UNIVERSITY OF GRONINGEN

FACULTY OF ECONOMICS AND BUSINESS

PROJECT FOR LEARNING AND PRACTICE RESEARCH

Volatility Prediction based on Machine Learning Methods

Author

SHUYI WANG

S3068145

Supervisor

Prof. Dr. WALT POHL

February 17, 2021

Contents

Abstract

Implied volatility is an indicator that investors care and use to evaluate the expectation of market returns. It is negatively correlated with returns. We care about the positive or negative changes of VIX, a measure of implied volatility, and commonly considered macroeconomic variables that affect market performance. Moving variances of the return of underlying asset are another features, proved by previous researches. We use monthly recorded time-series data from 2012 to 2019. Besides, in order to improve the predicting accuracy, we consider prevalent machine learning methods. In this article, eight machine learning techniques are considered and compared to a base model, logistic regression model (with accurcay 57.9%) in econometric. To control for over-fitting, we consider cross validation technique with 10 splits. We show that Multi-Layer Perception (with acurracy 63.2%) algorithm outperforms other models using the original data. We further apply dimension reduction technique (PCA) for avoiding noise in the original data. We select the first six principal components that explain 96.96% of the variance of the data, and we model again with the extracted data. We find that accuracy scores are all improved, and the Perceptron method offers the highest accuracy (68.4%). However, under both situations (with or without PCA), only one method can outperform the base model. Compared to logistic regression, most machine learning methods require higher time and space complexity. Hence, we do not consider machine learning suits our scenario.

Keywords: implied volatility, VIX, machine learning, dimension reduction

1 Introduction

In financial engineering, volatility plays a crucial role in option design. Options investors, especially in short positions, are concerned with an unexpected volatility spike causing rapid price changes. In other words, volatility represents the risk that is highly relative to returns. In order to gain stable returns, option sellers should construct dynamic hedging strategies. Hence, analyzing the features of volatility and predicting its shocks are prioritized. There are two main kinds of volatility: *Historical Volatility* and *Implied Volatility*. Unlike historical volatility that is calculated from simple time-series of financial return (see, Koopman et al., 2005), the implicit volatility is not observable and reveals information on expected market(see, e.g., Dumas et al., 1998; Goncalves and Guidolin, 2006). Implied volatility is backward calculated from the well-known Black-Schole (1973) formula. As four given parameters of the formula are pre-determined by the market, we can retrieve this volatility. General approaches such as the class of ARCH models (see, Engle, 1982) provided with parameterized methods in estimating and forecasting volatility. Although provided with appropriate accuracy, Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) model still suffers from its assumption that it treats upward and downward price changes equally. This is easily violated according to classic asset pricing theory (see, Cochrane, 2009). In order to reach higher precision of forecasting, we demonstrate several prevalent machine learning techniques. We are interested in the directions of volatility changes since they imply investors reactions to future prices. Given the significant improvements of computational power in the past decades, we are capable of manipulating big data techniques, recursively training models and generating more accurate outcomes. Besides, the ability to analyze huge amount of historical data with computational systems allows us to build intelligent predictive models (see, Chiang et al., 2016). Further, ensemble learning allows us to construct algorithm with combining multiple models to enhance generalization ability. Breiman 1996 introduced Bootstrap AGGregatING (Bagging) and improved the forecasting by reducing variance. We implement the ensemble techniques to improve the current learning performance and compute the precision changes across sub-samples.

We access an average volatility measure Volatility Index (VIX, also known as fear index), which represents the market expectation of implied volatility based on the underlying options. Jiang and Tian (2007) demonstrate the calculation and an error correction process that polishes VIXs. Economic activity and financial instability are considered two components of VIXs (see, Bekaert and Hoerova, 2014). However, most works apply technical analysis indicators when applying machine learning research. To differentiate and to gain more economic sense, we apply macroeconomic indicators as predictors. In addition, we investigate the developing market.

The remainder of the paper is organized as follows. Section 2 reviews literature related to both financial time-series research with machine learning and volatility research from economic and econometric views. Section 3 introduces our prioritized learning techniques, including KNN, SVMs, Random Forest, Gradient Boosting Decision Tree (GBDT), etc., and data structure applied in this paper. Section 4 compares machine learning methods with classic Logistic Regression in dealing with classification problems. Section 5 finalizes the paper with conclusions and limitations.

2 Literature Review

2.1 *Machine Learning in Financial Time-Series Analysis*

The application of the machine learning in financial market rises along with the concept artificial intelligence brought out at Dartmouth Conference in 1956 (see, McCarthy et al., n.d.). Time-series analysis is widely considered in financial market, machine learning methods present promised qualities in forecasting time-series. Chen (2003) presents that probabilistic neural network (PNN) approach outperforms generalized methods of moments (GMM) with Kalman filter. The methods predict the direction of index returns in Taiwan. Chen and Xiao (2017) develop trading strategies based on novel double-layer neural (DNN) network for high-frequency forecasting. Given a set of good-of-fit measures indicating different layers, they propose such framework that beats benchmark methods in terms of the prediction accuracy and return. The difficulties in financial time-series are related to the non-stationary nature. Tay and Cao (2001) examine the feasibility of SVMs and prove the advantages of applying SVMs in financial time-series forecasting. Zhang and Lin (2017) compare K nearest neighbor (KNN) with econometric models (such as ARIMA), and they extend the method with ensemble empirical mode decomposition (EEMD) in multidimensional space (MKNN) in order to predict close and high price simultaneously. By EEMD-MKNN, they manage to reach higher precision than by ARIMA. In addition to predict direction, classification methods (such as SVM and KNN) are adapted to regression style method and thus are able to predict values for financial time-series. Huang and Tsai (2009) apply a Support Vector Regression (SVR) and combine it with Self-Organizing Feature Map (SOFM) to predict values for Taiwan market index. Henrique and Sobreiro (2019) summarize applications of machine learning and conclude that, in financial time-series analysis, SVMs and neural networks are the most commonly considered algorithms. They also highlight the opportunities still exist in emerging markets, considering research scenarios and accessibility of data.

2.2 *Volatility Research*

This paper focuses on implied volatility. Volatility contributes to a (unexpected) significant price change on spot market. Becker et al. (2009) analyze price “jumps” and conclude that VIX captures the information content regarding future price change on spot market. They also highlight the current implied volatility (indicated by VIX) is uncorrelated with the realized information content regarding price jumps. According to their research, using VIX as a measure is reasonable since VIX contains the information of volatility that affects the market. Becker and Hoerova (2014) also investigates in the relation between equity variance premium (VP) and conditional variance (CV). They decompose VIX ($VIX^2 = VP + CV$) and reveal the VIX is related to economic activity and financial stress. To connect with the economic activities, in this paper, we select macroeconomic indicators as predictors. A classic econometric view of estimating and forecasting volatility is offered by GARCH model (Bollerslev, 1986). Chen (2010) applies SVMs under GARCH framework to forecast volatility. The result shows that the SVM-GARCH model strictly outperforms the standard GARCH, EGARCH and moving average predictions. Hybrid methods attract attentions regarding volatility forecating. Kim and Won (2018) combine deep learning method Long short-term memory (LSTM) with GARCH framework. They discover GEW-LSTM, a hybrid method combining LSTM with three GARCHs, and show significance improvements on three error measurements, which are error (MAE), mean squared error (MSE), heteroscedasticity adjusted MAE (HMAE). Those researches not only found new approaches that promise performance, but also optimize the econometric model from a statistical perspective. It is of interest to investigate outcome if the assumption of GARCH does not hold. Since the Maximum Likelyhood estimator (MLE) assumed normal distribution is easily violated, Hwang and Shin apply kernel techniques for GARCH and improve the predicting power. Without conventional model or hybrid models, researches apply fancy machine learning and deep learning methods for forecasting (see, e.g., Zeng and Klabjan, 2019, Horvath et al., 2019, Xiong et al., 2015).

Another practical views are from Bayesian statistics. Different from frequentist, Bayesian inference relies on posterior estimation. Its MLE concentrates on conditional probability distribution.

Bayesian logic, in some cases, can generate more accurate prediction and lower variance. Jacquier (1994) performs a comparison between Bayesian estimator and method of moments and quasi-maximum likelihood estimators for prediction. By sampling experiments, Bayesian estimator outperforms the others. Law and Shawe-Taylor (2017) apply bayesian support vector regression for financial series modeling. They propose new parameter in SVR to realize an automatic parameter selection based on bayesian learning.

3 Descriptive Statistics and Methodologies

3.1 *Data Description*

As previous review showing research opportunities of developing market, this article accesses China ETF VIX recorded by CBOE[1] and China 50ETF price. Constructed similarly to other main indexes, 50ETF contains the top 50 stocks covering more than one third of the market value. Its corresponding VIX reveals market expected reaction to price changes. A time-series plot displays these trends in Figure.1. The circled parts highlight the opposite trends (50ETF vs. VIX) as theory mentioned. We construct data-frame with monthly data of VIX since macroeconomic indicators

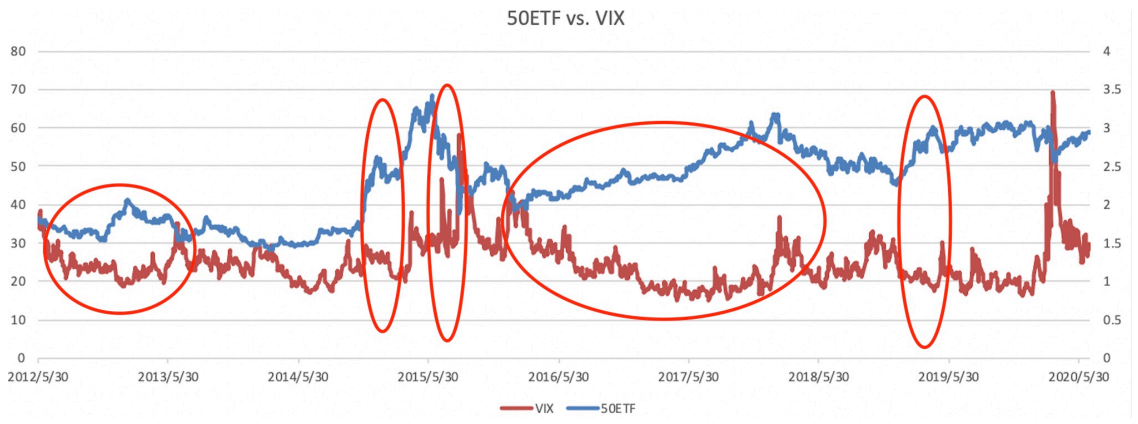

Figure 1: 50ETF vs. VIX

are monthly recorded. The response variable is VIX changes direction (positive/negative). We present the predictors description in Table.1. In order to have a clear view of correlation between variables, we demonstrate a pair-plot for all the variables as shown in Figure.2. The basic setup of

[1]Check by http://www.cboe.com/products/vix-index-volatility/volatility-on-etfs/cboe-china-etf-volatility-index-vxfxi

Variable Names	Description
CBI	Consumer Booming Index measures the consumer confidence of purchasing goods
PMI	Purchasing Managers' Index measures the prevailing direction of economic trends in manufacturing
CPI	Consumer Price Index measures average change overtime in the prices paid by urban consumers for a market basket of consumer goods and services
M2	Money Supply measures the amounts of cash, checking deposits, and easily convertible near money
Gold&FX	Gold and foreign currency reserves
MV10&MV20&MV30	Moving variance for 10, 20 and 30-day window

Table 1: Variables Descriptive

machine learning is to split data for training, validating and testing. In this article, the complete data-frame is an 89×10 matrix, including response variable. The test set is not observable and is not involved in training or validating the model. We set 20% of the data for testing, and the rest for modeling. Given the auto-correlation nature of time-series, we apply cross validation method *timeseriessplit* rendered by Scikit-Learn to control potential over-fitting. The cross validation method splits data successively. The Kth set is for validation with only accessing $(K-1)th$ for training. The corresponding parameters is evaluated by their accuracy, and the optimal ones will be applied to test set. This paper splits the data into 10 iterations. The Figure.3 presents how this process works.

3.2 *Models*

This article considers the following prevalent classification methods, Logistic Regression as a base model, Perceptron and with Multi-layer Perceptron (simple supervised Neural Network algorithm) and Naive Bayesian. Besides, this article focuses on another five widely applied methods, where are Support Vector Classifier, K Nearest Neighbor Classifier, Random Forest, Adaptive Boosting (AdaBoosting) Classifier and Gradient Boosting Decision Tree (GBDT) Classifier. The realization of those methods from scratch is beyond this article, we implement most of them by

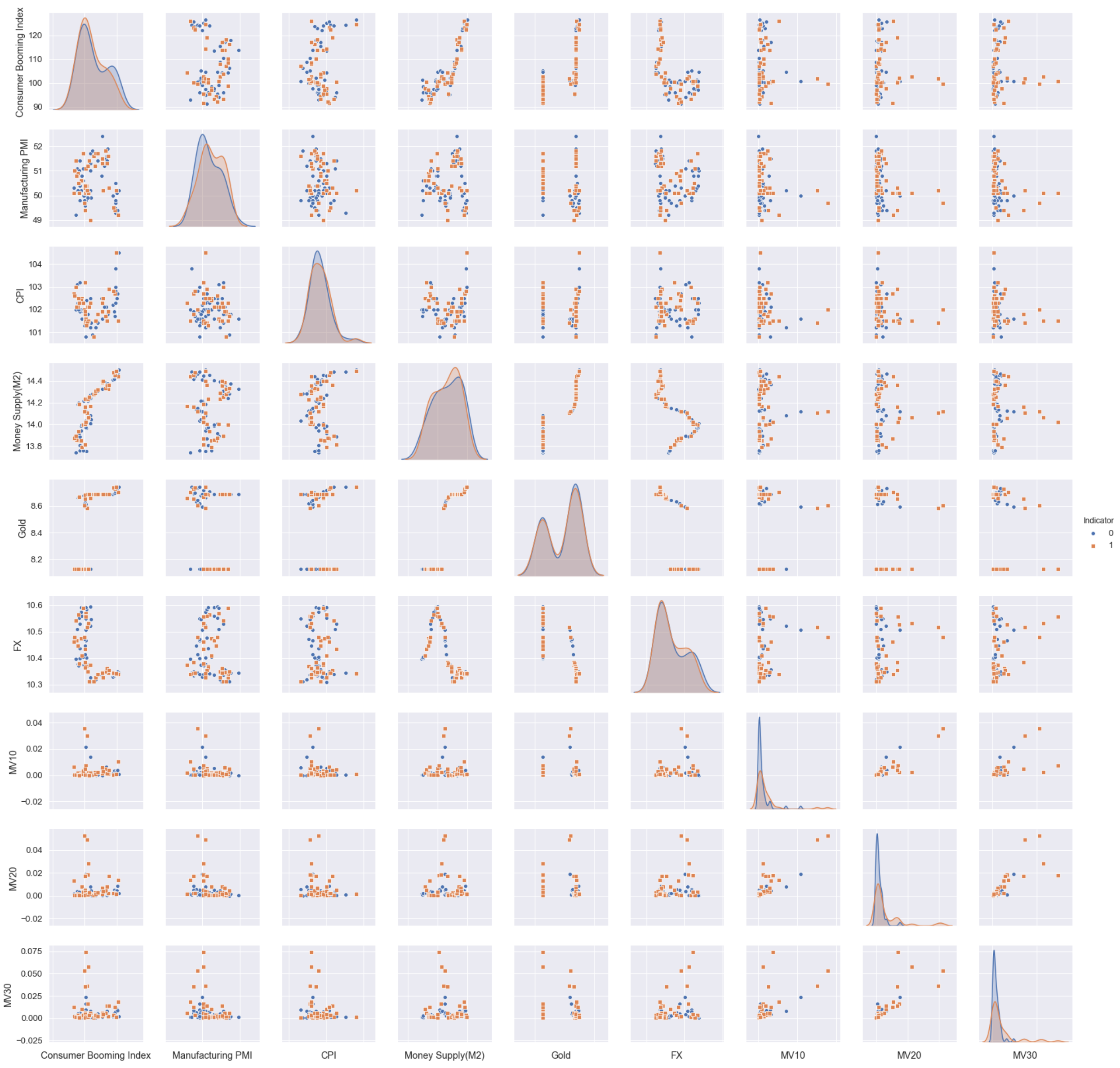

Figure 2: Pair-plot for Features

Python Package SciKit-Learn (see, Pedregosa et al., 2011). However, we managed to derive the formulas as seen in APPENDIX.Given the short time series and 9 features, we apply Principal Component Analysis (PCA) to reduce dimensions for improving predictions.

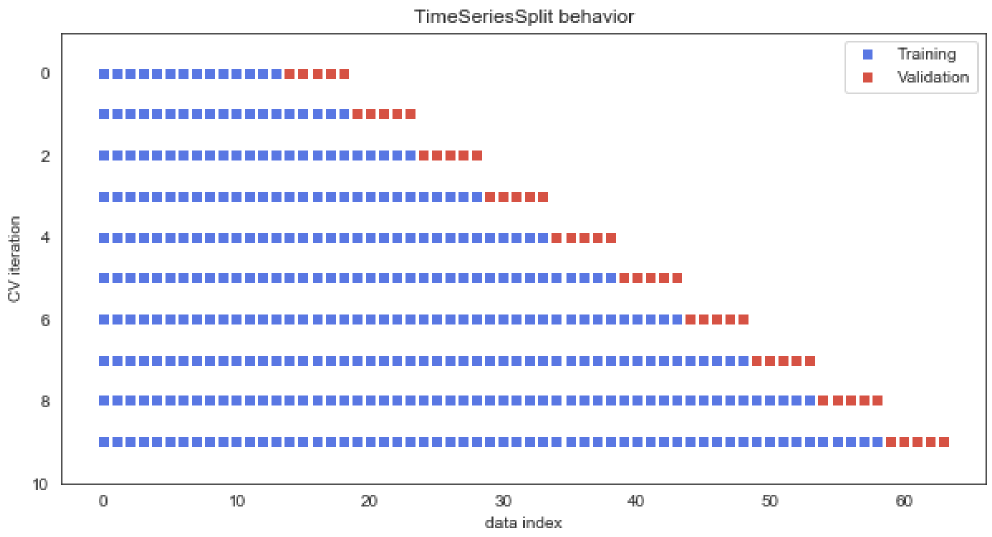

Figure 3: Time Series Cross Validation

3.3 *Model Selection*

As an important component for construction of model, tuning hyper-parameters is always considered a time-consuming and technical work. SciKit-Learning provides *GridSearchCV* and *RandomizedSearchCV*. The former search tests every possible combination of parameters and returns the tuple of parameters with best model selection metrics, and the latter search tests randomly across the sets of parameters with given iteration times. This paper demonstrates an automatic search using the rendered methods.

In order to evaluate the model, we apply four commonly considered indicators accuracy score, recall score, precision and f1 score. They are calculated from the confusion matrix as shown in Table.2.

	Positive Predicted	Negative Predicted
Positive Actual	**TP**	**FN**
Negative Actual	**FP**	**TN**

Table 2: Confusion Matrix

$$\begin{aligned}
Accuracy &= \frac{TP + TN}{TP + FN + FP + TN};\\
Recall &= \frac{TP}{TP + FN};\\
Precision &= \frac{TP}{TP + FP};\\
F1 &= \frac{2 \times (Precision \times Recall)}{Precision + Recall}.
\end{aligned}$$

This paper displays the confusion matrix dynamics with different sets of parameters and selects the optimal sets for our prioritized models.

4 Empirical Learning Outcomes

4.1 *Learning without Dimensional Reduction*

Logistic Regression is a classic approach for classification problem so that this paper treats it as a base model, with accuracy score 0.579. In other words, 11 out of 19 in the test set are correctly classified. Likewise, we access the accuracy and confusion matrix of another three methods as displayed in Figure.4. Tuned to optimal parameters through $GridsearchCV$, Perceptron and MLP

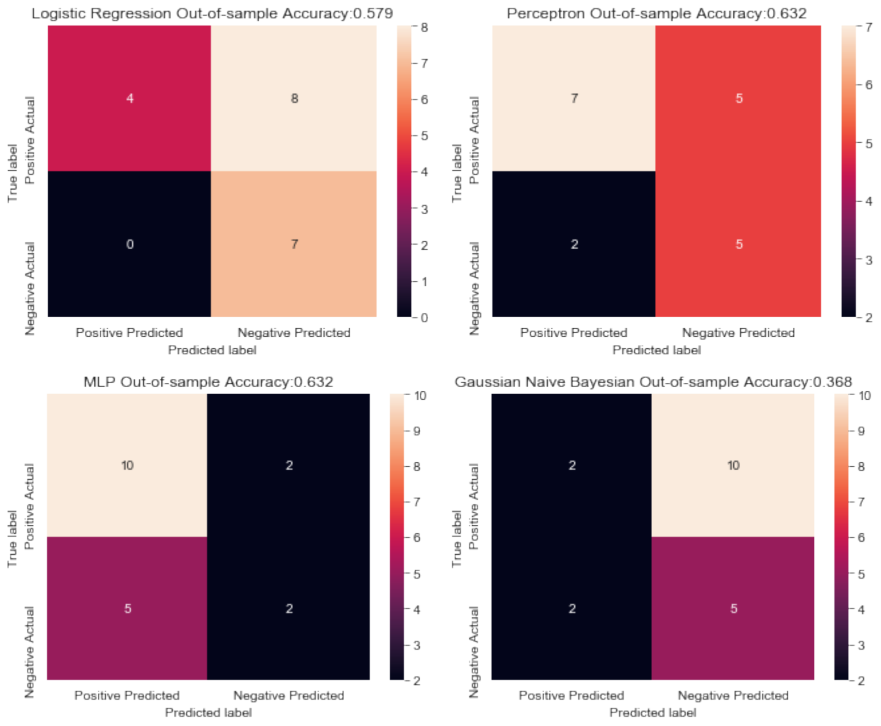

Figure 4: Confusion Matrix Heatmap

classifiers present the same accuracy 0.632. These two methods outperform the base model and Naive Bayesian, with 0.368 for accuracy. An overall summary of the model selection metrics is shown in Table.4.

A more thorough parameters tuning is applied to KNN, SVM, GBDT, Random Forest and AdaBoosting methods. We observe an increasing accuracy with the sequence of the methods. Starting from KNN, we perform a selection for number of neighbors and weighted methods. A grid search screen all the combinations of parameters, finding out $n\_neighbors = 3$ and $weight = uniform$ are the optimal fit. For SVM, we present a selection of γ, indicating how much influence a single training example has, and C, which is a regularization parameter. $\gamma = 0.01$ and $C = 61.5454$ with a *rbf* kernel is the optimal choice.

For three tree models, we are concerned with three parameters that affect the model performance, *number of estimators, maximum features* and *learning rate*. Number of estimator decides the number sub-samples extracted from bootstrap. Small estimator causes under-fitting, however large estimator will not boost the model significantly after a critical value. The maximum features determine the size of the random subsets of features to consider when splitting a node. These two parameters should be seriously considered since default setting is not able to capture the characteristics of the data. In this article, we set the number of estimator in [1,100] and maximum features in [1,9], and we use $GridSearchCV$ to search and test all the possible combination and return with the highest score. Learning rate, also known as gradient coefficient, is a parameter that scales the gradient descending speed. It is possible that we will miss the optimal result if we set the learning rate too large. On the other hand, a low learning rate will cause computation burden. In the article, we set learning rate in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. It should be noticed that the set for learning rate is still “jumpy”, however, it is already 9 times the load of calculation. Due to the small dataset, we leave the nodes and leaf as default. As shown in Figure.5, Random Forest and AdaBoosting have the highest accuracy 0.579, however Adaboosting has higher Precision and lower Recall. GBDT does not outperform SVM, despite its complexity, hence we do not consider this model. A more detailed parameter tuning information for all methods is listed in the Table.3.

Method	Parameters
LR	C=1.0, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='auto', penalty='l2', random_state=0, solver='liblinear',tol=0.0001
Perceptron	alpha=0.0001, eta0=0.1, fit_intercept=True, max_iter=1000, n_iter_no_change=5, random_state=0,tol=0.001,validation_fraction=0.1
MLP	activation='tanh', alpha=0.0001, batch_size='auto', beta_1=0.9, beta_2=0.999, epsilon=1e-08, hidden_layer_sizes=(5, 4), learning_rate='constant', learning_rate_init=0.001, max_fun=15000, max_iter=200, momentum=0.9, n_iter_no_change=10, tol=0.0001, power_t=0.5,random_state=0,shuffle=True, solver='sdg', validation_fraction=0.1
KNN	algorithm='auto', leaf_size=30,metric='minkowski',metric_params=None, n_jobs=None, n_neighbors=3, p=2, weights='uniform'
SVM	C=61.5454, break_ties=False, cache_size=200,class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=0.01, kernel='rbf', max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001
Random Forest	bootstrap=True, ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features=2, max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,n_estimators=6, n_jobs=None, oob_score=False, random_state=1
AdaBoosting	algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=18, random_state=1
GBDT	ccp_alpha=0.0, criterion='friedman_mse', learning_rate=0.8, loss='deviance', max_depth=3, max_features=1, n_estimators= 13, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, validation_fraction=0.1

Table 3: Tuning Parameters

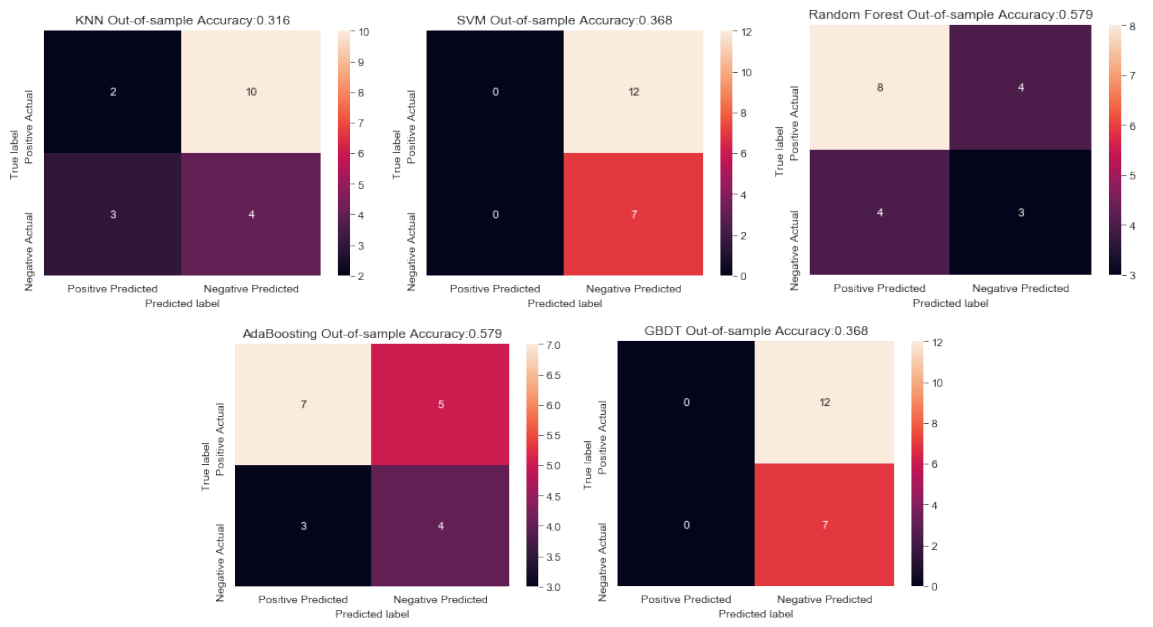

Figure 5: Confusion Matrix Heatmap

4.2 ***Learning with Dimensional Reduction***

It is of interest to improve the current learning outcomes with dimensional reduction techniques and to see parameter changes. Principal Components Analysis (PCA) is able to decompose a multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance. With loss of some information, we can use PCA to level up accuracy if the omitted variables are indeed noisy. In order to determine the number of principal components, this paper arranges the explained variance ratio from top to bottom as displayed in Figure.6. We select the first six components explaining 96.96% of the variance and reconstruct the data-frame based on this selection.

We use *GridSeachCV* for tuning hyperparameters and demonstrate the confusion matrix in Figure.7 with detailed metrics in Table.4. The parameters tuning range remains the same as in previous part. Accuracy scores are all significantly improved. We find that tree models reach higher accuracy to 0.579 when applied dimensional reduction. Perceptron classifier generates the highest out-of-sample accuracy 0.684. Compared to methods without PCA, fewer dimension models successfully obtain higher accuracy across methods. Hence, in our case, we believe modelling with PCA efficiently improves predictions.

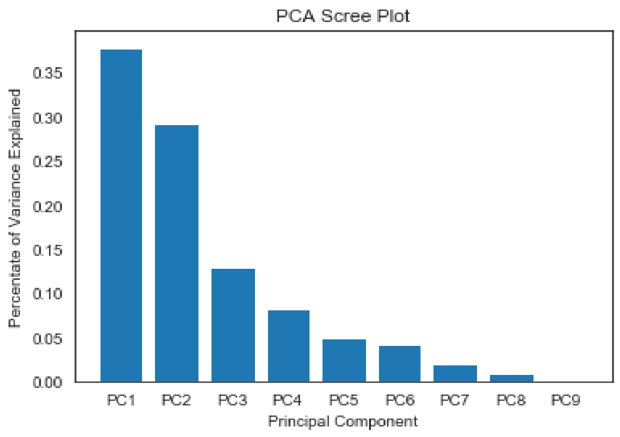

Figure 6: Principal Components Analysis

As expected, parameters across methods change after PCA except for based model. For KNN, the optimal number of neighbors changes from 3 to 2 with weight remaining $uniform$. Multi-layer Perceptron switches activation function from hyperbolic tan function to rectified linear unit function, and it changes $sovler$ to $adam$ gradient-based optimizer. For tree models, the reshape of data can easily affect related parameters. A more detailed parameter tuning information for methods based on PCA is displayed in Table.5.

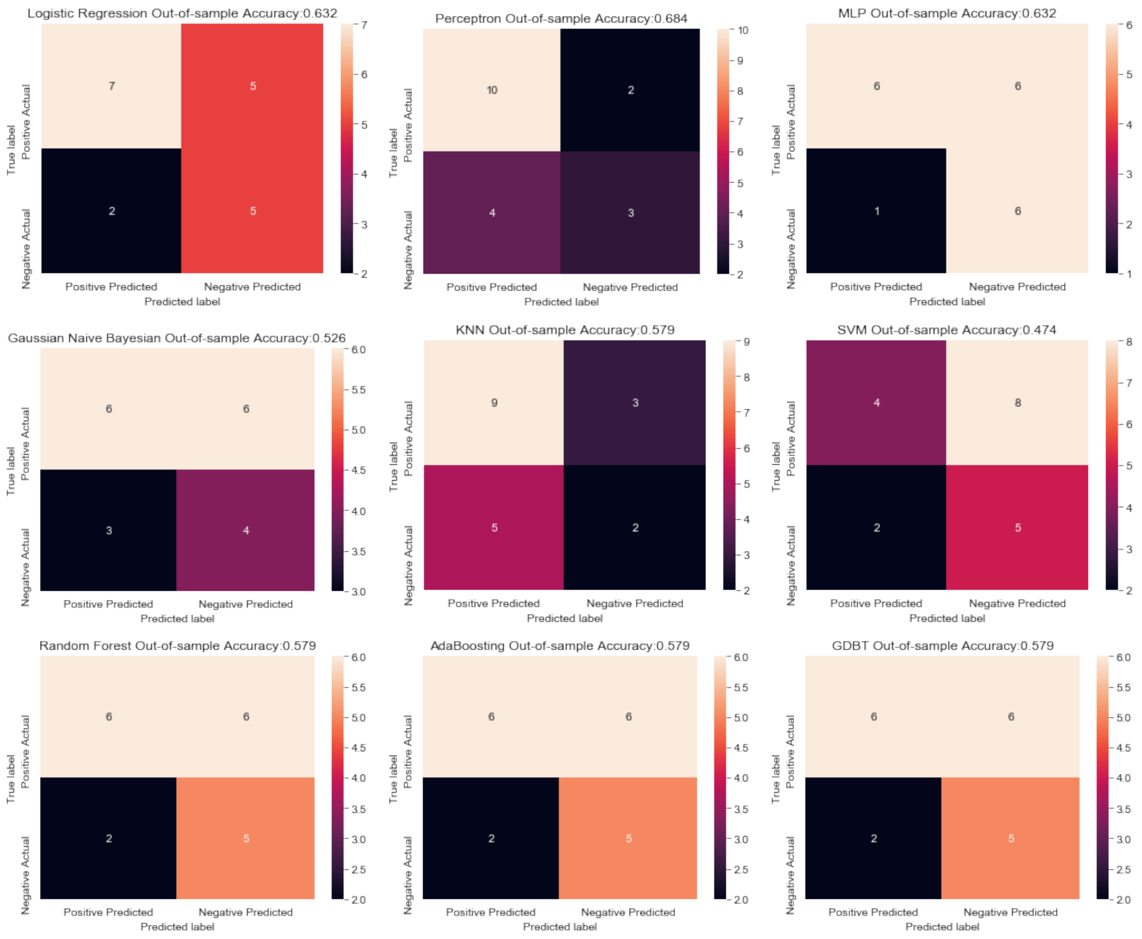

Figure 7: Learning with PCA Confusion Matrix Heatmap

	Accuracy	Precision	Recall	F1 Score	AUC
without PCA					
Logistic Regression	0.579	1.000	0.364	0.534	0.667
Perceptron	0.632	-	1.000	-	0.649
Multi-layer Perceptron	0.632	0.667	0.883	0.741	0.560
Naive Bayesian	0.368	0.500	0.286	0.364	0.440
KNN	0.316	0.400	0.333	0.363	0.369
SVM	0.368	-	-	-	0.5
Random Forest	0.579	0.667	0.727	0.696	0.548
AdaBoosting	0.579	0.700	0.636	0.688	0.577
GBDT	0.368	-	-	-	0.5
with PCA					
Logistic Regression	0.632	0.778	0.583	0.667	0.689
Perceptron	0.684	0.714	0.769	0.740	0.631
Multi-layer Perceptron	0.632	0.857	0.5	0.632	0.648
Naive Bayesian	0.526	0.667	0.600	0.632	0.536
KNN	0.579	0.643	0.818	0.720	0.518
SVM	0.474	0.667	0.444	0.533	0.524
Random Forest	0.579	0.750	0.545	0.631	0.607
AdaBoosting	0.579	0.750	0.545	0.631	0.607
GBDT	0.579	0.750	0.545	0.631	0.607

Table 4: Model Selection Metrics

Methods	Parameters
LR	C=1.0, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class=’auto’, penalty=’l2’, random_state=0, solver=’liblinear’,tol=0.0001
Perceptron	alpha=0.0001, eta0=0.1, fit_intercept=True, max_iter=1000, n_iter_no_change=5, random_state=0,tol=0.001,validation_fraction=0.1
MLP	activation=’relu’, alpha=0.0001, batch_size=’auto’, beta_1=0.9, beta_2=0.999, epsilon=1e-08, hidden_layer_sizes=(5, 4), learning_rate=’constant’, learning_rate_init=0.001, max_fun=15000, max_iter=200, momentum=0.9, n_iter_no_change=10, tol=0.0001, power_t=0.5,random_state=0, shuffle=True, solver=’adam’, validation_fraction=0.1
KNN	algorithm=’auto’, leaf_size=30,metric=’minkowski’,metric_params=None, n_jobs=None, n_neighbors=5, p=2, weights=’uniform’
SVM	C=11.0909, break_ties=False, cache_size=200,class_weight=None, coef0=0.0, decision_function_shape=’ovr’, degree=3, gamma=0.01, kernel=’rbf’, max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001
Random Forest	bootstrap=True, ccp_alpha=0.0, class_weight=None, criterion=’gini’, max_depth=None, max_features=1, max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,n_estimators=5, n_jobs=None, oob_score=False, random_state=1
AdaBoosting	algorithm=’SAMME.R’, base_estimator=None, learning_rate=1.0, n_estimators=36, random_state=1
GBDT	ccp_alpha=0.0, criterion=’friedman_mse’, learning_rate=0.6, loss=’deviance’, max_depth=3, max_features=2, n_estimators= 3 max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, validation_fraction=0.1

Table 5: Parameters after PCA

5 Conclusion and Limitation

5.1 *Conclusions*

To summarize, in this article, most machine learning methods do not demonstrate more competitive predicting power than the classic econometric model, namely, logistic regression. However, this is partly due to irrelevant noisy data and variables. We thus reduce the number of variables to remove noise and build models afterward. The accuracy scores are improved with the first six principal components explaining 96.96% of the variance of the data.

Prior to dimensional reduction, base model logistic regression (with accuracy 0.579) outperforms most of machine learning methods considered in this article, only beat by a simple neural network Multi-layer Perceptron (with accuracy 0.632). Furthermore, we can observe five methods (Perceptron, Naive Bayesian, KNN, SVM and GBDT) present accuracy lower than 0.400. As shown in Table.2, PCA increases accuracy across the models. Perceptron algorithm sees the largest improvement and is finally considered the best method with 0.684 accuracy in this article. The dimension reduction shows a positively promising effect on prediction, and we find the lowest accuracy is 0.474 presented by SVM, compared to 0.316 presented by KNN. Although machine learning method provides higher accuracy and precision, it always requires more time and space complexity. From a comprehensive perspective, machine learning seems not very productive, compared to some simple models. Given enough time and computational power, machine learning should perform better.

5.2 *Limitations*

Three limits are also considered. First, the dataset only contains 89 observations, given the data is monthly recorded. However, we find six commonly considered macroeconomic indicators that affect the economy and thus index ETF. This might cause the so-called “Dimensional Trap”, even if we try to drop variables in order to achieve higher accuracy. A solution should go with finding more observations and expand the longitude of data. Besides, given the small sample of test set, it is not ideal to distinguish the prediction accuracy. For tree models, their accuracy scores are the same. It is might due to the small test size. Second, we only access an average value of our research object. Although papers have shown that VIX can equally function as market expectation of investors, we still lose some information contained in the implied volatility. Third, we do not tune all the hyper-parameters when building the model. Partly since tuning some parameters is not necessary in a small dataset, partly because tuning all the parameters is time-consuming given my computer power.

References

Becker, R., Clements, A. E., & McClelland, A. (2009). The jump component of s&p 500 volatility and the vix index. *Journal of Banking & Finance*, *33*(6), 1033–1038.

Bekaert, G., & Hoerova, M. (2014). The vix, the variance premium and stock market volatility. *Journal of econometrics*, *183*(2), 181–192.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, *81*(3), 637–654.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327. https://doi.org/https://doi.org/10.1016/0304-4076(86)90063-1

Breiman, L. (1996). Out-of-bag estimation.

Chen, A.-S., Leung, M. T., & Daouk, H. (2003). Application of neural networks to an emerging financial market: Forecasting and trading the taiwan stock index. *Computers & Operations Research*, *30*(6), 901–923.

Chen, H., Xiao, K., Sun, J., & Wu, S. (2017). A double-layer neural network framework for high-frequency forecasting. *ACM Transactions on Management Information Systems (TMIS)*, *7*(4), 1–17.

Chen, S., Härdle, W. K., & Jeong, K. (2010). Forecasting volatility with support vector machine-based garch model. *Journal of Forecasting*, *29*(4), 406–433.

Chiang, W.-C., Enke, D., Wu, T., & Wang, R. (2016). An adaptive stock index trading decision support system. *Expert Systems with Applications*, *59*, 195–207. https://doi.org/https://doi.org/10.1016/j.eswa.2016.04.025

Cochrane, J. H. (2009). *Asset pricing: Revised edition*. Princeton university press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.

Dumas, B., Fleming, J., & Whaley, R. E. (1998). Implied volatility functions: Empirical tests. *The Journal of Finance*, *53*(6), 2059–2106.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the Econometric Society*, 987–1007.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1). Springer series in statistics New York.

Goncalves, S., & Guidolin, M. (2006). Predictable dynamics in the s&p 500 index options implied volatility surface. *The Journal of Business*, *79*(3), 1591–1635.

Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, *124*, 226–251.

Horvath, B., Muguruza, A., & Tomas, M. (2019). Deep learning volatility. *Available at SSRN 3322085*.

Huang, C.-L., & Tsai, C.-Y. (2009). A hybrid sofm-svr with a filter-based feature selection for stock market forecasting. *Expert Systems with Applications*, *36*(2, Part 1), 1529–1539. https://doi.org/https://doi.org/10.1016/j.eswa.2007.11.062

Hwang, C.-H., & Shin, S.-I. (2010). Estimating garch models using kernel machine learning. *Journal of the Korean Data and Information Science Society*, *21*(3), 419–425.

Jacquier, E., Polson, N. G., & Rossi, P. E. (1994). Bayesian analysis of stochastic volatility models. *Journal of Business Economic Statistics*, *12*(4), 371–389. http://www.jstor.org/stable/1392199

Jiang, G. J., & Tian, Y. S. (2007). Extracting model-free volatility from option prices: An examination of the vix index. *The Journal of Derivatives*, *14*(3), 35–60.

Kim, H. Y., & Won, C. H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, *103*, 25–37.

Koopman, S. J., Jungbacker, B., & Hol, E. (2005). Forecasting daily variability of the s&p 100 stock index using historical, realised and implied volatility measurements. *Journal of Empirical Finance*, *12*(3), 445–475.

Law, T., & Shawe-Taylor, J. (2017). Practical bayesian support vector regression for financial time series prediction and market condition change detection. *Quantitative Finance*, *17*(9), 1403–1416.

McCarthy, J., Minsky, M., Rochester, N., & Shannon, C. (n.d.). August 31, 1955. a proposal for the dartmouth summer research project on artificial intelligence. the 1956 dartmouth summer research project on artificial intelligence, bell telephone laboratories.

Minsky, M., & ML, P. (1990). Sa: Perceptrons. MIT Press, Cambridge.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Tay, F. E., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, *29*(4), 309–317. https://doi.org/https://doi.org/10.1016/S0305-0483(01)00026-3

Wooldridge, J. M. (2010). *Econometric analysis of cross section and panel data*. MIT press.

Xiong, R., Nichols, E. P., & Shen, Y. (2015). Deep learning stock volatility with google domestic trends. *arXiv preprint arXiv:1512.04916*.

Zeng, Y., & Klabjan, D. (2019). Online adaptive machine learning based algorithm for implied volatility surface modeling. *Knowledge-Based Systems*, *163*, 376–391.

Zhang, N., Lin, A., & Shang, P. (2017). Multidimensional k-nearest neighbor model based on eemd for financial time series forecasting. *Physica A: Statistical Mechanics and its Applications*, *477*, 161–173. https://doi.org/https://doi.org/10.1016/j.physa.2017.02.072

Methods in Paper

Linear Models for Classification[2]

Linear model possesses good qualities in both econometric and machine learning prediction. Its comprehensibility is desirable from both theoretical and applicable aspects. Consider d features and observations on those features: $\boldsymbol{x} = (x_1; x_2; ...; x_d)$, x_i is the observation for $\boldsymbol{x}$ on feature i. To write it in vector form:

$$f(\boldsymbol{x}) = \boldsymbol{\omega}^T \boldsymbol{x} + b,$$

where $\boldsymbol{\omega} = (\omega_1; \omega_2; ...; \omega_d)$. Suppose observations:

$$\mathcal{D} : \{(x_i, y_i)\}_{i=1}^N$$

where $x_i \in \mathcal{R}^d, y_i \in \mathcal{R}$. Linear model try to minimize the gap between $f(x_i)$ and y_i, so we can minimize the variance of this gap:

$$\begin{aligned}(\hat{\boldsymbol{\omega}}, \hat{b}) &= arg \min_{\boldsymbol{\omega}, b} \sum_{i=1}^N (f(x_i) - y_i)^2 \\ &= arg \min_{\boldsymbol{\omega}, b} \sum_{i=1}^N (y_i - \boldsymbol{\omega}^T \boldsymbol{x}_i - b)^2\end{aligned}$$

As shown, to minimize $\mathbb{E}_{(\hat{\omega}, \hat{b})} = \sum_{i=1}^N (y_i - \boldsymbol{\omega}^T \boldsymbol{x}_i - b)^2$, a general way is to take derivative and set them to 0:

$$\begin{aligned}\frac{\partial \mathbb{E}_{(\hat{\omega}, \hat{b})}}{\partial \boldsymbol{\omega}} &= 2(\omega \sum_{i=1}^N x_i^2 - \sum_{i=1}^N (y_i - b) x_i) = 0, \\ \frac{\partial \mathbb{E}_{(\hat{\omega}, \hat{b})}}{\partial b} &= 2(Nb - \sum_{i=1}^N (y_i - \omega x_i)) = 0,\end{aligned}$$

A closed-form solution for the above equation:

$$\begin{aligned}\omega &= \frac{\sum_{i=1}^N y_i (x_i - \bar{x})}{\sum_{i=1}^N x_i^2 - \frac{1}{m} (\sum_{i=1}^N x_i)^2}, \\ b &= \frac{1}{m} \sum_{i=1}^N (y_i - \omega x_i),\end{aligned}$$

[2] According to Wooldridge, 2010 and some of my derivation

where $\bar{x} = \frac{1}{m}\sum_{i=1}^{N} x_i$. Simple as it looks, linear model has fruitful transformations applying a link function g^{-1}:

$$y = g^{-1}(\boldsymbol{\omega}^T \boldsymbol{x} + b),$$

and $g(\cdot)$ is monotonous and differentiable. A popular transformation is to function y as:

$$y = \frac{1}{1+e^{-z}}.$$

This function is the so-called logistic function. As one form of Sigmoid function, it transform z to [0,1]:

$$y = \frac{1}{1+e^{-(\omega^T x+b)}},$$

$$ln\frac{y}{1-y} = \boldsymbol{\omega}^T \boldsymbol{x} + b$$

where $\frac{y}{1-y}$ is called "odds" showing the probability of $\boldsymbol{x}$ is positive (1). Rewrite the above function as probability:

$$ln\frac{\mathbb{P}(y=1|\boldsymbol{x})}{\mathbb{P}(y=0|\boldsymbol{x})} = \boldsymbol{\omega}^T \boldsymbol{x} + b$$

$$\longrightarrow \mathbb{P}(y=1|\boldsymbol{x}) = \frac{e^{\boldsymbol{\omega}^T \boldsymbol{x}+b}}{1+e^{\boldsymbol{\omega}^T \boldsymbol{x}+b}},$$

$$\mathbb{P}(y=0|\boldsymbol{x}) = \frac{1}{1+e^{\boldsymbol{\omega}^T \boldsymbol{x}+b}}.$$

Maximum Likely-hood method is commonly used in this case. The log-likelyhood function is:

$$\mathcal{L}(\boldsymbol{\omega}, b) = \sum_{i=1}^{N} ln\mathbb{P}(y_i|\boldsymbol{x}_i; \boldsymbol{\omega}, b).$$

Let $\boldsymbol{\beta} = (\boldsymbol{\omega}; b)$ and $\hat{\boldsymbol{x}} = (\boldsymbol{x}; 1)$, then

$$\boldsymbol{\omega}^T \boldsymbol{x} + b \Rightarrow \boldsymbol{\beta}^T \hat{\boldsymbol{x}}$$

. Let $p_1(\hat{\boldsymbol{x}}; \boldsymbol{\beta}) = \mathbb{P}(y = 1|\hat{\boldsymbol{x}}, \boldsymbol{\beta})$ and $p_0(\hat{\boldsymbol{x}}, \boldsymbol{\beta}) = \mathbb{P}(y = 0|\hat{\boldsymbol{x}}, \boldsymbol{\beta})$. We can then rewrite the log-likelyhood function:

$$\mathcal{L}(\boldsymbol{\omega}, b) = \sum_{i=1}^{N} ln\Bigg[y_i p_1(\hat{\boldsymbol{x}}; \boldsymbol{\beta}) + (1-y_i) p_0(\hat{\boldsymbol{x}}; \boldsymbol{\beta})\Bigg]$$

Substitute with $\boldsymbol{\beta}$:

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{N}\Bigg[-y_i \boldsymbol{\beta}^T \hat{\boldsymbol{x}} + ln(1+e^{\boldsymbol{\beta}^T \hat{\boldsymbol{x}}})\Bigg]$$

Gradient descent method can solve this function.

Perceptron[3]

Consider an activation function for hard classification problems:

$$sign(a) = \begin{cases} +1, a \geq 0 \\ -1, a \leq 0 \end{cases} \Rightarrow f(x) = sign(\boldsymbol{\omega}^T \boldsymbol{x})$$

We count the wrongly classified $f(x_i)$ and minimize the number of it. Write the loss function:

$$\mathcal{L}(\boldsymbol{\omega}) = \sum_{i=1}^{N} \mathbb{I}\{y_i \boldsymbol{\omega}^T x_i < 0\}$$

We set $D : \{WronglyClassifiedsample\}$, therefore:

$$\mathcal{L}(\boldsymbol{\omega}) = \sum_{i=1}^{N} \mathbb{I}\{y_i \boldsymbol{\omega}^T x_i < 0\} \Rightarrow \mathcal{L}(\boldsymbol{\omega}) = \sum_{x_i \in \mathcal{D}} -y_i \boldsymbol{\omega}^T x_i$$

Gradient descent method solves the function.

Support Vector Machine[4]

Consider a two-group linear separable space with two sub-spaces (input space and feature space). Suppose the training dataset

$$\mathcal{D} : \{(x_i, y_i)\}_{i=1}^{N}$$

where, $x_i \in \mathbb{R}^p, y_i \in \{-1, +1\}$ for $i = 1, 2, ..., N$. Assuming the training dataset is linearly separable, our learning objective is to find a hyperplane that is able to divide the feature space into two parts (positive and negative). The corresponding equation of hyper-plane is:

$$\omega^T x + b = 0$$

where ω is the normal vector of the hyperplane, b is intercept. We measure the accuracy of classification by the distance between points and the hyperplane. We can solve the following convex quadratic programming problem maximize the margin:

$$\underset{w,b}{argmax}[\min_i \frac{|w^T x_i + b|}{||w||}] \ s.t. \ y_i(w^T x_i + b) > 0$$

$$\Longrightarrow \underset{w,b}{argmax}[\min_i \frac{y_i(w^T x_i + b)}{||w||}] \ s.t. \ y_i(w^T x_i + b) > 0$$

[3]According to Minsky and ML, 1990 and some of my derivation

[4]According to Cortes and Vapnik, 1995, Friedman et al., 2001 and some of my derivation

Simplify it:

$$\underset{w,b}{argmin}\,\frac{1}{2}w^Tw \; s.t. \; \min_i y_i(w^Tx_i+b)=1$$

$$\Rightarrow \underset{w,b}{argmin}\,\frac{1}{2}w^Tw \; s.t. \; y_i(w^Tx_i+b)\geq 1, i=1,2,\cdots,N$$

Solving the problem in the high dimensional space, we introduce Lagrange function and solve the its duality. The construction of Lagrange function:

$$L(w,b,\lambda)=\frac{1}{2}w^Tw+\sum_{i=1}^{N}\lambda_i(1-y_i(w^Tx_i+b))$$

The dual problem:

$$\underset{w,b}{argmin}\,\max_{\lambda} L(w,b,\lambda_i) \; s.t. \; \lambda_i\geq 0$$

$$\Longrightarrow \max_{\lambda_i}\min_{w,b} L(w,b,\lambda_i) \; s.t. \; \lambda_i\geq 0$$

Take partial differential of ω, b on $L(w,b,\lambda)$ and set them to 0:

$$\nabla_w L(w,b,\lambda)=w-\sum_{i=1}^{N}\lambda_iy_ix_i=0$$

$$\nabla_b L(w,b,\lambda)=-\sum_{i=1}^{N}\lambda_iy_i=0$$

$$\Longrightarrow \omega=\sum_{i=1}^{N}\lambda_iy_ix_i$$

$$\Longrightarrow \sum_{i=1}^{N}\lambda_iy_i=0$$

Hence, we have:

$$L(w,b,\lambda_i)=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_jy_iy_jx_i^Tx_j+\sum_{i=1}^{N}\lambda_i$$

The dual probelm is then:

$$\max_{\lambda}-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_jy_iy_jx_i^Tx_j+\sum_{i=1}^{N}\lambda_i, \; s.t. \; \lambda_i\geq 0$$

By KKT conditions, we can calculate the optimal parameters:

$$\hat{w}=\sum_{i=1}^{N}\lambda_iy_ix_i$$

$$\hat{b} = y_k - w^T x_k = y_k - \sum_{i=1}^{N} \lambda_i y_i x_i^T x_k, k, 1 - y_k(w^T x_k + b) = 0$$

k Nearest Neighbor[5]

Consider a training dataset:

$$\mathcal{D} : \{(x_i, y_i)\}_{i=1}^{N}$$

where, $x_i \in \mathbb{R}^p, y_i \in \{c_1, c_2, ..., c_K\}$ for $i = 1, 2, ..., N$. x represents eigenvectors and y represents the classes. The objective is to gain the class that x belongs to. Define $N_k(x)$ as the neighbor area that contains k points for x. The basic learning process is to vote for the class for x:

$$y = \underset{c_j}{argmax} \sum_{x_i \in N_k(x)} \mathbb{I}(y_i = c_j), i = 1, 2, ..., N; j = 1, 2, ..., K$$

where $\mathbb{I}$ is indicator function that equals 1 for $y_i = c_j$ and that equals 0 for $y_i \neq c_j$.
Euclidean distance is normally considered when measuring the distance between neighbors. Let feature space $\mathcal{X} \in \mathbb{R}^p$, $x_i, x_j \in \mathcal{X}, x_i(x_i^{(1)}, x_i^{(2)}, ..., x_i^{(p)})^T, x_j = (x_j^{(1)}), x_j^{(2)}, ..., x_j^{(p)})^T$,

$$L_{Euclidean}(x_i, x_j) = \left(\sum_{l=1}^{N} |x_i^{(l)} - x_j^{(l)}|^2 \right)^{\frac{1}{2}}$$

. The value of k, or the number of neighbors will significantly affect the prediction. In practice, we apply cross validation to choose for the optimal k.

[5]According to Cover and Hart, 1967 ,Friedman et al., 2001 and some of my derivation