the targets are calculated based on the stock excess returns in May, in August, in November and in February correspondingly. Only in this way can we make sure the target labels are exactly in between of two report periods, which means the target variable is controlled by the prior financial report and not affected by future information.

For market similarity modeling, one month daily OHLC prices of market benchmark before financial report publishing days are used for estimating the mean vectors and covariance matrices of Gaussian distributions, where the price of market benchmark is the summation of all stocks being analyzed in the market. As shown in figure 4, the blue curve represents the density or the counts of financial reports published, the cut-off dates are the endings of report periods, the labels are constructed based on the one-month stock excess returns, and the market similarity is based on the one-month benchmark daily trading data.

Accuracy and back-test performance are used for measuring the goodness of the model. To be more specific, portfolios are constructed based on the prediction. The portfolios always buy in those stocks with an outperform label and hold the positions for 1 month. The back-test performance is presented in both simple cumulative return

$$r_{add} = \Sigma_{t=1}^{T} \frac{1}{S} \Sigma_{s=1}^{S} r_{s,t}$$

and compound cumulative return

$$r_{exp} = \prod_{t=1}^{T} \frac{1}{S} \Sigma_{s=1}^{S} (1 + r_{s,t}) - 1$$

, where S is the total number of stocks being back-tested and T is the back-test

time duration.

In financial investment problems, it is usually unnecessary to adopt the result on the whole testing set. Instead, people often look at those most reliable predictions and make decisions accordingly. By most reliable predictions, it usually means those predictions that come with the highest scores given by machine learning models, e.g. the instances that are furthest away from hyperplane in SVM and the instances with largest posterior in Naive Bayes. In this paper, we also apply post-selection on the predictions to keep those most reliable predictions and dispose of the unreliable ones. Consequently, lift curve is an appropriate performance visualization tool. The lift curve being presented in the paper is the average of the positive instances correctly labeled divided by the positive base rate and the negative instances correctly labeled divided by negative base rate given different threshold α. In here, α corresponds to the top proportion and the bottom proportion being selected. For example, $\alpha = 0.4$ means we select top 40% of the predictions as positives and the bottom 40% as negatives. We also look at the Area Under Lift Curve to compare the performance.

We use 10-fold cross validation and grid search and compare the accuracy scores to determine the hyper-parameter values of machine learning models.

6 Results and Discussion

During the initial experiment, we set the the size of training dataset to be 6, i.e., for each time stamp as a testing set, we will attain six historical time stamps that have the closest market environment and pattern based on KL-divergence. The lift curves are significantly above the reference line $y = 1$ of random guess and indicates that the models work much better than random guess (Figure 5). We also observed that the Support Vector Machine combined with KL approach (SVM+KL) does not have noticeable improvement over the SVM model based on the sliding window. One potential reason is that the SVM creates its decision boundary only based on some representatives of the dataset (a.k.a, support vectors) even our KL-divergence approach produces more samples for training than does the sliding window. The resultant support vectors between the KL-divergence and sliding-window remain the same and thereby define the same decision boundary with comparable performance.

During the experiment, we noticed that the Naive Bayes model has the largest score for Area-Under-Curse (AUC) and demonstrates the best performance. Moreover, Naive Bayes dominates other models in terms of training time (several minutes versus up to several hours on personal computer). We did thorough experiments to confirm its robustness. Figure 6 shows the robustness of the Naive Bayes model that is trained with the training datasets of different sizes. The AUCs scores by KL and SW are 1.080 and 1.070 for the training size 4, and 1.074 and 1.060 for the training size 2 respectively. These results are consistent with initial experiments. We developed a portfolio management

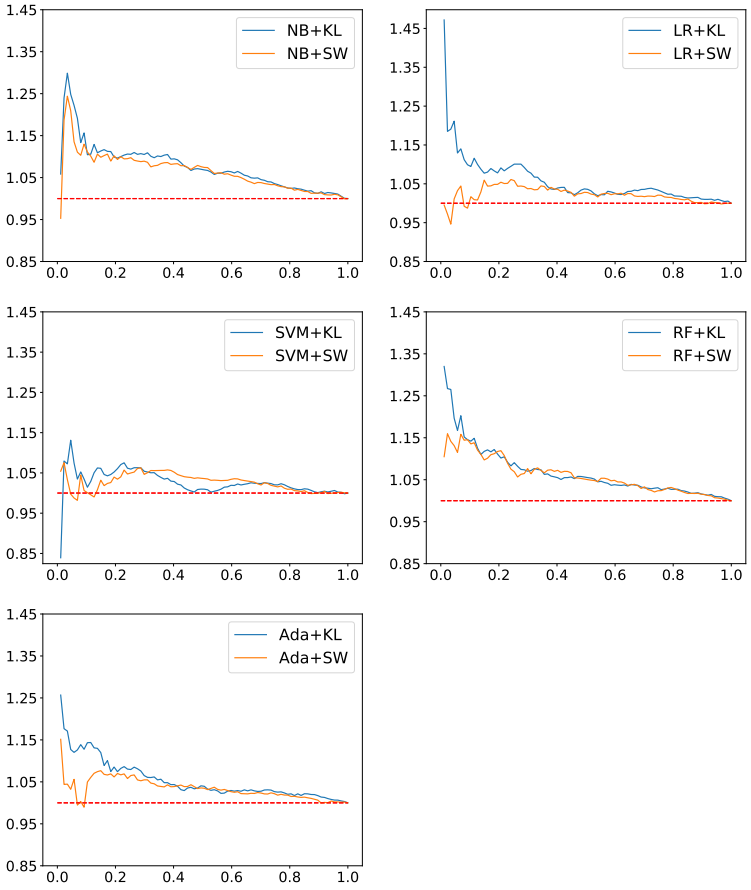

Figure 5: Model Lift Curves Based on KL-divergence v.s. Sliding Window

Table 2: AUC of Lift Curves on Testing set

	KL	SW
Naive Bayes	1.076	1.064
LR	1.056	1.022
SVM	1.027	1.027
Random Forest	1.068	1.060
AdaBoost	1.054	1.034

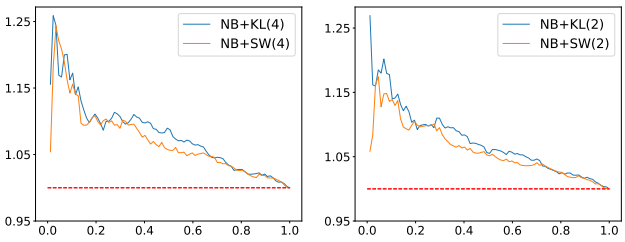

Figure 6: Naive Bayes Lift Curves Given Different Training Set Size

strategy that uses prediction result to make trading decision and did back-test based on this trading strategy. The S&P500 index has a simple addition return of 24.51% within the same back-test period and the benchmark (stock pool) has 27.29% within in the same period. They have the compound returns of 30.68% and 35.32% within the same period respectively. Most of our trading strategies based on prediction outperform S&P500 and the stock pool (Table 5), especially for the KL-divergence approach and those selected predictions. Apparently, the post-selection is effective in applying machine learning in selecting stocks in finance market because the smaller the α is, the larger the accuracy and back-test return we will obtain.

In terms of accuracy and back-test performance, KL-divergence mostly dominates sliding window. The area under lift curve given by KL is always larger than of equal to sliding window although the curves are indeed rough and some parts of KL lift curve are under the corresponding parts of sliding window. This is partially due to the relatively small size of the data set. In addition,

Table 3: Naive Bayes Accuracy on Testing Set

α	KL(6)	SW(6)	KL(4)	SW(4)	KL(2)	SW(2)
0.5	53.48%	53.84%	54.69%	52.87%	52.75%	52.75%
0.4	54.54%	53.85%	54.69%	53.70%	54.23%	53.17%
0.3	55.31%	54.02%	55.11%	54.82%	55.11%	54.22%
0.2	54.89%	55.03%	55.34%	55.19%	54.89%	55.33%
0.1	56.34%	55.10%	59.44%	57.59%	58.20%	56.97%

Table 4: Naive Bayes Back-test Performance on Testing Set (simple addition)

α	KL(6)	SW(6)	KL(4)	SW(4)	KL(2)	SW(2)
0.5	29.09%	29.27%	30.49%	26.08%	29.30%	27.63%
0.4	30.99%	30.87%	31.89%	28.67%	30.97%	29.40%
0.3	32.91%	34.29%	35.05%	30.43%	34.77%	31.95%
0.2	34.58%	32.91%	35.52%	32.01%	35.43%	33.08%
0.1	38.58%	34.71%	42.67%	33.11%	36.29%	44.60%

Table 5: Naive Bayes Back-test Performance on Testing Set (compound cumulative)

α	KL(6)	SW(6)	KL(4)	SW(4)	KL(2)	SW(2)
0.5	34.60%	34.88%	36.31%	30.64%	35.46%	32.52%
0.4	36.57%	36.31%	38.15%	33.33%	36.31%	33.70%
0.3	39.72%	41.72%	42.98%	35.56%	41.09%	37.48%
0.2	41.82%	39.97%	44.49%	39.04%	42.70%	39.83%
0.1	46.84%	42.94%	54.19%	39.04%	45.74%	56.73%

as stated in the beginning, financial time series prediction has always been a difficult problem not only because of its heterogeneity, but also the noise and human factors–in many situations we are not even confident about the signals or the signals could be completely fake.

As a supplementary to the back-test return, we also use the prediction to construct a portfolio and combine with the historical price to recover the return trend of S&P500, the stock pool and the portfolio (figure 7, training set size of 4 and $\alpha = 0.3$). The KL portfolio significantly outperforms the rest while the SW portfolio couldn't beat the stock pool. It is interesting to see that KL portfolio rises up remarkably in the second half of the back-test period. One possible explanation is that in the beginning, there is barely anything to select for KL. This leads to the fact that the training set of KL is quite similar to the training set of SW in the beginning and only when the time goes further there appears difference.

7 Conclusion and Future Work

In this paper, we demonstrate that the commonly used machine learning models are effective in predicting the trend of financial time series and ensuring the investment strategies based on the prediction results to be profitable. We discussed an essential problem in financial market prediction, i.e., the heterogeneity caused by the stochastic nature of stock movement and market environment will invalidate any machine learning that claims to capture the patterns over an extended period. On the other hand, the market has cyclic patterns, which motivates our KL-divergence-based similarity to measure the relevant between two time points and identify the time regime with similar market behavior. Ex-

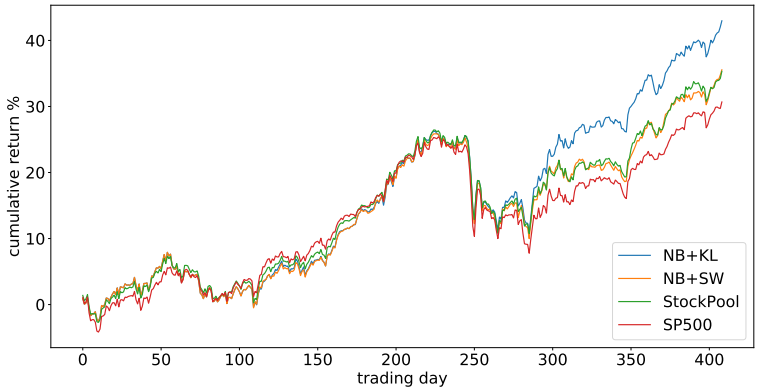

Figure 7: Back Test Cumulative Return

periments confirm that the proposed approach integrates all historical values of multiple time points and trains machine learning models with sufficient signal. In particular, our approach has a better generalization performance than those based sliding windows. The portfolio based on our prediction algorithm outperforms the stock pool benchmark by up to 20% and S&P500 index by up to 25%. In addition, we discovered that a simple Naive Bayes model is effective, robust and fast during training and inference comparing to other relatively complex models.

We envision our future work will incorporate active learning and transfer learning to quantify the data shift problem and include more statistical distributions (student t-distribution and Laplace distribution) to model the stock return that often does not fit Gaussian distribution very well because the empirical stock return distribution usually has a tall peak in the middle and heavy tail. Rescaled student t-distribution may describe stock returns[1] better than the standard Gaussian model.

References

[1] Argimiro Arratia. *Atlantis Studies in Computational Finance and Financial Engineering. Statistics of Financial Time Series*. Springer, 2014.

[2] Michel Ballings et al. "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500". In: *Expert Systems with Applications* 42 (2015), pp. 7046–7056.

[3] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. "Active learning with statistical models". In: *Journal of artificial intelligence research* 4 (1996), pp. 129–145.

[4] John Duchi. *Derivations for linear algebra and optimization*. Berkeley, Califonia, 2007.

[5] Yoav Freund and Robert E. Schapire. "A Short Introduction to Boosting". In: *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1999, pp. 1401–1406.

[6] David Hallac et al. "Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[8] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting stock market movement direction with support vector machine". In: *Computers & Operations Research* 32.10 (2005), pp. 2513–2522.

[9] Eamonn Keogh et al. "An Online Algorithm for Segmenting Time Series". In: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE. 2001.

[10] Christopher Krauss, Xuan Anh Do, and Nicolas Huck. "Evaluating multiple classifiers for stock price direction prediction". In: *European Journal of Operational Research* 259 (2017), pp. 689–702.

[11] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *Ann. Math. Statist.* 22.1 (1951), pp. 79–86.

[12] Tim Loughran and Bill McDonald. "Information decay and financial disclosures". In: *working paper* (2014).

[13] Jose G Moreno-Torres et al. "A unifying view on dataset shift in classification". In: *Pattern Recognition* 45.1 (2012), pp. 521–530.

[14] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[15] Burr Settles. "Active learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (2012), pp. 1–114.

[16] Leo Breiman Statistics and Leo Breiman. "Random Forests". In: *Machine Learning*. 2001, pp. 5–32.

[17] Masashi Sugiyama, Neil D Lawrence, Anton Schwaighofer, et al. *Dataset shift in machine learning*. The MIT Press, 2017.